OCR GCE A

Computer Science

Project

H446-03

Candidate Name: Naglis Slamiskis

Candidate Number: 3207

Centre Number: 17723

Title of project: Logica

Contents

Section 1: Analysis

Problem

Over the past few years, Computer science and development of software and video games have gotten increasingly popular and is a more viable job than before, many students may be interested in computer science. "The power of computer science lies in augmenting our thinking and in its ability to accelerate research exponentially in other areas," says Julio M. Ottino, dean of Northwestern's McCormick School of Engineering. "Even areas as diverse as art, economics, medicine, and political science can benefit from integrating computational thinking into their research and education. The possibilities are endless." Computational thinking is becoming more and more important in society and this quote proves this idea. Even if the students will lose interest, their problem solving, and logical thinking will be essential to their later life and can benefit from these skills a lot. However, the coding aspect may be too complex as they may not have any prior coding knowledge and may also not understand logical concepts yet and computational thinking is a key part of computer programming and understanding how computers logically function. Hence, I am designing a game to introduce logic gates as they are a key part of the GCSE and A Level course. It can also be an excellent way to get into a habit of thinking computationally and advancing the students problem solving skills.

Therefore, the game will teach the students how to construct various logic circuits according to certain expressions which allows the user to solve puzzles to allow you to progress to the next level, this allows the player to develop the key skill of logical thinking through problems consisting of logic gates. This problem is suitable for a game as it immerses the player to try and solve these puzzles and develop logical thinking and ease them into the world of computer science. This approach is best way to teach students as it would be fun for the students to problem solve different logic gate puzzles in a game as opposed to the lesson which may be less immersive.

Stakeholders

My users are going to be schoolteachers who are either computer science specialised teachers or primary school teachers who want to give a fun and challenging puzzle to their students to develop problem solving and computational thinking skills. They want a game that is challenging, fun, good game mechanics and playability. My main teacher client is Mr Flain, head of computer science department in The Saint John Henry Newman Catholic School. They also want a simple game with a simple structure and an understandable tutorial that is easy for young students to understand and is appealing to them through the voxelated graphics of the game. My client has told me that this will be used within a lesson to help engage those students that are struggling with the topic or are not engaged in the topic. It should make it a more fun way of teaching a key but complex topic to a key stage 3 class where many of the class with not progress with Computer Science post year 9, hence developing their logical thinking skills and potentially intrigue them into taking computer science post Year 9. I will also produce a survey in google forms to get a better understanding of what my teacher client wants for his students and make sure I have met user requirements. I will also have a student client for usability to see if it is easy to use and understand the controls, I will also observe to see what aspects of the current version of the game are not user friendly and adjust the features accordingly. This game can be used by teachers in class to have a fun and engaging way of learning logic gates and circuits and is an extremely straightforward way to help the students visualise the gates and how it works in a simple 3D environment.

Platform

The platform I am using to make this game is Unity as it is one the best game development software especially for solo developers and I have already done tutorials to get to grips with Unity.

My clients using this game will have to download the game and launch it from their computer or laptop and will be played from desktop as a PC game.

Research

Introduction

The three methods I am using for research are a Client Questionaries, current solutions, and observations.

I chose to use client questionnaire as a form of research as it allows me to get an insight into what my client wants and why. This also allows me to build my success criteria based on the user requirements and to make sure I know what my client wants so that my client doesn't get an unwanted application.

I decided to choose current solutions as a form of research so that I can research into current games that apply similar logic and puzzles and I am able to take inspiration from how they approached a logical aspect to their games. This information is helpful to my project as it allows me to see what type of games that people enjoy have this aspect implement and current solutions through websites that schools use so I can see how I can merge the aspect of games and websites together and create an application that can be fun and useful in school and a win-win for the teacher and students.

I have also participated in observations as a form of research. I have past papers and see how past GCSE papers have structured their logic questions so I can base my level questions from the GCSE questions, this will help the students have a familiarity with the style of questions and help them get ready for their exam in a fun way.

Client Questionnaire

In this survey I have asked multiple questions to get an idea for what the client requirements are for my project and how I should approach making this game.

1. How do you teach Logic Gates to your students currently? *

At the moment we start in year 9. Here we teach the rules and how to draw AND, NOT and OR gates. Students then combine these up to 2 x levels. IN year 11 we then recap this and build circuits up to 3 levels. Finally in year 12 students add the XOR gate in. Students perform tasks writing truth tables and also drawing circuits, normally using logic.ly

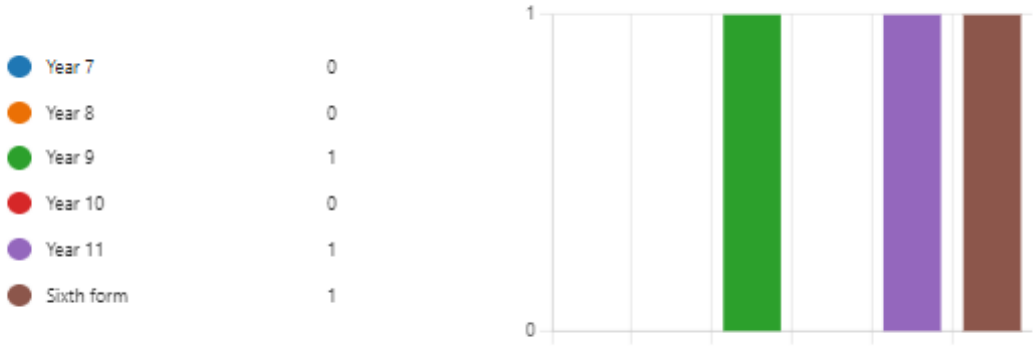
Question 1:

I asked this question to understand how my client teaches logic gates to their students and the difficulty of the circuits needed and also current solutions that my client uses.

The response I got from my client explained the level of difficulties needed for each year group and also current solutions such as logic.ly

2. What year groups will this game be used for?

More Details



3. Would you be willing to set this as homework or a fun task in class

More Details

Question 2:

I asked this question to get an understanding on who to aim my game towards and who will most likely be playing the game and using it

My client responded with the year groups that the game should be aimed. The year groups are Year 9, Year 11 and sixth form

Question 3:

I asked this question to know what environment the game will be used in. For example, if it's used exclusively outside of class then the game can be longer in length and give more leeway for students to explore making logic gates and using certain gates. But in class I will create something mores strict that is aimed more towards teaching the content than letting the students exploring logic gates

As a response I got both so I now know to create versatile gameplay to allow strict and linear gameplay and non-linear gameplay where students can possibly explore logic gates in their own free time and learn what inputs with what gates receive certain outputs.

4. How long do you want the game to be? *

15-20 mins

5. How quick should the learning curve be? (Should the levels go from easy to hard quickly or ease the user in etc.) *

I think it should start by checking students understanding of the main gates e.g. what would the output be for an AND gate if the inputs were T and F etc. Then after for example 5 questions this should then build into simple circuits that combine gates e.g. A AND (NOT B). Eventually moving to harder questions e.g. (A AND B) OR (NOT A AND B). Finally theey should include 3-4 inputs. I think 20 questions would work well e.g. 5 single gate, 5 simple circuit. 5, more complex. 5 using 3-4 inputs

6. What is the most important aspect that should be included in the game that you think that will be important and helpful to your students ? *

The main aspect should be that students will see a circuit and be given the inputs and they need to identify the correct output. It could also show for example a circuit with an output and a missing gate and students have to identify the missing gate.

7. What other key aspects and components that you want in this game for your students? *

I would like it to store the student name and score so that I can track how well they got on.

Question 4:

I asked this question to get an understanding of how long the game should take in general for an average student playing the game and those levels for the first time and base the levels based on the time restrictions

My client's response was 15-20 minutes thus I should make each level fairly easy to understand and quick to complete based on the difficulty.

Question 5:

This question was important to ask because it allows me to influence how fast the students should learn the core logic needed for logic gates and understand the core mechanics and movement for the game itself

The client's response was to include 20 questions / levels each getting more complex every five questions. Therefore, I think I will include 5 levels as a tutorial, another 5 for year 9, another 5 for year 11 and then a final 5 for sixth form with up to 4 inputs. I will make these levels accessible from a in game UI that allows the user to click on their year group and then they will be able to select the level / question that they want to play.

Question 6

I wanted to ask this question to get a better understanding of what the main component and the most necessary feature needed from the client's requirements

The response from my client suggest that the main feature is to be able to see the gates. Hence, I will model each gate for my 3D game view of the circuit thus it can translate into lesson and allow the user/ student to learn the gates in an immersive 3D world.

Question 7:

For question 7 I intend to acknowledge any other key aspects and components that my client wants in this game for their students

In this response from my client, they talk about creating a score system to track the students' progress. This can also be beneficial as it could create some competitiveness which allows the students to spend more time learning these logic gates and thus will improve their understanding due to a score making the students competitive.

Question 8:

I wanted to know what the preferred display size for the client and the application / game.

My client wants a full screen game, possibly so the students will not tab out and get distracted while playing the game.

Question 9:

For question 9, I wanted to know if my client wanted any music to accompany the game and if so, what genre my client would think would suit the game the most

The response from my client was N/A thus no music was needed

10. Any other further ideas or questions?

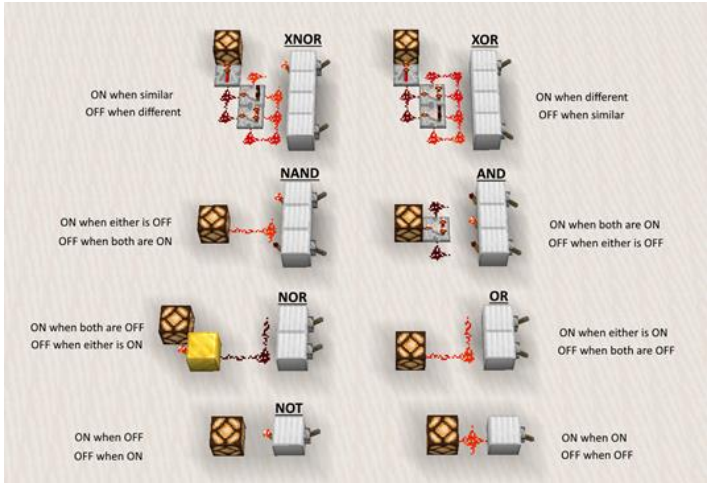
The Karnaugh maps task would not be suitable for Year 9 or 11 as this is something that is learnt in year 12.

Question 10:

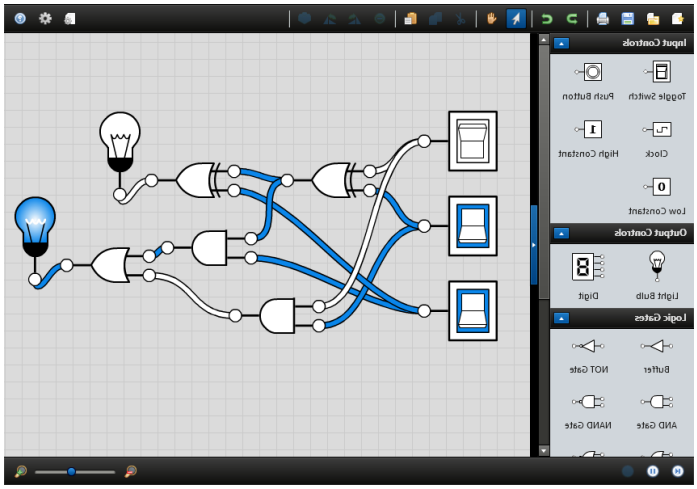
This question was asked to cover any further areas of the game that my client wanted to address and wasn't able to in the previous questions or any other further questions.

In the response my client pointed out that my original idea with having a blank and a completed Karnaugh Maps and create a circuit which will print the outputs onto the blank Karnaugh maps and having the aim to match the two Karnaugh maps to understand the different gates needed and the expressions that are able to be represented in a Karnaugh map and a circuit was not applicable to anyone not in sixth form as Karnaugh maps are introduced then. So now I will create levels such as ones with missing pieces or create a circuit to match the expressions input and output thus meeting the client's requirements.

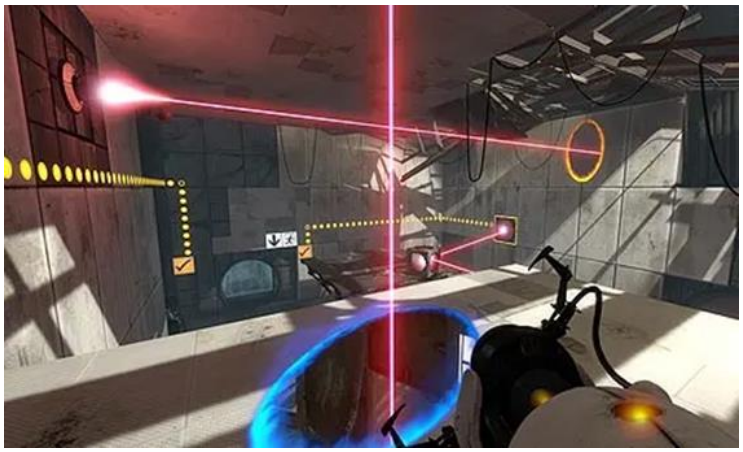
Current Solutions



Minecraft is a 3D sandbox game with procedurally generated worlds using Perlin noise and many different features that engage the player to explore different aspects of the game such as Redstone. This concept uses logic gates in their own form of current with switches and gates. Redstone is the red dust which connects components together much like a wire, a redstone torch which acts like a not gate and can be used as a power source as well, repeaters which allow for long distance redstone as redstone fades after 15 blocks and there are much more different components.



Logic.ly is a website that allows you to construct your own logic gates using various gates, inputs and outputs. This website effectively simplifies logic gates and is easy to use and understand the concept of logic gates and how it works Logic.ly also includes affective visuals to understand when something is on and when something is off throuh the use of blue lights contrasting the white and thus making it pop out and easy to understand. This website also displays logic gates with their different symbols used in computer science which allows the user to learn these symbols and associate this to the different inputs andoutputss from different logic gates



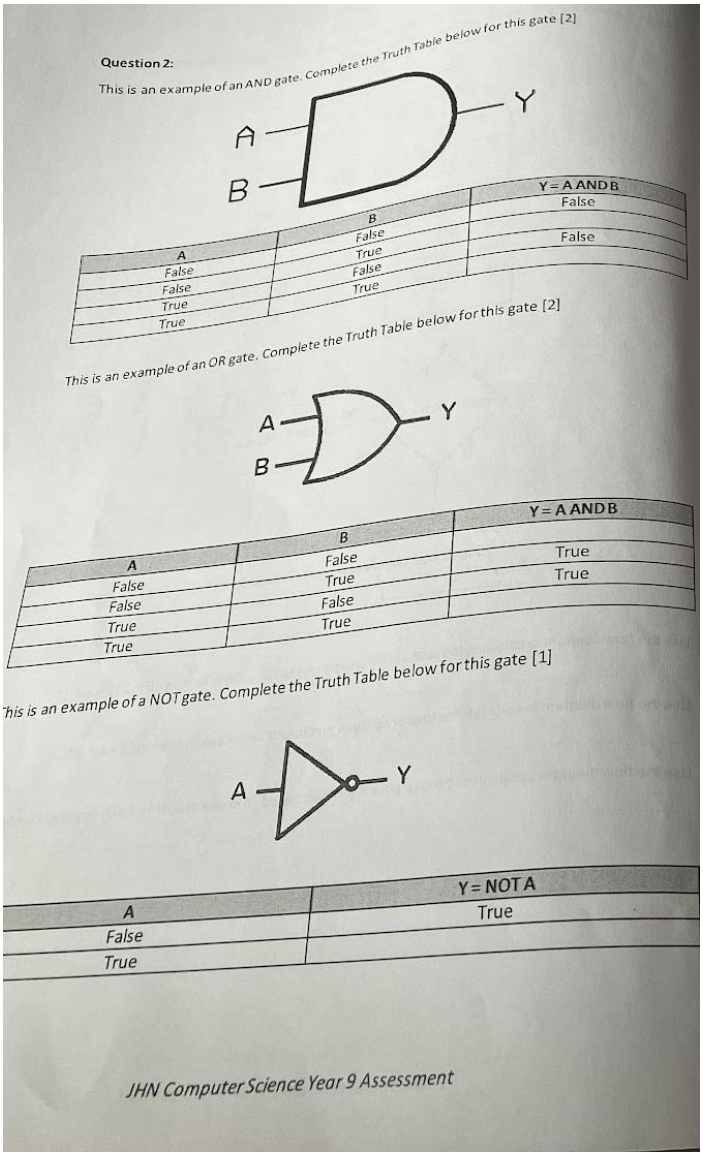
Portal is a 3D puzzle solving game including portals to allow you to get from one area to another just by placing down the portal and entering through the portal. This puzzle solving game has a futuristic, sci, laboratory aesthetic with large white walls and marble looking floors that look like they are made just for the lab work. Solving these problems require lasers and creating portals for the lasers to enter through and reach a specific goal which either opens a door or a new pathway or allows you to go across to a different room

I considered both games and analysed what aspects of the game will have to be explored more and can be beneficial for my project and my stakeholders. These games both have a problem-solving aspect which is largely loved by their respective communities surrounding the games. The simple mechanics can lead to intriguing and interesting problems. I want to create this in my game by having different goals you must achieve by matching two Karnaugh maps (the user and the pre-set) together by creating a logic gate circuit and entering the inputs which is then entered onto the Karnaugh maps. I also want to take inspiration from portal to create a laboratory style graphics to appeal to the audience and immerse them into the problem-solving experience and also use the puzzle aspect to influence my game and create challenging, unique and fun challenges.

Observations

I will talk about the observations I have made through different question papers and how they quiz students on logic gates and how I can apply these forms of questions to my own levels and puzzle logic.

Year 9 question papers:



Question 2:

This is an example of an AND gate. Complete the Truth Table below for this gate [2]

A	B	Y = A AND B
False	False	False
False	True	
True	False	False
True	True	

This is an example of an OR gate. Complete the Truth Table below for this gate [2]

A	B	Y = A AND B
False	False	True
False	True	True
True	False	
True	True	

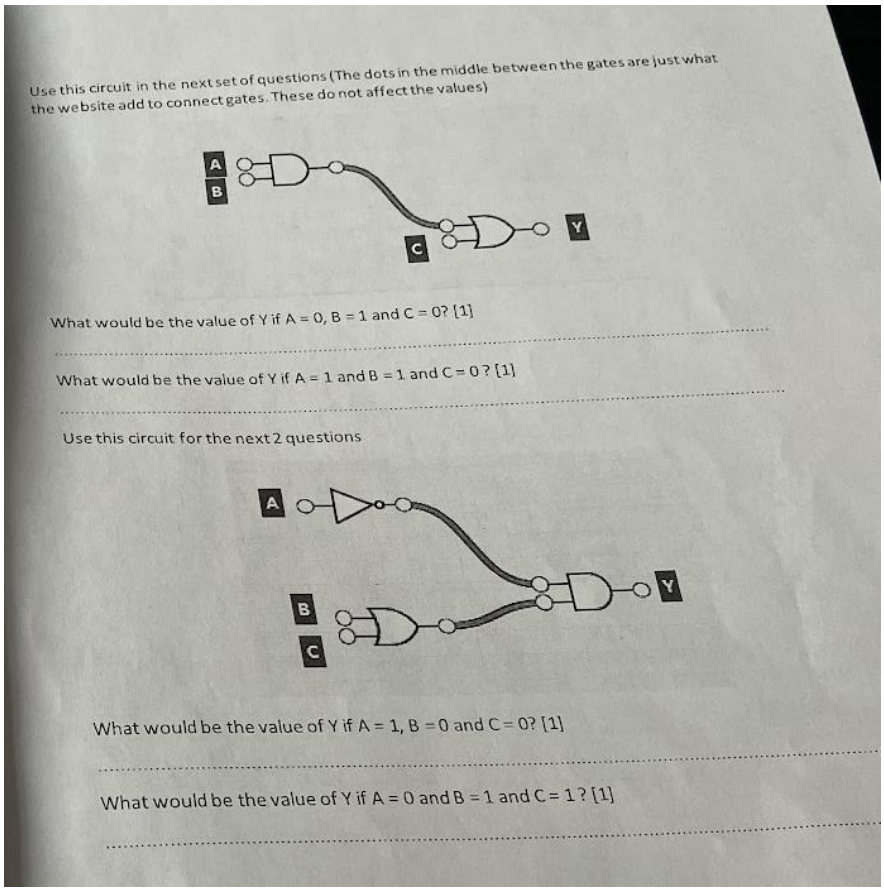
This is an example of a NOT gate. Complete the Truth Table below for this gate [1]

A	Y = NOT A
False	True
True	

JHN Computer Science Year 9 Assessment

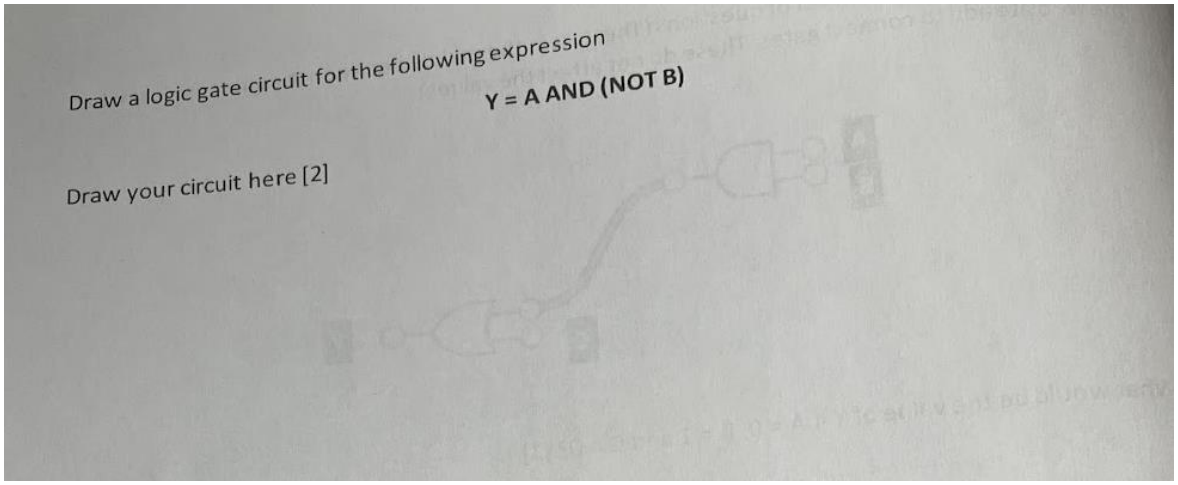
These questions allow the students to fill in the missing blanks which show their understanding in one or two input logic gates and if they are familiar with inputs and outputs of the different basic gates

I can include this in my game by having to click a true or false button based on the logic gate shown and the input shown by the lasers being on and illuminated in front of the student / user.



These questions are showing the students understanding of three input logic gates and if they understand the inputs and outputs of different logic gates and how they work and multiple logic gates connected to each other.

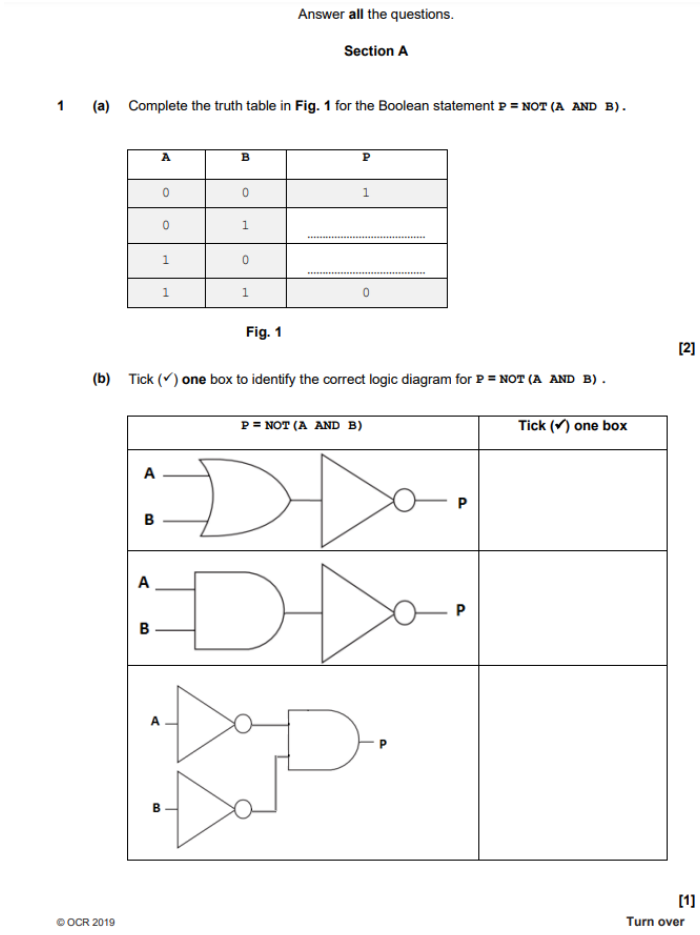
I will try to include this type of question in my project by showing the player a circuit with gates and different inputs and they will have to figure out the output by pressing a button that corresponds to the ‘Y’ being true or ‘Y’ being false. Or I can make the player stand in a certain area of the level which will



This question involves the student making their own logic gate circuit to showcase their understanding of logic gates and how well they can read Boolean expressions

I can include this in my game by showing an expression to the player on a wall or a UI and then having the player to construct a circuit to match the expression this will be checked by running the logic gate circuit and testing all outputs behind the scenes and then checking if it matches the expressions outputs.

Year 11 question papers:



Answer **all** the questions.

Section A

1 **(a)** Complete the truth table in **Fig. 1** for the Boolean statement `P = NOT (A AND B)`.

A	B	P
0	0	1
0	1	.....
1	0	.....
1	1	0

Fig. 1

[2]

(b) Tick (✓) **one** box to identify the correct logic diagram for `P = NOT (A AND B)`.

P = NOT (A AND B)	Tick (✓) one box

[1]

This question for year 11, is testing their understanding of logic gates and if they understand inputs and outputs, it then also checks the student's capabilities of understanding logic circuit diagrams by letting them tick a box from the multiple-choice question below the previous question

My game can include this style of question by letting students interact with a truth table and choose between 0/1 and then include multiple logic gate circuits and let them choose by standing or pressing a button to resemble them choosing that option as the correct answer which emulates the multiple-choice style question

(f) (i) Draw the logic diagram for the logic system **P = A OR (B AND C)**

[3]

(ii) Complete the truth table for the logic system **P = NOT (A OR B)**

A	B	P
0	0	1
0	1	
1	0	

[4]

This exam question in part i) tests the students understanding of the logic gate circuit diagrams and how well they are able to draw this diagram with the correct gates and connections to these gates. In part ii), the questions ask the student to complete the truth table which tests their understanding of outputs and leaving the bottom row blank to see if they understand what other inputs are missing from the truth table and thus testing their understanding.

I can include this question by allowing the player to create their own circuit to match the expression shown either on their screen in a UI or on a wall in a cryptic 'clue/message' style to escape a room etc. I can also include the second part by letting the user interact with a truth table and letting the users choose between 0/1 for each blank cell.

Sixth Form question papers:

8

6 (a) Draw an XOR gate.

[1]

(b) Explain the difference in the function of OR and XOR gates.

.....

.....

.....

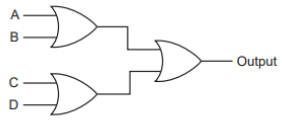
..... [2]

In the sixth form question paper in starts to introduce new gates straying away from the basic gates of 'OR', 'AND' and 'NOT' and tests the student's knowledge of other gates such as 'XOR' as seen in the question paper and then tests the student's knowledge of how the differences between similar logic gates such as 'XOR' and 'OR'.

I can include this in my game to resemble this question in game by allowing the player to use an OR gate and an XOR gate and test all the inputs and outputs so that they understand the XOR gate

9

(c) A circuit contains the logic gates shown below.



(i) Complete the logic table below.

A	B	C	D	Output
1	1	1	1	
1	1	1	0	
1	1	0	1	
1	1	0	0	
1	0	1	1	
1	0	1	0	
1	0	0	1	
1	0	0	0	
0	1	1	1	
0	1	1	0	
0	1	0	1	
0	1	0	0	
0	0	1	1	
0	0	1	0	
0	0	0	1	
0	0	0	0	

[4]

(ii) Complete the Boolean expression below to represent the circuit.

..... ≡ Output

The question in this sixth form paper tests the students understanding of logic gate circuits to its full potential as they are tested on 4 input circuits and what outputs are presented as thus must finish the truth table. Part ii) examines the students' knowledge and if they can recognise the pattern and write out the expression.

I can include this style of question in my game by having an interactive truth table, and also have a popup UI keyboard with special characters just for logic gate expressions this will be checked by comparing strings.

Summary

I have learnt from my Client Questionnaire to have different difficulty for each year group based on the number of inputs and the type of logic gates used within the circuits for example, sixth form levels will use up to 3 or 4 inputs and include NAND gates, NOR gates, XOR gates, and XNOR gates.

I have learnt from my current solutions research, to create something that resembles logic gates in 3D and to create a similar sandbox element as seen in Minecraft's Redstone which has multiple components which can be broken and placed, thus allowing the user to create their own circuits. I also want to have a similar aesthetic and setting from the Portal Games as it is in a laboratory setting with metallic slabs and walls making up the laboratory. I will also use logic.ly as a way to base my 3D designs from their 2D designs and how the different outputs and inputs used.

My research through observations has taught me to approach questions within levels in a different way to conform to the questions used in exams as this will help the students become familiar with the question styles in exams through this game.

Success Criteria

Menu General Criteria	
Testing Criteria	**Justification**
Logica title must be in 04b_19 font, #FF0000 no outline, lowercase	Easy to read will contrast the background. Also, that font matches the aesthetics of the game.

VCR OSD Mono font (same for every menu or text that is on screen). Colour will usually be #FFFFFF. Usually, will be uppercase and have an outline	I chose this font as it perfectly represents the blocky, pixelated theme I am going with. This also will contrast the background and make it easy to read.
White background for all menus apart from inventory	Contrast the text to make it easier to read
All buttons can be pressed and execute the correct action	Provide usability of the UI
All input boxes can be written in	Provide usability of the UI
All scrollable screens or sections must have a scrollbar and be able to scroll via moving the page with your mouse or via the scroll wheel or via moving the scrollbar	Provide usability of the UI, and see more content that was not previously be able to be seen as it does not fit in the screen
All scrollbars must be dependant to where the user is in the page and update the scrollbar	User convenience and provides them a better understanding of the screens size
All drop down menus must show the correct options assigned to them and must also have a smaller menu that will drop down	Provide usability of the UI
All drop down menus should be able to select each option	Provide usability of the UI
Cursor in menus	Allows the user to see where the current mouse position to enable interactivity with UI

Game General Criteria	
Testing Criteria	**Justification**
Each Level must be able to be completed	Allows the student to learn and move on in the levels
Each Level must have a score that is stopped when the level is completed	The teacher will be able to track their students' progress
Each level must have a game over screen if the student does not complete it	The student will have limited time to complete the level so that they don't spend too long and will feel some time pressure like an exam
Each Item in the inventory must be able to be placed and broken via left click and right click respectively	These controls are the same as Minecraft, a popular sandbox game meaning that the controls will be comfortable to most people
When a block is broken and is part of a circuit it will update the circuit	This is to reduce errors and save time building the circuits in case the students mess up
When a block is placed in the middle of a circuit it will update the circuit	This is to reduce errors and save time building the circuits in case the students mess up
Each item in the inventory should be accessible via the scroll wheel	Allows the user to interact with the game
The selected item must be visually represented	The user will be able to see which item they are using, and it will be shown
Pressing WASD allows you to move forward left backward and right respectively	These controls are the most widely used within games so these will be comfortable to anyone

Moving your mouse around will be affected by the user's current sensitivity	Allows user customisation in case the computer cannot handle it
3d levels will be affected by the users currently selected graphics option	Allows user customisation in case the computer cannot handle it
User can press escape to bring up the pause Menu	Allows the user to customise their settings in game and access the main menu
Whenever walking next to a switch, a pop up will be shown that says how to interact with the switch	This increases the usability as it tells the user how to play the game
User can press E when the pop up is active to interact with the switch and it will produce a laser	Allows the user to interact with the game
User can press Q in the 3D level to bring up their inventory and close it	Allows the user to interact with the game and will enable customisability
User can rearrange their inventory when it is open	Let's the user customise the game to their liking
All gates must do their corresponding action with the inputs that are received	Game features must interact with each other to create an immersive experience
Mirrors will bounce the laser in a 90-degree rotation based on its own rotation	To allow users to create more Logic gate like circuits as the lasers can change direction because of mirrors
Lasers will stop when it hits an object unless it is a player	To increase performance since lasers going through walls will have too many objects in the scene.
All levels must have the correct answers to win	To allow the user to not get the correct answer and not be able to move on
Wrong answers will either be told to the user that it is wrong, or nothing will happen	The user will be able to learn from their mistakes
Each 3D level must have a laboratory aesthetic	This will make the experience more immersive and fun to learn in
Crosshair in each 3D level	Allows the user to see where the centre of the screen is to see where they are looking to be able to place or break blocks
Cursor in 2D levels	Allows the user to see where the current mouse position to enable interactivity with UI

Database Criteria	
Testing Criteria	**Justification**
Each table must be created if it does not exist	Database will be in first normal form this is because representing relationships using foreign keys is more flexible, where a hierarchical model only can represent one-to many relationships. This also makes further normalisation possible
Each primary key must be unique	Database will be in first normal form this is because representing relationships using

	foreign keys is more flexible, where a hierarchical model only can represent one-to many relationships. This also makes further normalisation possible
As little redundant or useless data as possible	Database will be in first normal form this is because representing relationships using foreign keys is more flexible, where a hierarchical model only can represent one-to many relationships. This also makes further normalisation possible. Removing useless data saves space and only stores data that is being used hence will abide by GDPR laws.
Each field is atomic, it cannot be broken down further	Database will be in first normal form this is because representing relationships using foreign keys is more flexible, where a hierarchical model only can represent one-to many relationships. This also makes further normalisation possible
The teacher table must be able to store the teacher's data	Allows the teacher to log back in with the data associated with that teacher
Teachers must be linked/ associated with a class	Allows the teacher to view their class scores via the database
Student table must be able to store the student's data and settings	Allows the student to log back in with the data associated with that student
Students must be linked/ associated with a class	Teachers will be able to view the students' progress via a leader board
Students must be linked/ associated with many levels	Teachers will be able to view the students' progress via a leader board in different levels and their scores
Passwords must be hashed	Provide security for the users when storing their passwords in the database
Passwords must be more than 8 characters long	Provide more security for the users when storing their passwords in the database
Passwords must contain a special character	Provide more security for the users when storing their passwords in the database

Limitations

My current limitations within my game are that if it does not immerse the user enough and is not able to effectively teach logic gates in a way that is engaging enough or maybe the user is just not interested in computer science and does not like the puzzle aspect of the game or the problem-solving component as it might not be their cup of tea.

Another limitation I will have been wanting to create a level builder so teachers and my clients can customise levels for their students however I might not be able to create this feature as it will need me to develop and code an effective level builder that is easy to understand and follow which I might not have enough time to develop in this project.

This causes another limitation of replay ability due to the limited levels once a player has done the levels, they are not able to go back and play again as they know the answer. This can be solved by random generation of questions and circuits or a Freeplay area of the game. However, this will take too much time to be completed

I will also be limited as I will not be able to make a website or any form of way to access the download file from the internet hence making this solution a local solution. Therefore, this will only be available in computer science classes where it has been on installed on a school network and will not be available online to download from a file or at home as revision.

I will also need to take into consideration any potential groups of clients who will not be able to use this software. Students with visual impairment and will not be able to navigate the user interface and interact with the game. Therefore, I will contrast the user interface screen and the words to make it more visible to students who may have visual impairment. This will allow them to join in with the class and complete homework without major problems. Another limitation are students who are amputees, this is problem due to most of my inputs being on a keyboard and mouse. To minimize this problem, I will try to limit the inputs to a minimum number of keys and make it more accessible for handicapped students and not stray them away from computer science just because of their disability.

Computational Thinking

My problem that I proposed is suitable to be solved with computational methods such as developing a game since my target audience of users generally play video games, so a video game about computer science and logical puzzles will be a good way to either bring more people into computer science or develop students already studying computer science and making their minds think in a more logical way. Players playing the game can be influenced by the knowledge of a student making this game and want to make one for themselves and can start researching and starting their own path of game development.

The game will be set in a laboratory aesthetic like that of the portal game that I mentioned in my research. This means I will need abstraction to remove the environment since the players will be in a building and will also abstract the inner workings of the circuit components. The levels will be based on being in simple plain white room with a logic gate puzzle to help you leave. I will abstract the details in the graphics and choose to make a smooth and simplistic blocky art styled of game as details aren't important and will save time and have better performance.

This game is a suitable computation solution has it has clear inputs and outputs, e.g., to move which will make the game more immersive. Selecting suitable gates for a given expression all allows the student to immerse themselves within the game and make decisions within the game on what gate fits the expression the best.

Storing the students' score and progress using databases is also a suitable solution to teachers tracking the students' progress in this topic as it allows the teacher to base further questions based on the data given to the teacher. This would be difficult for a human to track for many players/

students and which teacher has what student in their class and what class they belong to furthermore the program will do this automatically and requires no manual sorting into databases.

I will use decomposition to break down the game into smaller tasks:

- Player movement
- Player camera
- Player interaction with the world
- Level UI
- Level progression (e.g., pistons moving to allow parkour and a finish line)
- Object design and animation
- Level design
- Making the logic gates interact with each other properly
- Database Score

Decomposition will make this easier as breaking this big project down into smaller projects will allow me to keep motivation to complete the project as it will feel less overwhelming. It will also let me know which parts of my code are not working and will help me know where the problem is.

The inputs for movement will be WASD and/or the arrow keys. The mouse will be used to look around as it is a 3D environment, and to also interact with the 3D world such as to pick answers that pop up as a UI in certain levels, and also place and break circuit block similar to Minecraft, thus allowing the user to create their own logic gate circuits to either answer a question or mess around and learn on their own. The mouse will also be used to allow the user to interact with UI provided by the game to access levels and exit levels etc.

Hardware/ software requirements

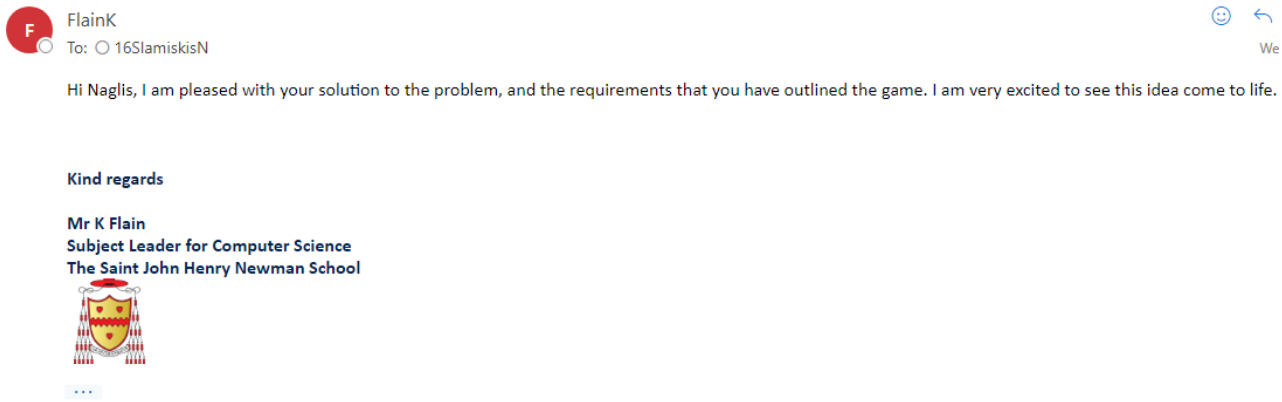
Type	Minimum Requirement	Reason
CPU	X64 architecture with SSE2 instruction set support	Minimum requirements to run Unity
RAM	8GB	Store Unity application
GPU	DX10, DX11, and DX12-capable GPUs	Minimum requirements to run Unity
Hard drive	16GB	Store Unity application
OS	Windows 10	Minimum requirements to run Unity
Game Engine	Unity Game Engine	The application was made in unity

Type	Minimum Requirement	Reason
CPU	Intel Core i5-3330 3Ghz	School's computer CPU
RAM	4 GB DDR3	School's computer RAM
GPU	DX10, DX11, and DX12-capable GPUs	Minimum requirements to run Unity
Hard drive	5GB	Store game application
OS	Windows 7	School's might not have windows 10

Peripherals:

- Mouse with working scroll wheel
- Keyboard
- Monitor

Client sign off

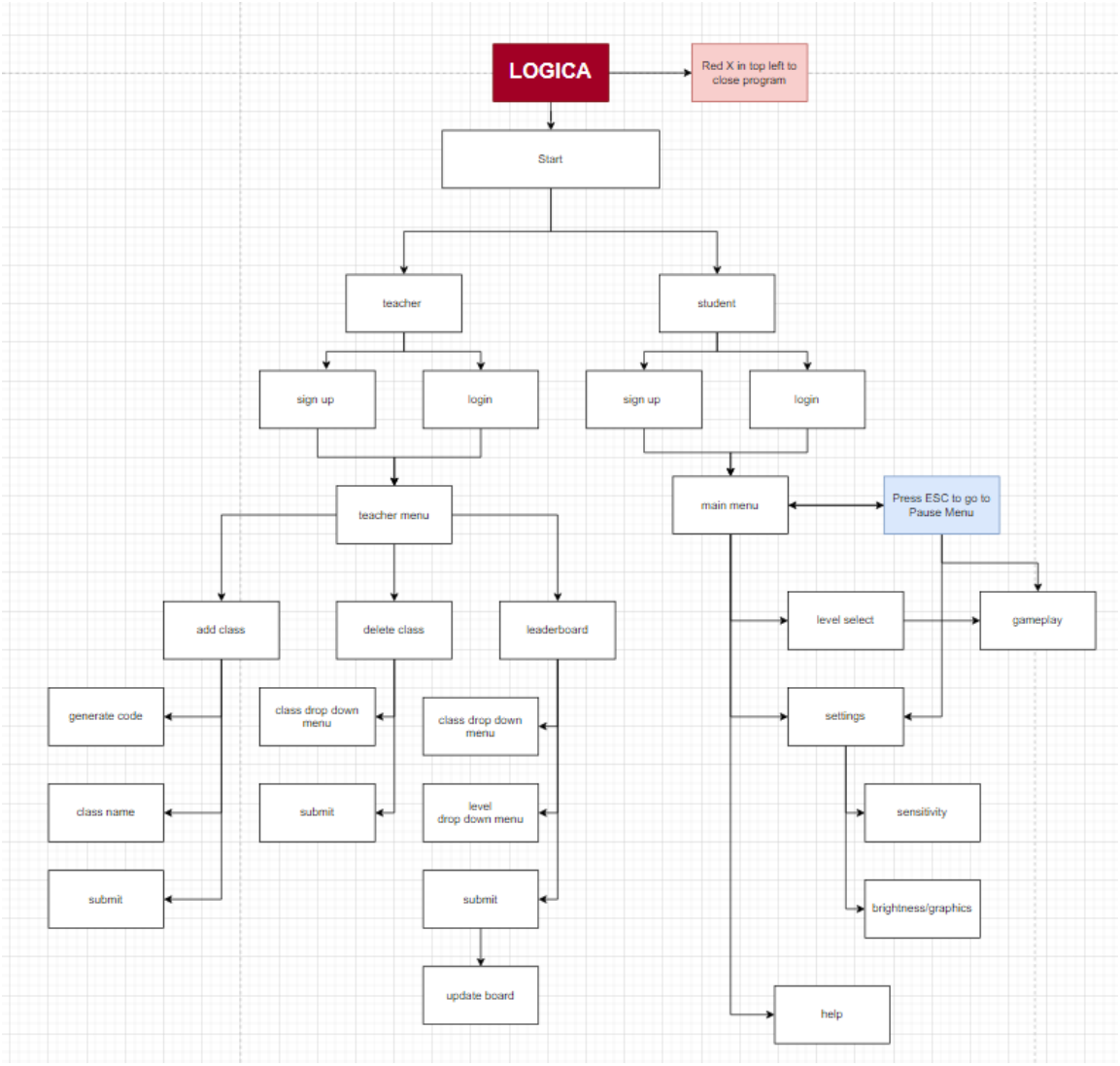


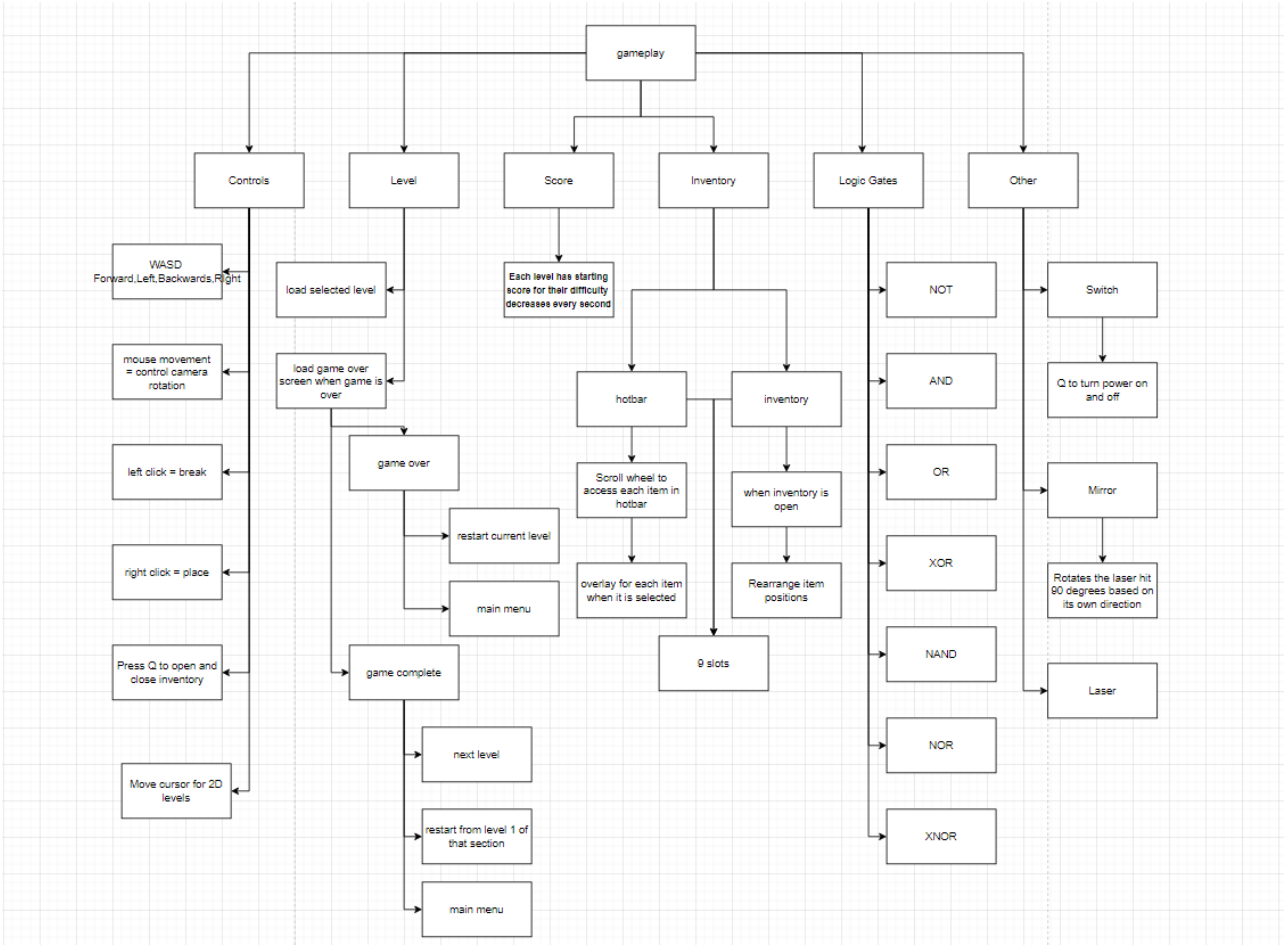
Section 2: Design

Systems Diagram

I chose to do top-down diagrams, because it simply shows each process needed to be made within the game. It is easy for me as the developer to read, and it is also easy for the client to understand in case they want to make any changes.

I decomposed each section into smaller subsections so that I can work in a modular fashion when designing and developing my program

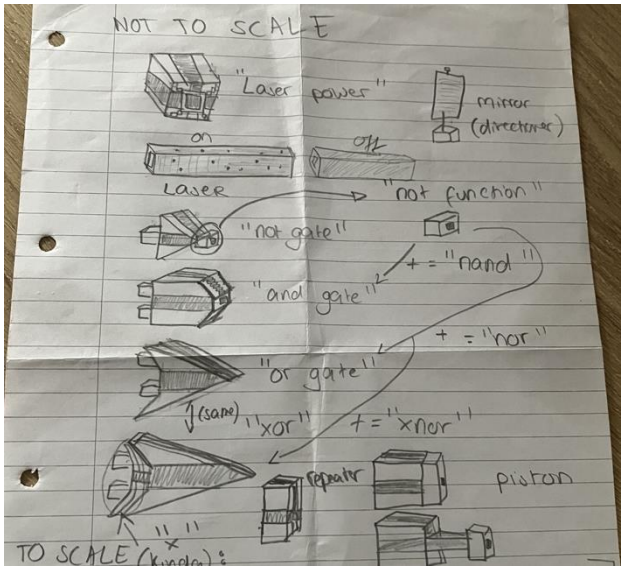

LOGICA
Red X in top left to close program
Start
teacher
student
sign up
login
sign up
login
teacher menu
main menu
Press ESC to go to Pause Menu
add class
delete class
leaderboard
level select
gameplay
generate code
class drop down menu
class drop down menu
settings
class name
submit
level drop down menu
sensitivity
submit
submit
brightness/graphics
update board
help



Above shows two photos of the overall top-down design for my program. Showcasing the main sections and functionality I will have to develop for my game.

For each section I will design code or include something related to it in my code and design of UI.

Logic Gate design



I decided to go for a cube / uncircular feel to my game like Minecraft as it will be kid friendly graphics and look simple to not confuse the students that are playing the game so they can just focus on the completing levels. The main colour scheme will be white and red with some black to separate the two colours. I chose this because I wanted to give it a laboratory aesthetic.

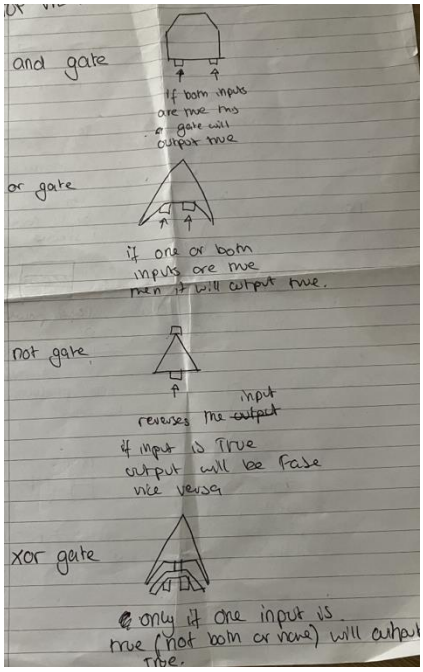
I chose this design for the Laser Power to act as a switch or a source of energy, so I wanted to make sure it looked powerful enough to store this amount of energy.

I then decided to use lasers as the current, I chose this because I think that it will intrigue the students as it looks sci-fi and interesting to the students. I chose this design for the lasers to have a visible on and off state that the students can clearly see. When visible the tube will be filled with red and particles 'flowing' through it.

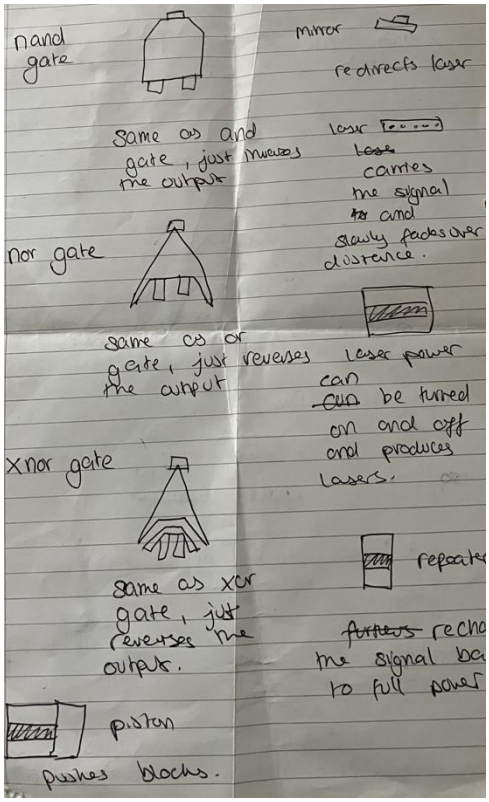
The Logic Gates I tried to keep in line with the original symbols for the logic gates and make them 3D, but I have tweaked them a bit to not contain circular lines as that is the games aesthetic (e.g., not gate has a rectangular at the end instead of a circle)

Piston is an output point you want to reach; the pistons will allow you to move onto the next level when activated. I chose this design, heavily inspired by the Minecraft piston, because it is an effective way to show a 'block' moving due to the extension being made.

I have decided to scrap the repeater and mirror as originally, I was planning for the lasers to be 'emitted' by the laser power block and redirected with mirrors by 90° instead of the lasers being manually placed. In the end I decided for the lasers to be manually placed as it is much more user friendly and less confusing to deal with.



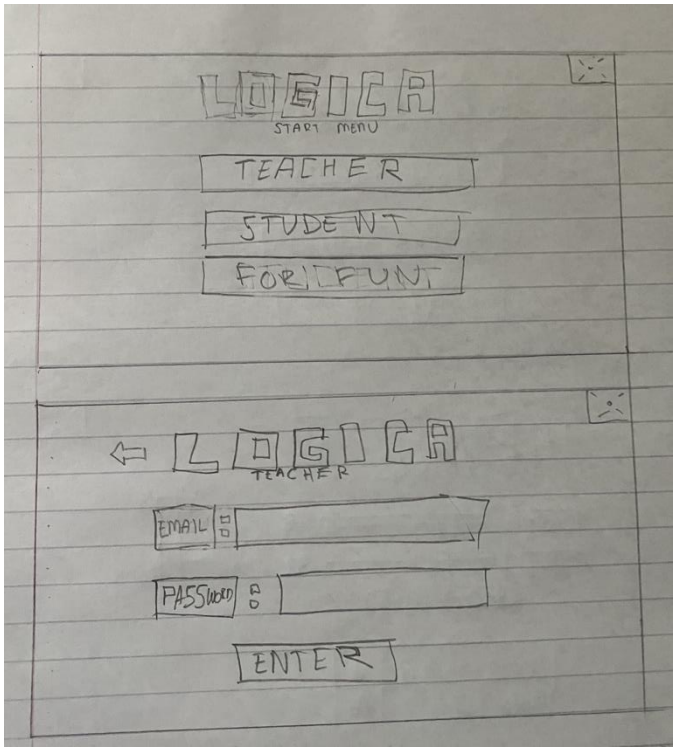
AND GATE = IF both inputs are true this gate will output true
OR GATE = IF one or both inputs are true this gate will output true
NOT GATE = INVERSES the output (input = true output = false, input false output true)
XOR GATE = IF only one input is true this gate will output true



NAND GATE = SAME as AND GATE but outputs the opposite output due to the not function
NOR GATE = SAME as OR GATE but outputs the opposite output due to the not function
XNOR GATE = SAME as XOR GATE but outputs the opposite output due to the not function
LASERS = CARRIES the signal / data between inputs, logic gates and outputs
LASER POWER = CAN be turned on and off, this powers the lasers.

User Interface

Start Menu and Teacher Register Screen



The 'logica' title is in the **04b_19** font, #FF0000, no outline. All other text is made using **VCR OSD Mono** font (same for every menu or text that is on screen). I chose these fonts as they perfectly represent the blocky, pixelated theme I am going with.

The background in these menus will have a panning video of a random level faded in with a black screen so that the text stands out

All buttons and boxes will have white background / fill with black text and a black outline, this stands out and is readable, it is also simple and minimalist which fits the aesthetic

All screens size will be full screen as stated by the client requirements

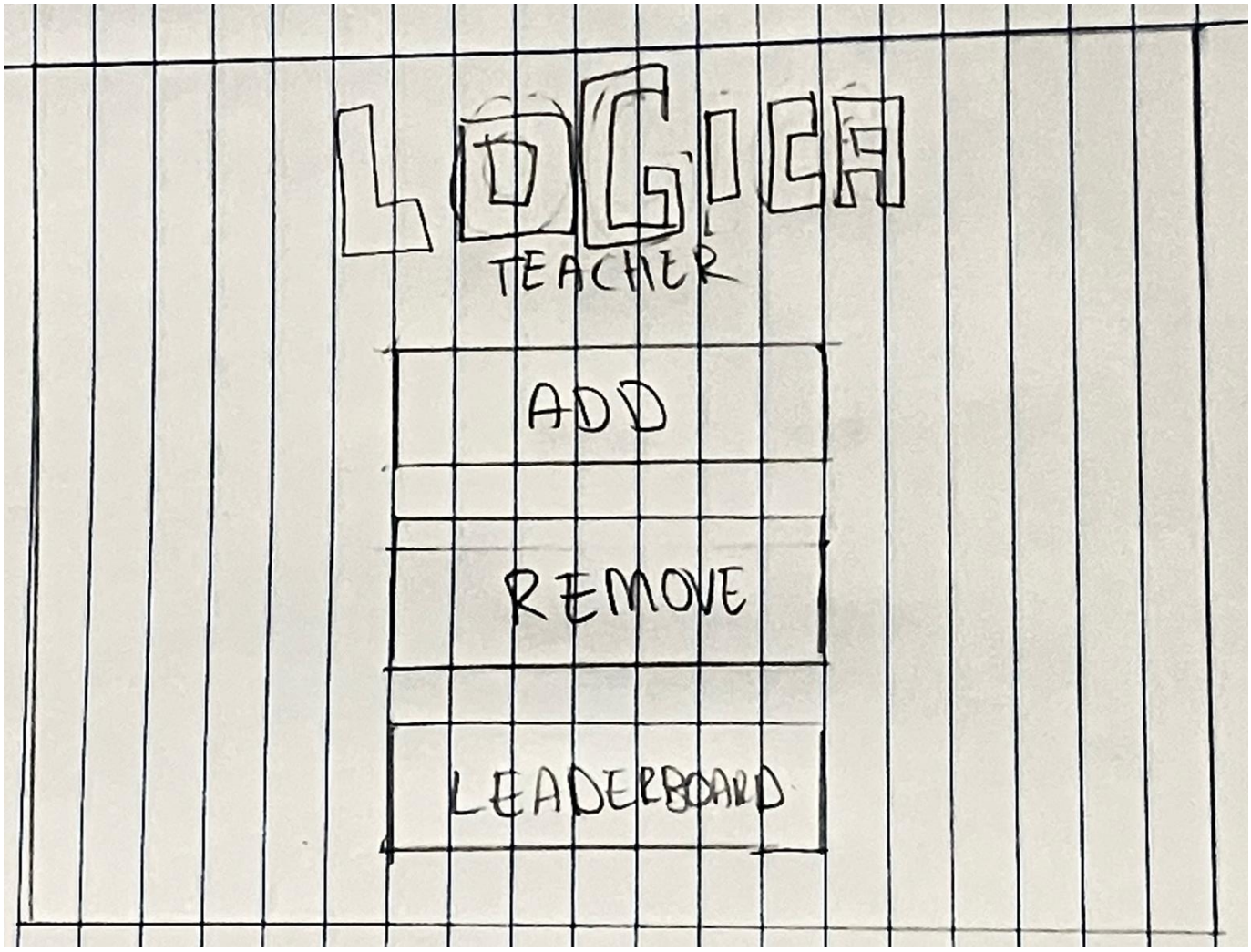
All interfaces apart from the inventory and in game UI overlay, will have a red button with a black X and a dot in the middle as seen in the drawing this will exit to desktop. Almost All interfaces will have back arrow, this will be white with black outline

The first UI screen contains a teacher button that redirects to a login page for them as seen in the second screen, student button redirects to student login page, for fun button will redirect to level selection

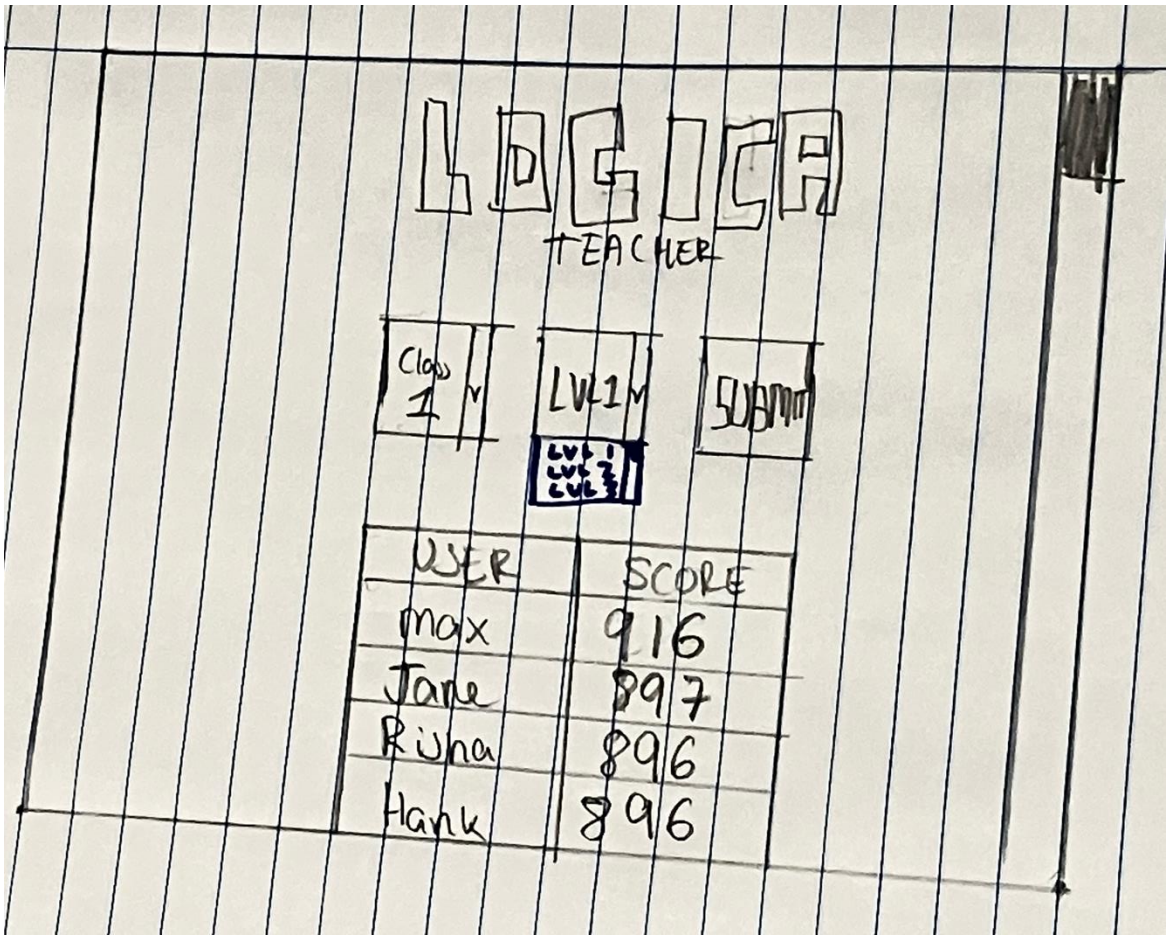
The second UI screen allows the teacher to enter their email and password to login or sign up into their account, after a successful login this page will be redirected to a Teacher Menu where they will be able to see the leader boards for their classes in different years

Teacher Menu

This is the teacher menu screen. There are three buttons. Each of them leads to a separate page; to the add page, remove page and the leader board page. The background is white (#FFFFFF or 255,255,255)



Teacher View Leaderboard Menu

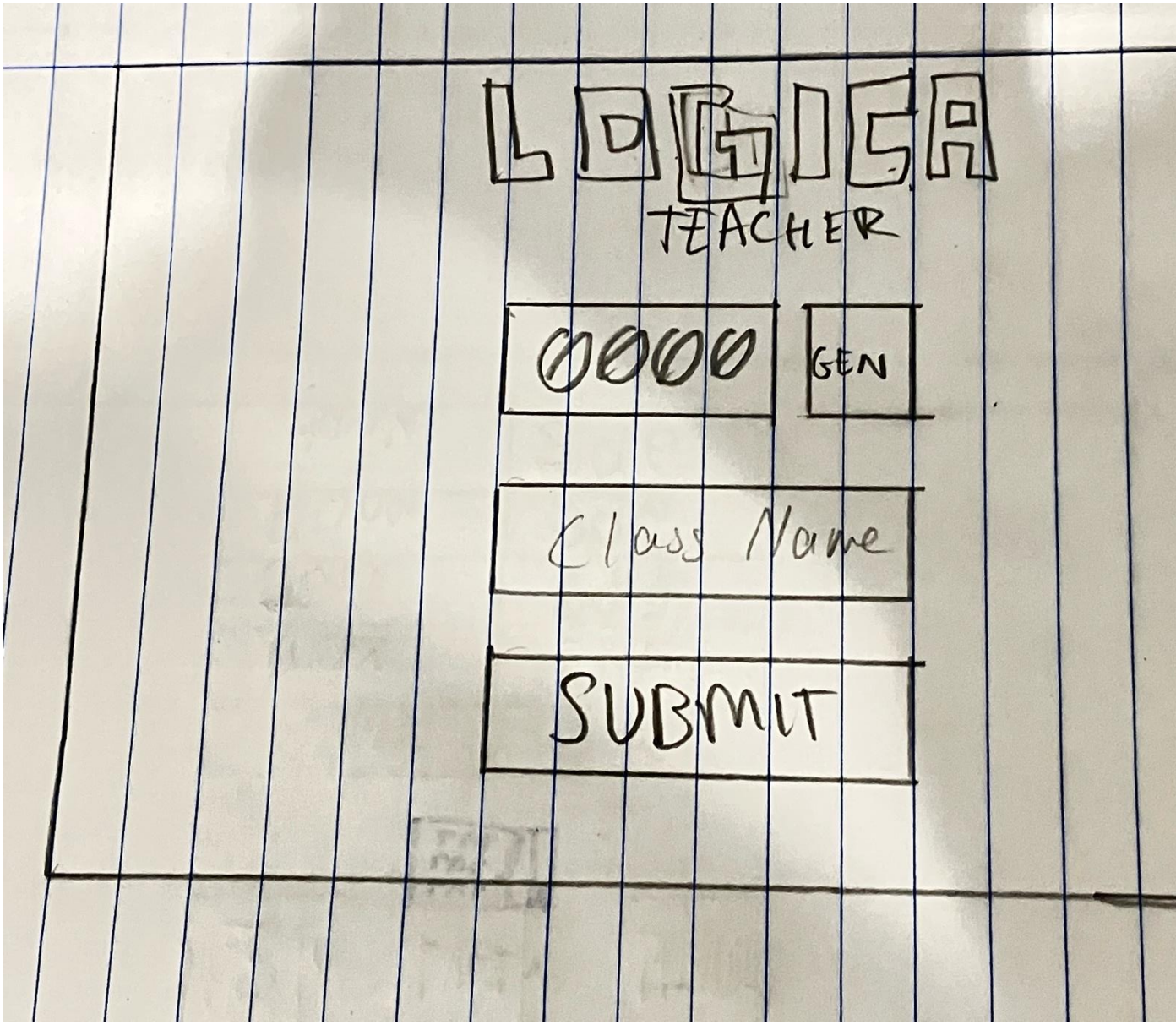


This UI screen is the leader board page. It has two drop-down menus for the teacher to select and choose what level from what class they want to view the scores., these classes will be retrieved from a database and update the options when you load the screen. After they press the submit button, the leader board will update and show the scores in descending order due to a higher sore meaning a quicker time. There will also be a scroll bar due to the page not being big enough to fit all the students. The drop-down bit in blue is what it will look like when you press the button to show the other options it will also have a scrollbar due to it looking better than having all the options show at once. This will not be blue and will have the same colour scheme as the buttons. The background is white (#FFFFFF or 255,255,255)

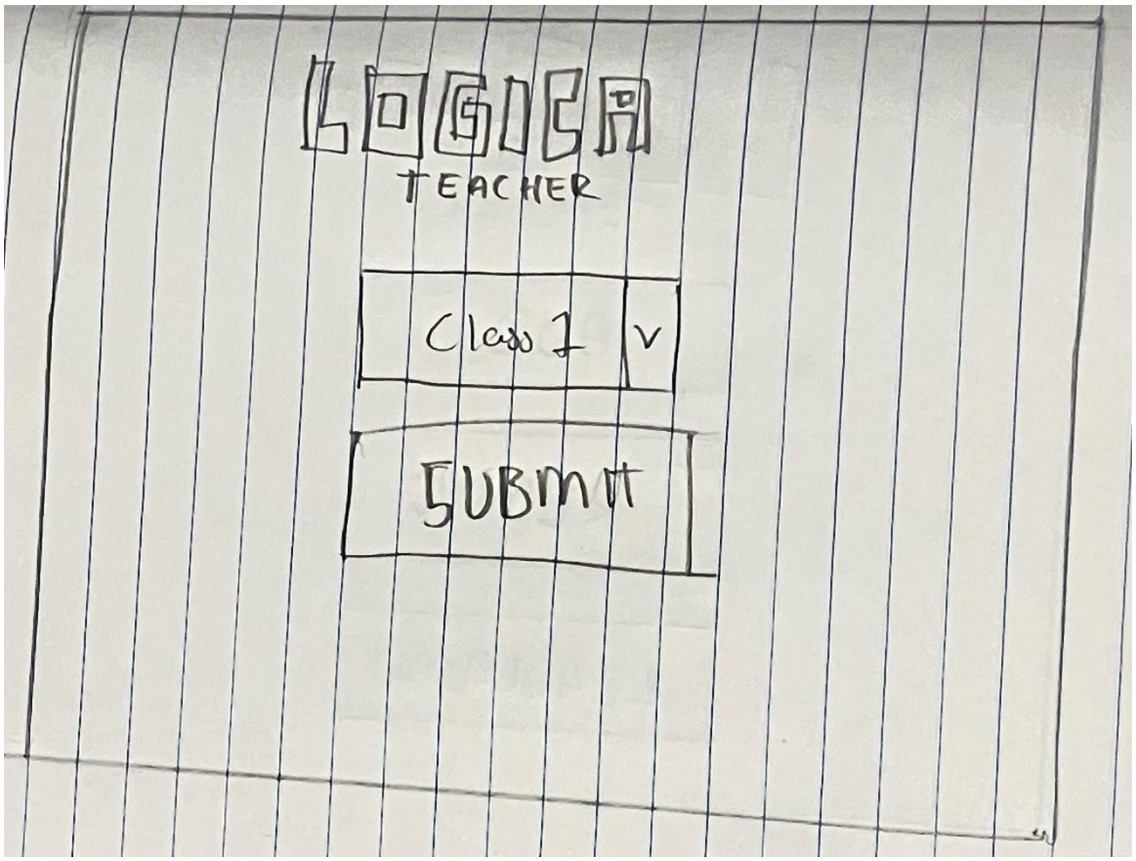
Teacher Add Class Screen

This is the Add a class page where you can generate a 4-digit number to act as a primary key for the class in the database. This will generate and show on your screen instead of the 4 0's. The teacher can then tell this code to their students whenever they want to them to join their class. There is an input box that has opacified text saying class name, so the teacher is able to tell what it is. This will also be added to the database when they submit.

Once the submit button is pressed it will quickly turn red and switch the text to say submitted.

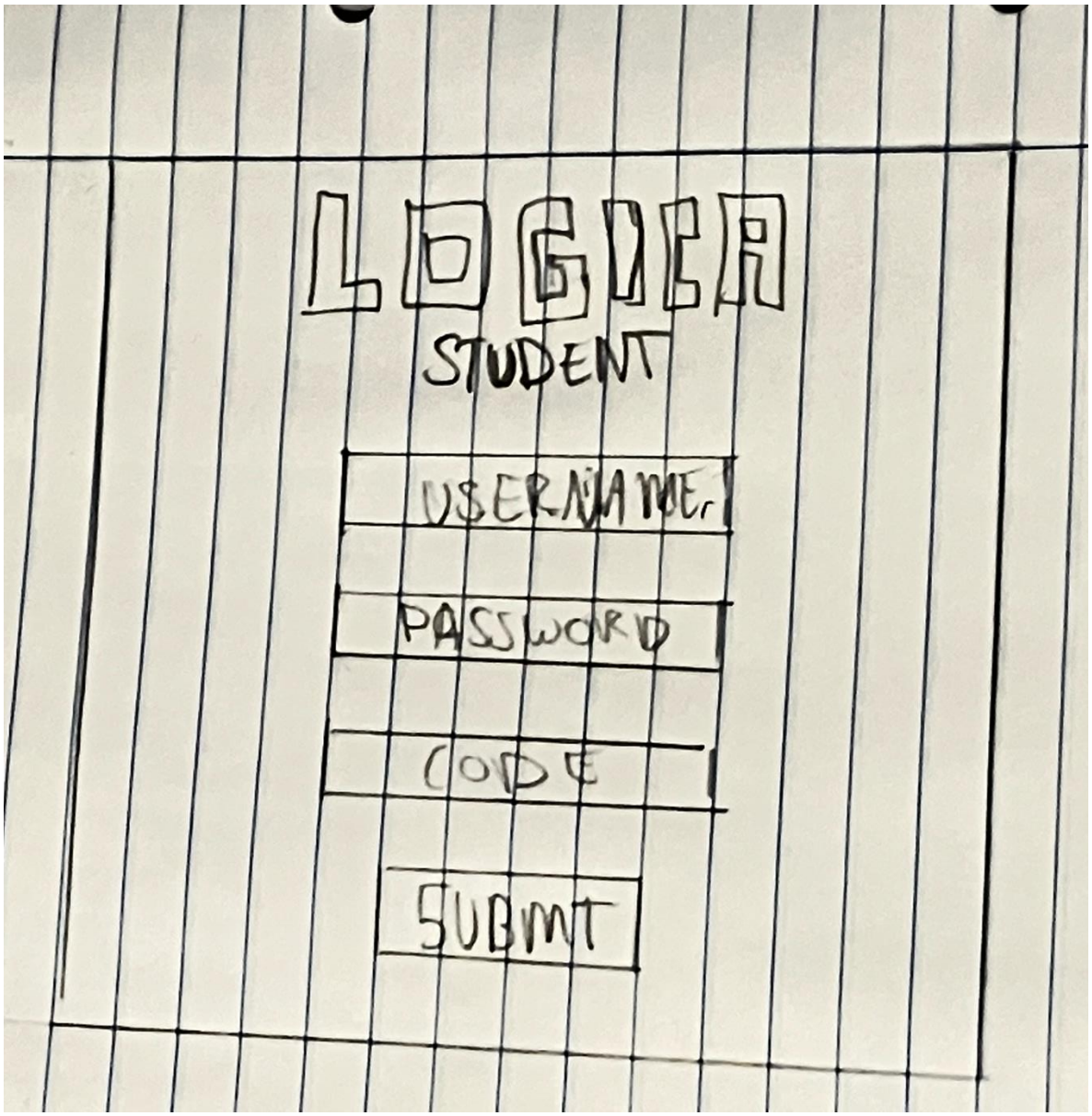


Teacher Remove Class Menu



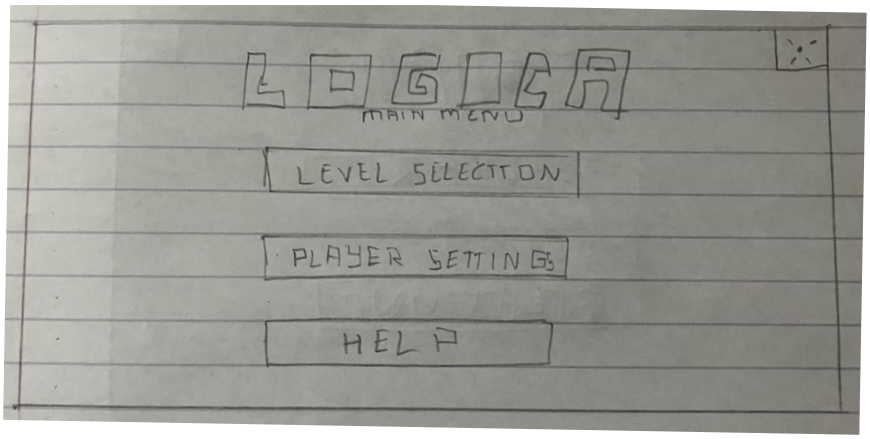
This is the remove page. It will retrieve all the classes of the current teacher logged in and add those to the options. The submit button will flash green quickly when pressed and will delete the current class selected. It will then update the drop-down menu.

Student Register Screen



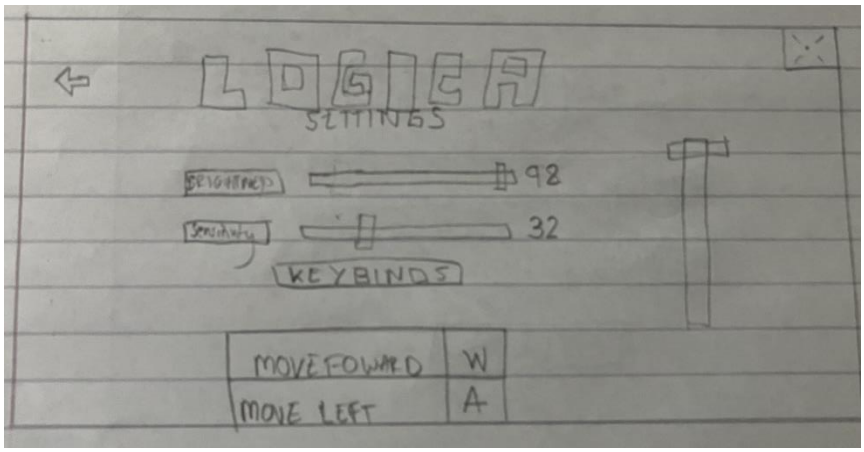
This student page allows the student to log in or sign up. This will join them to the teacher's class via the code and also store the students in a database with their settings and scores so that they will be kept when logged back in. Each input box will have translucent text as shown telling the student what to type.

Main Menu



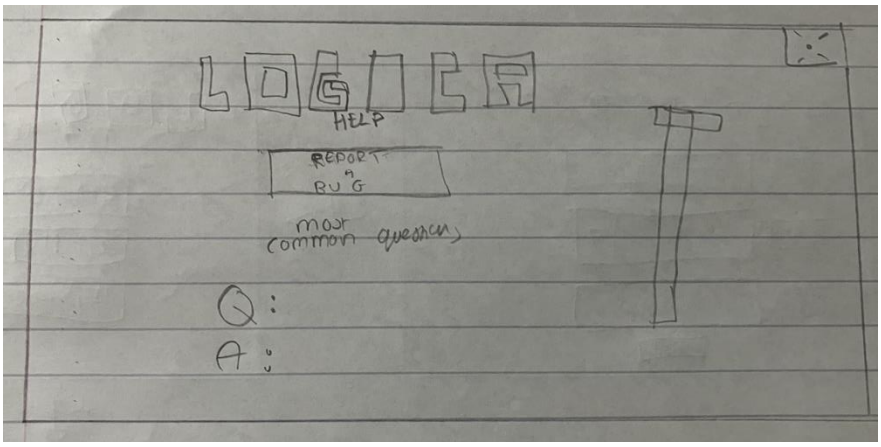
This is the main menu screen, this allows the user to press a button to the level selection menu, a button that redirects them to their specific player settings that will be stored in a database or the help page in case the user needs help figuring out how to play or what each gate does.

Settings



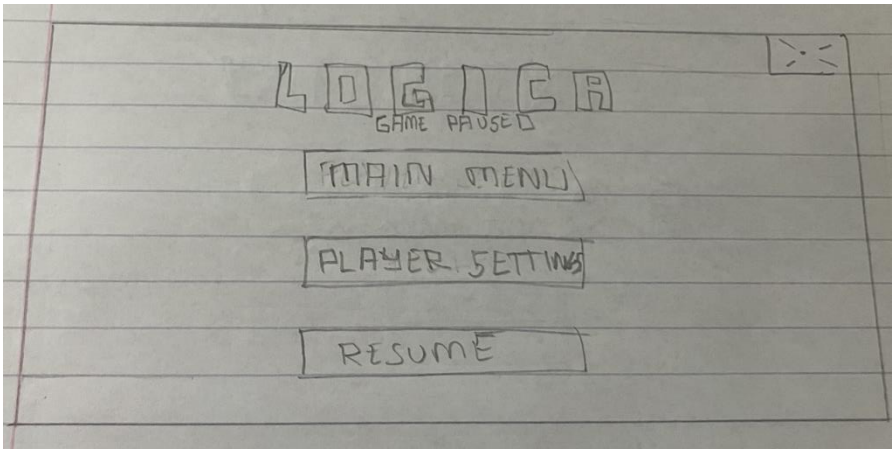
This UI screen stores player data about their in-game preferences. There are two grey sliders with black outlines. The first is for the game's brightness in case they want to reduce electricity or eye strain. Sensitivity can also be adjusted using the sliders to the user's preference. The user can also customise their key binds by pressing on the button with the action labelled that they want to change. There is also a grey with a black outline scroll slider that updates based on how far down the page the user is.

Help Menu



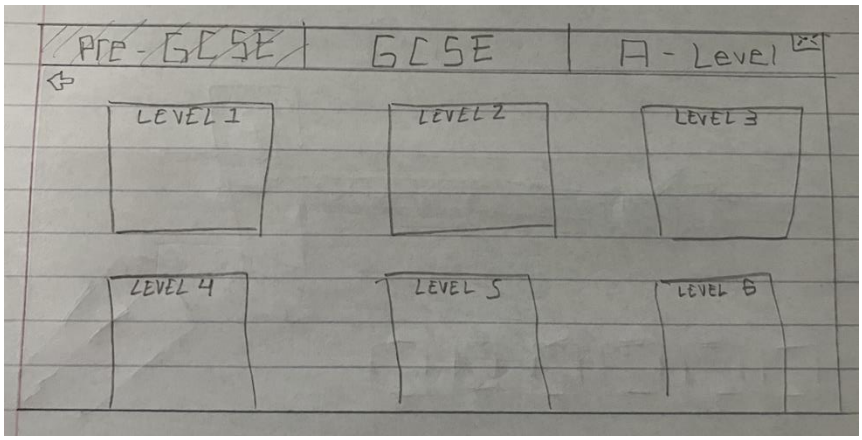
This user UI screen has a grey scroll slider and a button to redirect them to a report a bug page, underneath there are commonly asked questions with the questions and answers as shown above.

Pause Menu



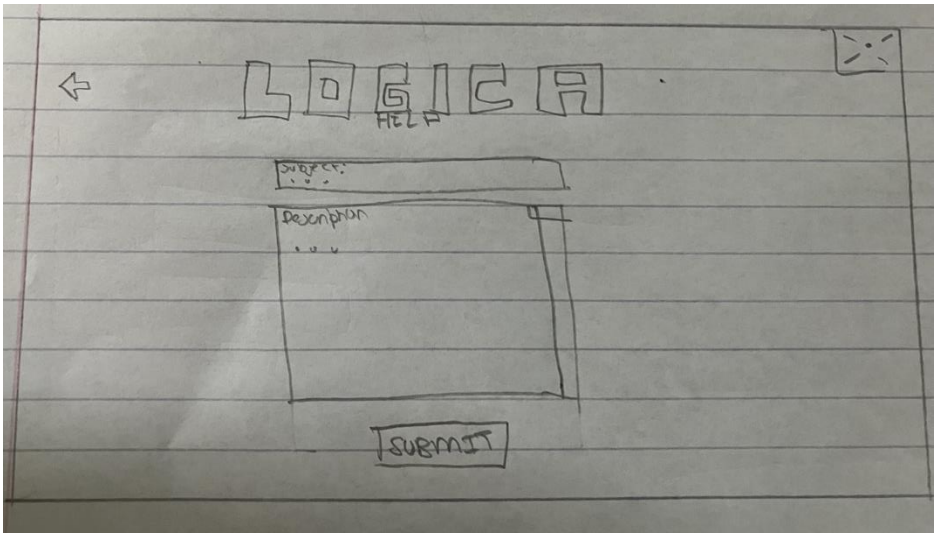
This is the pause menu accessed when the player hits the ESC key, top button redirects to main menu, second button redirects to player settings for that specific person and resume button carries the game on. The background is a freeze frame of the previous frame before the game was paused with a black overlay opacified over it

Level Selection



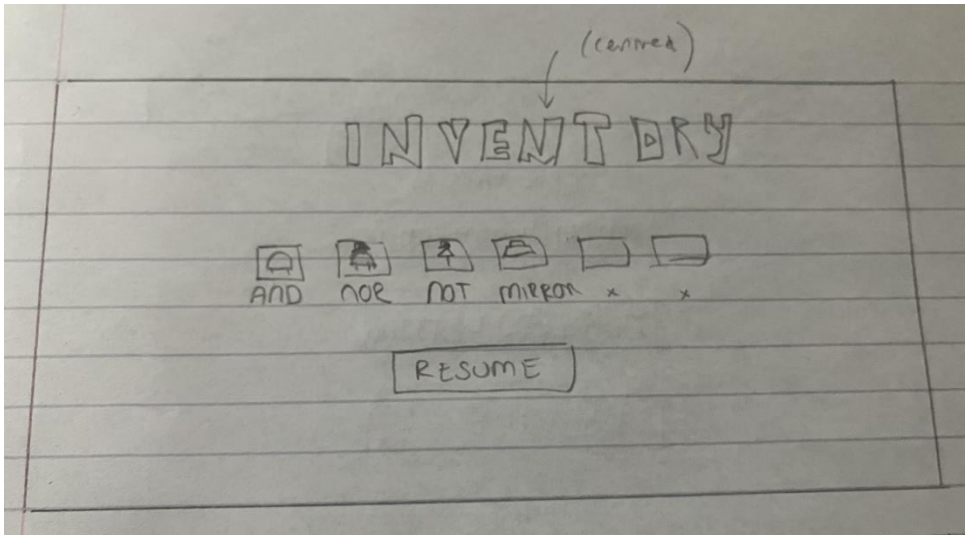
This is the level select screen with three tabs leading to different difficulties of levels based on the year groups. Regular tabs have #d9d7d7 background with black text and outline however the background turns completely white when selected. Each level has a square photo of that level with the white text and black outline over it in the VCR OSD Mono font. Each tab at the top can be clicked to access other levels. It will indicate which tab the game as the tabs will be red (FF0000, or 255,000,000)

Report A Bug Screen



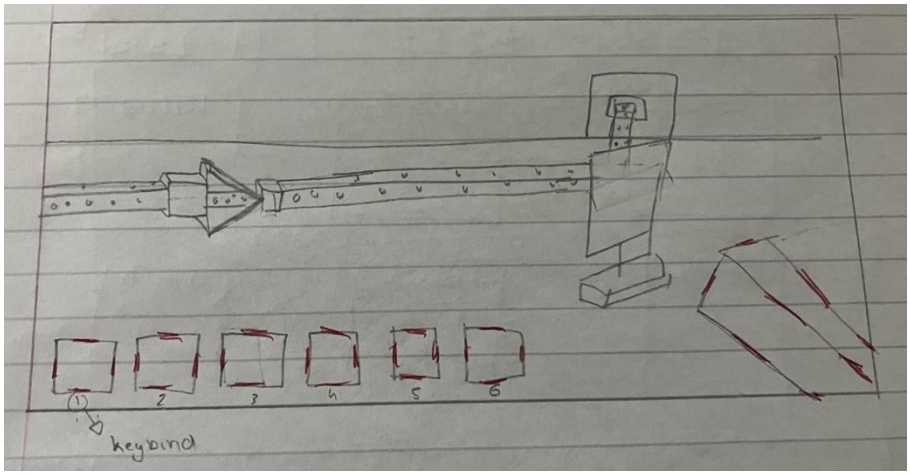
This is the report a bug page where the user can enter a bug and then provide a description of the bug and the press the submit button, The subject and description boxes are the same as buttons or other boxes and have black text. There is also a grey scroll slider with black outlines. It will also tell the user how to play the game and also what each gate does hence teaching them about logic gates and solving the problem using computational methods

Inventory Overlay



This is the inventory UI screen, the inventory title has the same style as the ‘logica’ title in other menus, the boxes containing the items have white backgrounds and a black outline, the text underneath these slots are the VCR OSD Mono font, white text and black outline and the items in the slots are the original 2D symbols used in exams. The background is a freeze frame of the previous frame before the game was paused with a black overlay opacified over it.

In-Game Overlay



In game overlay showcased in red. Inventory slots are the same as inventory UI key binds are shown in the VCR OSD Mono font underneath the slot boxes and the arm is a basic arm with long sleeve clothing so no skin is showing to not discriminate. There should also be a score

Level Design

I decided to use exam questions to allow the Teacher to monitor the students' progress in exam style questions along with the intended 3D levels that are more interactive and fun but still inspired by exam style questions. I emailed my client to find out what their requirements are for levels.

1 16SlamiskisN
To: FlainK
Wed 15/03/2023

Afternoon Sir,

Are there any specific questions that you want in my logic gates game. I need questions that involve building a circuit for pre-GCSE, GCSE and ALevel, for example building a circuit based on a certain karnaugh map.

Thank you,

Naglis

F FlainK
To: 16SlamiskisN
Wed 15/03/2023

I woudl start with e.g.

Identify the NOT gate, identify the AND gate etc

Then move to what would be the output of the following expression e.g. A AND B if A was 1 and B was 0

Then build up the complexity of the questions e.g. (A OR B) AND C etc

Kind regards

Mr K Flain
Subject Leader for Computer Science
The Saint John Henry Newman School

To make sure that the levels I design will suit these requirements I chose different exam questions to act as the 2d levels for the game.

Pre-GCSE

I started with Pre GCSE questions, these questions are aimed at Years 7-9, mostly Year 9.

5

3 The logic diagram below (Fig. 2) shows a system made up of two connected logic gates.

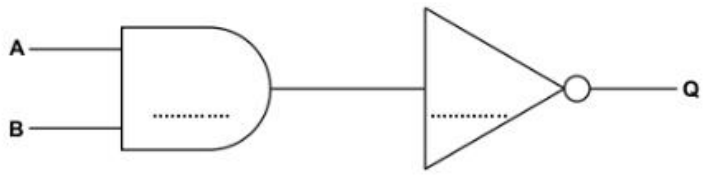


Fig. 2

(a) (i) Label the names of the two gates on the diagram above.

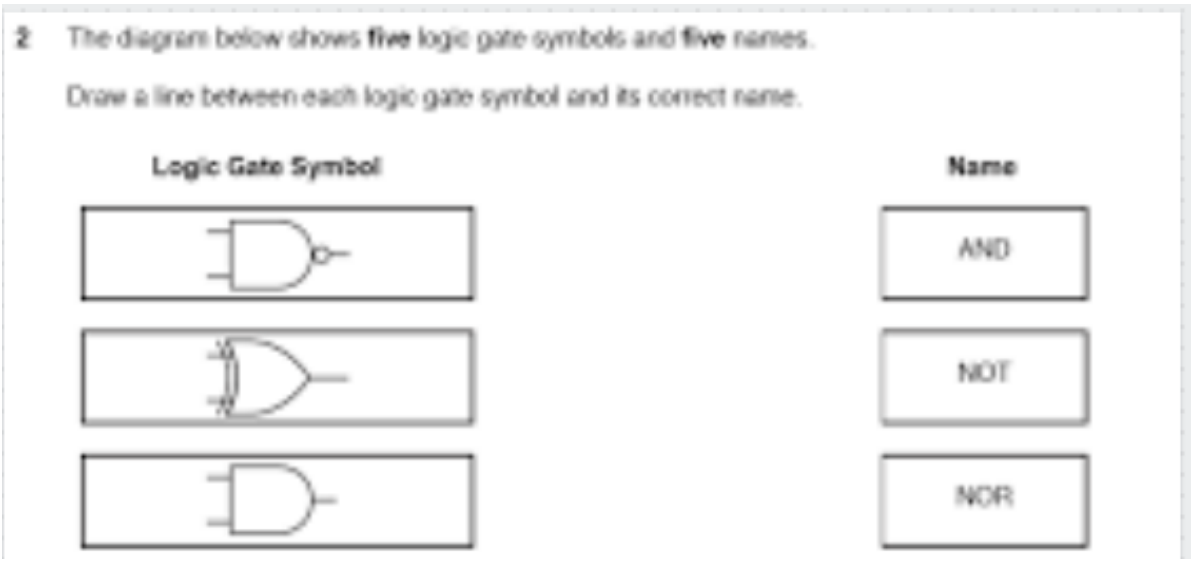
(ii) Complete the truth table below to show the output from this logic system.

A	B	Q
0	0	
0	1	
1	0	
1	1	

(b) Draw the logic diagram represented by **Q = A ∨ ¬B**

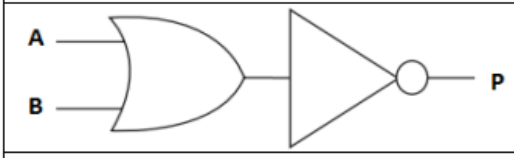
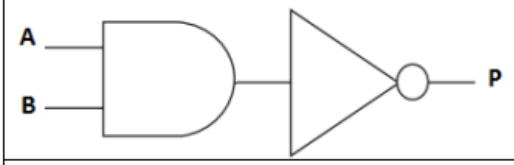
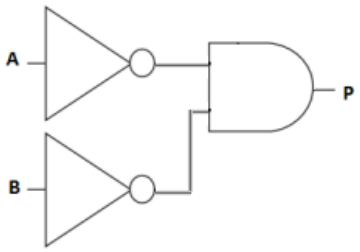
I chose this question for level one as it will effectively test the students base knowldege of logic gates hence giving the teacher a good understanding as to where the student is in their knowledge of logic gates. I will ues part Aii in one of my questions to test their understanding of inputs and outputs.

I will then use part B in my 3D level so that the student will have to create the circuit using the 3D gates



I will also use this exam question to let the teacher know if students are able to identify different logic gate symbols.

(b) Tick (✓) **one** box to identify the correct logic diagram for **P = NOT (A AND B)** .

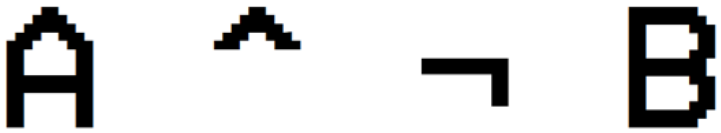
P = NOT (A AND B)	Tick (✓) one box
	
	
	

[1]

I chose this question as it tests their understanding of logic gate circuits and symbols. This question links to the first question and will test students on their ability to interpret expressions

Next for the 3D levels I made expressions that the students will be able to have to read and build the corresponding expression into a circuit.

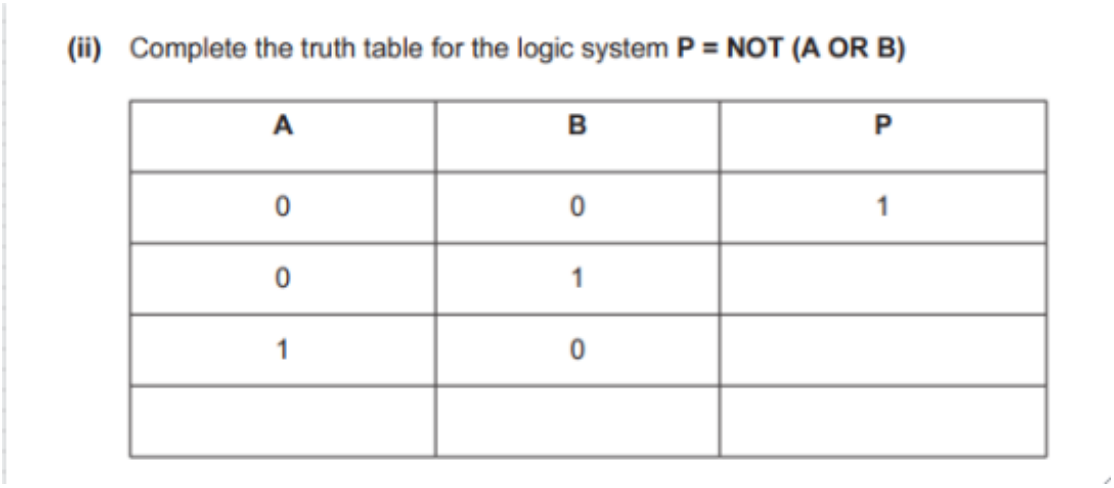
One of the expressions I came up with is shown below



This question will test the students understanding of how to read expressions and if they are able to convert them into a logic gate circuit. This is a very simple question to make sure the player understands how the program works when playing and will slowly ease them into the game's environment.

The last question shown above, Level 6, I introduced the concept of another switch that is needed to be involved. I also started to raise the complexity so the further down the levels the students go, to make it more challenging and entertaining to play.

GCSE



(ii) Complete the truth table for the logic system **P = NOT (A OR B)**

A	B	P
0	0	1
0	1	
1	0	

This question tests the student's ability to interpret expressions and convert them into a truth table using 1’s for True and 0’s for False.

A watering sprinkler (W) is on when the timer (T) is set, the temperature (C) exceeds 26° Celsius, and the rain sensor (R) does not detect rainfall.

Select the logic circuit diagram that corresponds to the circuit for the sprinkler.

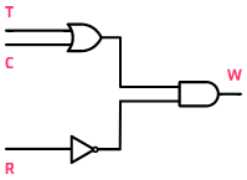


Figure 1: Option A circuit diagram

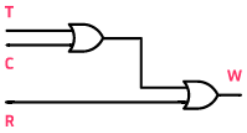


Figure 2: Option B circuit diagram

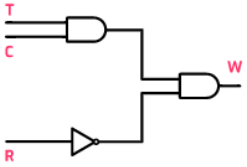


Figure 3: Option C circuit diagram

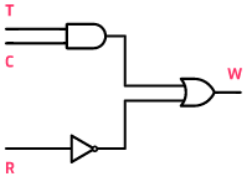


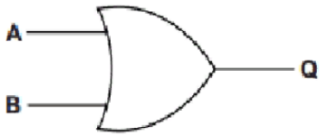
Figure 4: Option D circuit diagram

This question is borrowed from Isaac Computer Science, the link is here:

https://isaaccomputerscience.org/questions/gcse_bool_03?examBoard=all&stage=all

I chose this question as it tests the students' abilities to interpret a wordy question into a logic gate question.

(e) Complete the truth table for the following logic gate.



A	B	Q
0	0	0
0	1	1
ENTER	0	ENTER
1	ENTER	ENTER

Submit

$$((A \wedge B) \vee C) \wedge D$$

This question will be used in the 3D level of GCSE questions as they will have to build the circuit that requires 4 switches. This is more complex and tests the student's ability of understanding logic gate expressions with four different inputs.

The expression below will also be used for the same reason as the one previously.

$$(A \wedge B) \vee (C \vee D)$$

A-Level

10 (a) Complete the truth table for the NOR gate.

A	B	Output (XO
0	0	
0	1	
1	0	
1	1	

[1]

This question tests the students' abilities on understanding of new logic gates and their outputs for certain inputs

$(\neg A) \vee (\neg B) \vee (A \wedge B \wedge \neg C)$

A \ BC	00	01	11	10
0				
1				

This question will be given without the answer and will test the students understanding of Karnaugh maps which is a key topic in A-Level Computer Science

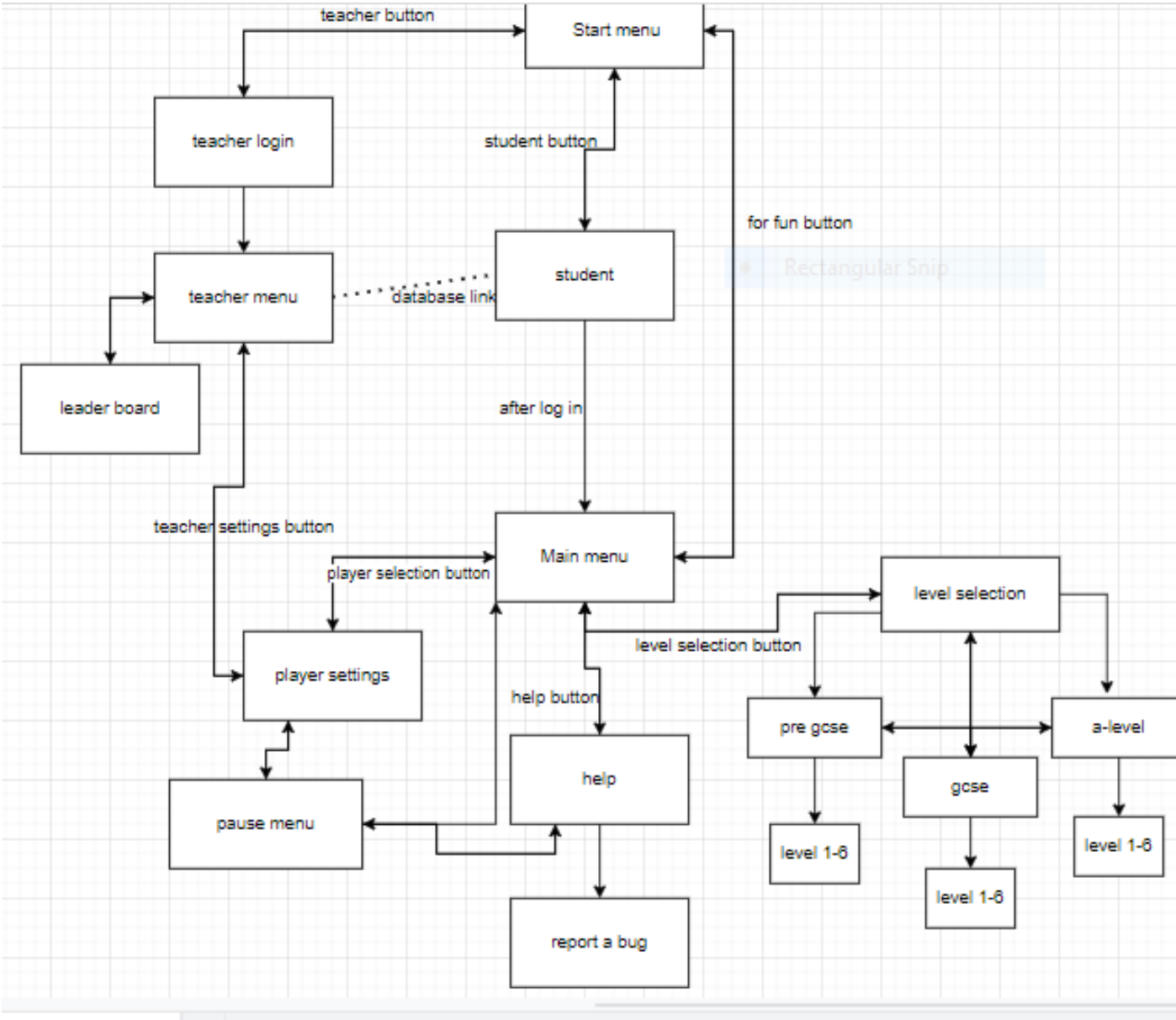
For the 3D questions I decided to do questions similar to GCSE with four inputs, however I decided to add an XOR gate into the answer to test some people if they are familiar with XOR gates. This is shown below

$$A \vee ((B \wedge C) \veebar D)$$

Top-Down Diagrams

I chose to do top-down diagrams, because it simply shows each process needed to be made within the game. It is easy for me as the developer to read, and it is also easy for the client to understand in case they want to make any changes.

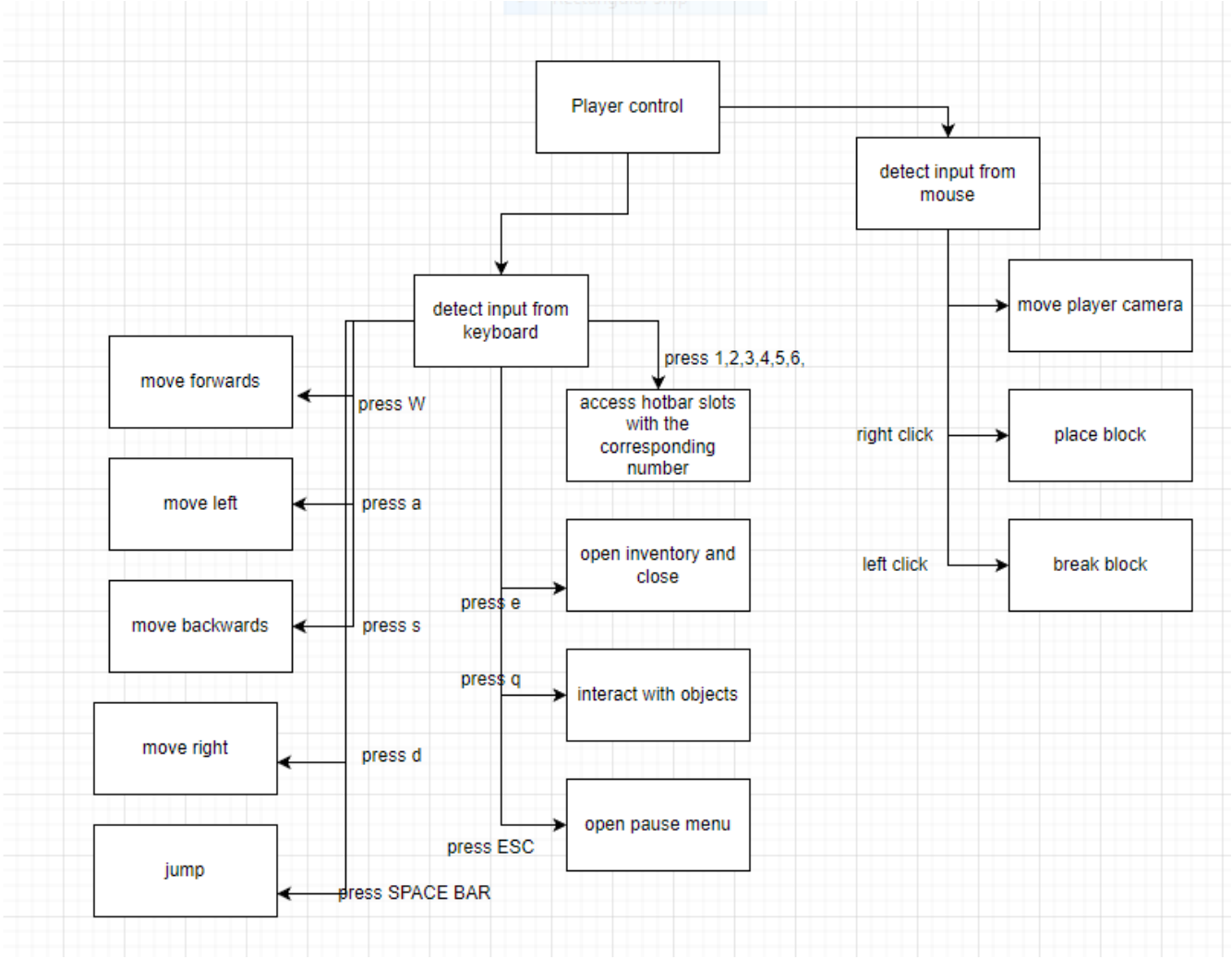
UI navigation



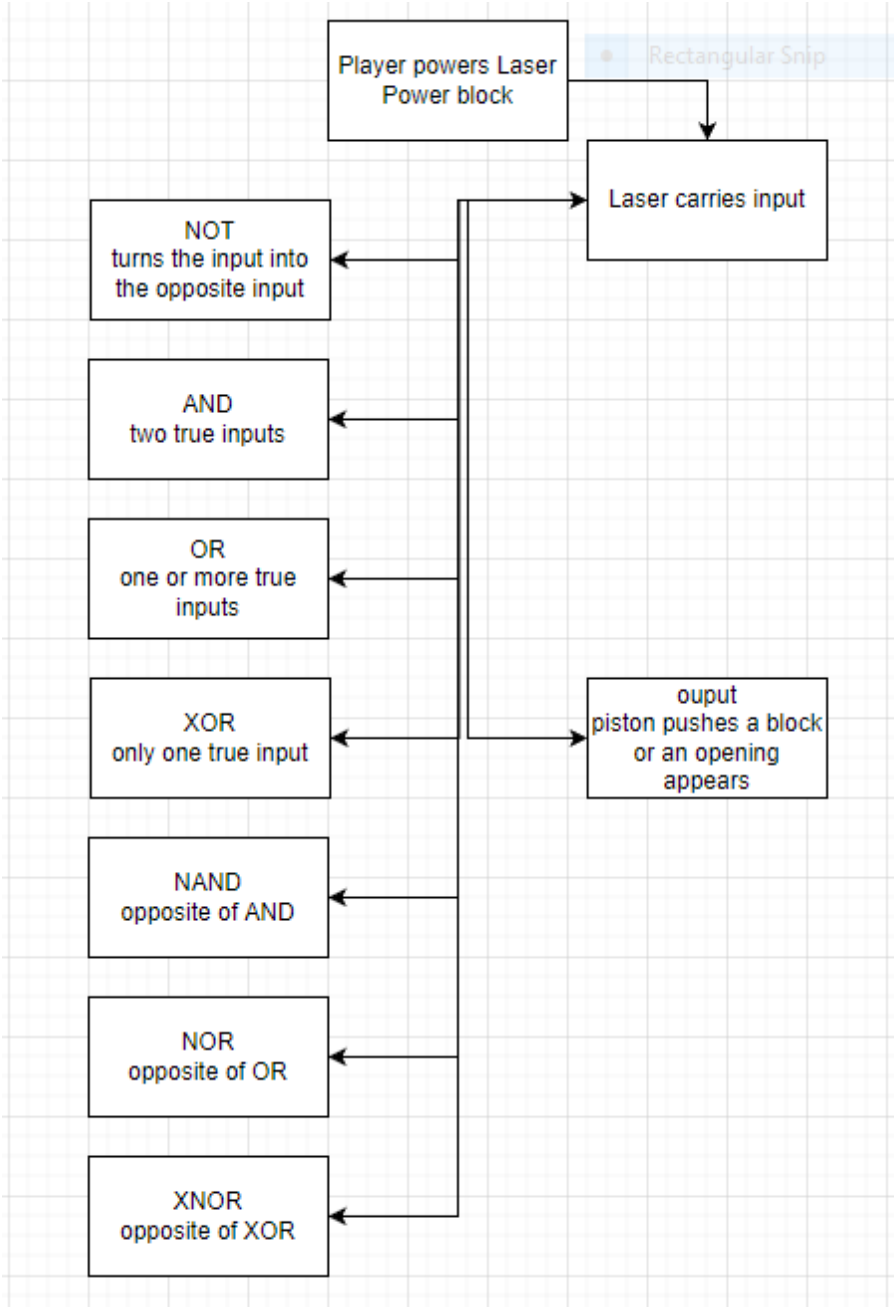
The image above is a flow diagram that connects all the UI screens together showing which user interfaces can go back to their previous screen and which screens are connected. It also shows the database link between the teacher's leader board and the students which will have their best score in the leader board so that the teacher is able to see their progress.

In game processes

Player Inputs



Other processes



Database design

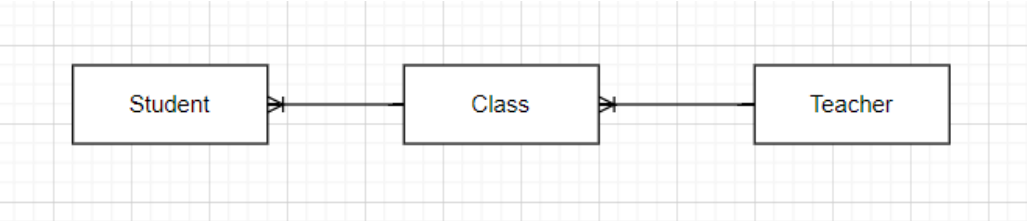
In my database design, I chose to create Entity Relationship Diagrams to design and plan out my tables. I first started working out trying to figure out the relationship between a student and a teacher.

A teacher has many students, and a student can have many teachers. Creating a many to many relationships.

I hence must create a third table to reduce uncertainty and duplications on data that will eventually result in wrong statements for queries

A teacher has many classes, and that instance of a class has one teacher

A class has many students, but students only have one class with that teacher.



I successfully created the relationship between the teacher and student.

For the teacher database

Teacher
TeacherID (primary key)
Teacher Username
Password

This table is used to store the teacher's details and ID. This will enable the teacher to login and sign up to use the application.

Student
StudentID (primary key)
ClassID (foreign key)
Student Username
Score

The student table takes in ClassID as a foreign key which means it is linked to the Class database hence creating a relational database instead of a flat file one. This removes repeated variables and removes the many to many relationships made by the teacher and the student. This will also store the username and score

Class
ClassID (primary key)
TeacherID (foreign key)
Class Name

ClassID is a primary key used as a unique identifier for each class, this will be generated so that the students can join the class

TeacherID is used as a foreign key so that the class which teacher is associated with the class

Class name is used to help distinguish classes for teachers and is mainly for user readability

I next must create a table for levels so that the student can store their score for a certain level

A student can play many levels, so I ended up with this design

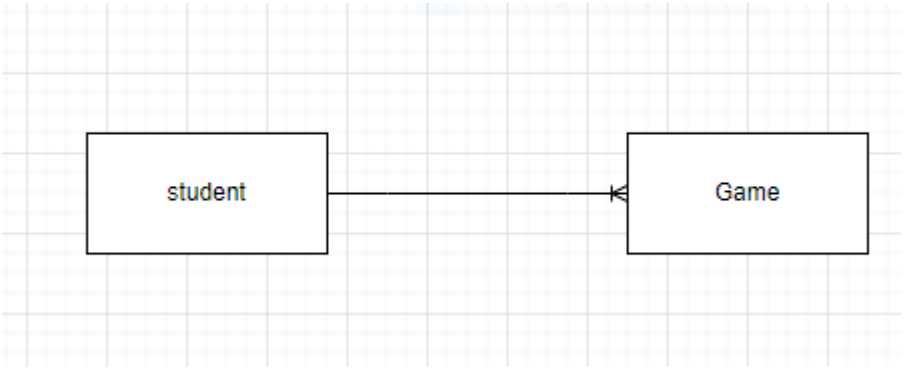
This diagram is supposed to represent the relationship between a student and a level. A student has many levels that they can play.

Level
LevelID (primary key)
Score

LevelID is used to differentiate all 18 levels within the game and the score that a player has however the score does not know which player it belongs to hence I tried to change this database to suit this need.

I had originally thought that one student will have a one-to-many relationship with a level as a student has many levels to complete but didn't consider that levels will have many instances of a student playing that level, so it is instead a many-to-many relationship.

I next considered to create a second iteration of the database using a game table to store the players instance of that game for that level.



The relationship between this would be one to many between student and game and also game and level.

Teacher
TeacherID (primary key)
Teacher Username
Hashed Password

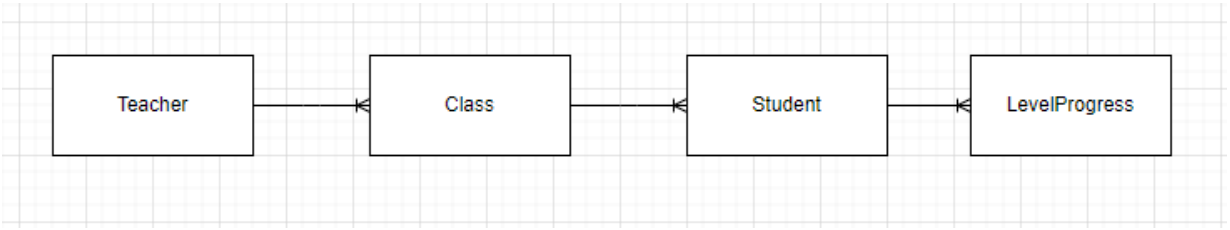
Student
StudentID (primary key)
ClassID (foreign key)
Student Username
Score
Sensitivity
Graphics quality

Class
ClassID (primary key)
TeacherID (foreign key)
Class Name

Level
GameID composite key)
Level (composite key)

Game
GameID(primary key)
StudentID(foreign key)

Hence, I made another table called Game, this allows me to link student and level together keeping track of their score for that level. However, this table still did not work as it stores score in the student table. This will not work as the student will have a different score for each level meaning that I can't store only one value hence I removed the level and game table to create the LevelProgress table.



This table links to student ID and the level number via a composite key creating a unique identifier. This then stores the score and whether the level has been completed.

I also added sensitivity and graphics quality as variables to store so that the student can customise their experience to their liking further improving usability.

Teacher
TeacherID (primary key)

Teacher Username
Hashed Password

Student
StudentID (primary key)
ClassID (foreign key)
Username
Password
Sensitivity
Graphics quality

Class
ClassID (primary key)
TeacherID (foreign key)
className

LevelProgress
StudentID (composite key)
Level number (composite key)
Score
Completed

This is my finalised database.

Algorithms

Opening and closing inventory

```
public inventoryObj
bool displayed = false

while True:S
    if input.key == "Q":
        inventoryObj.visible = !displayed
        displayed = (!displayed)
```

This algorithm will constantly check for a user input and if the key that was pressed was Q this then turns the object to be the opposite of the current state of the object's visibility

Data dictionary: OpenInv			
Variable name	**Variable Type**	**Purpose**	**Validation**
inventoryObj	Object	Inventory display object allows the	n/a

		player to change their item positions. Visibility set to false	
Displayed	Boolean	If inventoryObj is displayed or not	Default value set to False by the game, can be overridden if needed.

Inventory Management

```
procedure whenClicked(this):
    if dragging == False:
        objectClicked = this
        objectClicked.position, tempObj.position = tempObj.position, objectClicked.position
        dragging = True
        while dragging == True:
            objectClicked.position = input.MousePosition
    else
        objectClicked.position, this.position = this.position, objectClicked.position
        inventory[objectClicked.index], inventory[this.index] = inventory[this.index], inventory[objectClicked.index]
        hotbar = inventory

while True:
    if input.MouseScrollWheelUp:
        index += 1
    elif input.MouseScrollWheelDown:
        index -= 1
    hotbar[index].position = selectItemObj.position
```

This code checks if the player has pressed the button that is attached to the sprites, hence allowing the sprite to follow the mouse. If another sprite button is clicked, their places are swapped. There is a temporary object that acts as an empty sprite when the player first clicks on a sprite or clicks on a sprite when the inventory is full to swap the positions so that it appears that the sprite has left the slot.

The code also takes in the scroll wheel up and down features to access the inventory slots, this is displayed using an object that is overlayed on the current selected item in the hotbar

Data dictionary: Inventory

Variable name	Variable Type	Purpose	Validation
Inventory	Array of images	Stores the items that the player can place	Default value set to [] by the game, can be overridden if needed.
Hotbar	Array of images	Stores the items that the player can place. Is the same as the inventory	Default value set to [] by the game, can be overridden if needed.
dragging	Boolean	If the user is dragging an item or not	Default value set to False by the game, can be overridden if needed.

objectClicked	Object	The object that the player has clicked	n/a
tempObj	Object	The previous item that has been clicked	n/a
selectItemObj	Object	The object that is overlayed ontop of an inventory slot to visualise the selected item	n/a
index	Int	Index position of the currently selected item	Default value set to 0 by the game, can be overridden if needed.

Movement

```
playerObj
```

```
while True:
    if input.key == "W":
        playerObj.Forward
    if input.key == "A":
        playerObj.Left
    if input.key == "S":
        playerObj.Backward
    if input.key == "D":
        playerObj.Right
    if input.key == "Space":
        playerObj.Up
```

This is the movement script that checks for inputs for WASD and space bar and will apply forces in whatever direction is needed to be applied

Data Dictionary: Movement			
Variable name	**Variable Type**	**Purpose**	**Validation**
playerObj	Object	The object that the player controls allowing it to move	n/a

LookAround

```
while True:
    x = input.Mouse("X") * sens
    y = input.Mouse("Y") * sens
    playerCamera.rotation.y -= y
    playerCamera.rotation.x -=x
```

This script will take in the sensitivity which will be updated via a slider in the settings menu and also will be assigned to your saved sensitivity when you sign in.

It will change your camera rotation based on the mouse input and sensitivity the camera is a child of the playerObject which means whenever you move the camera will follow.

Data Dictionary: LookAround			
Variable name	**Variable Type**	**Purpose**	**Validation**
playerCamera	Camera	A camera that is a child of the playerObj so that it follows the playerObj and acts as a first-person view. This can be controlled by the mouse and sensitivity	n/a
sens	float	The sensitivity of the mouse, the player can adjust this in settings	Default value set to 2.5 by the game, can be overridden if needed.

Switch Power

```
while True:
    if playerObj.sphere.collide(switchObj):
        popUpObj.visible = True
        if input.key == "E":
            onOff = !onOff
    else:
        popUpObj.visible = False
```

This script constantly creates a sphere on the player that checks for any collisions with that sphere

If it has collied the popUp visibility will change to the opposite of its current state and when E is pressed the power will turn on for the switchObject

Data Dictionary: Power			
Variable name	**Variable Type**	**Purpose**	**Validation**
playerObj	Object	The player object that the user can control. It also checks if the player object is colliding with the switch Object.	n/a
popUpObj	Object	If the player is colliding with a switch object, then a pop up will appear with text	n/a

		saying to press E to turn on the power	
switchObj	Object	The switch object used to power a circuit	n/a
onOff	boolean	The state of the current / power	Default value set to False by the game, can be overridden if needed.
circuit	List of objects	Stores the objects that the switch powers	Default value set to [] by the game, can be overridden if needed.

Gate

```
onOff
input
type


# The current is equal to the amount of input it receives and if it satisfies the boolean
if type == 0:
    # notGate
    onOff = ! (len(inputs) >= 1)

if type == 1:
    # andGate
    onOff = len(inputs) >= 2

if type == 2:
    # orGate
    onOff = len(inputs) >= 1

if type == 3:
    # xorGate
    onOff = len(inputs) == 1

if type == 4:
    # nandGate
    onOff = ! (len(inputs) >= 2)

if type == 5:
    # norGate
    onOff = ! (len(inputs) >= 1)

if type == 6:
    # xnorGate
    onOff = ! (len(inputs) == 1)
```

Variable	Data Type	Description	Validation
onOff	Boolean	Represents whether the gate is currently on or off.	Default value set to False by the game, can

			be overridden if needed.
type	Integer	Represents the type of gate.	Required value set by the game for each object, and not intended to be changed at runtime. The value is predetermined by the type of logic gate that it is (between 0 and 6)
inputs	List of objects	Contains a list of all the inputs that the gate is connected to.	Required set by game Default value set to [] by the game, can be overridden if needed.

Generate

```
laser
lasers
direction
hit
maxDistance
switchObject
laserBeam


for each laser in lasers:

    direction = laser.direction

    // Cast a ray in the direction of the laser
    if Physics.Raycast(laser.position, direction, out hit, maxDistance):

        if hit.gameObject.tag == "LogicGate":

            hit.gameObject.script.gate.inputs.Add(gameObject)
            #adds the current game object to the inputs list of the hit gameObject
            switchObject.script.power.circuit.append(hit.gameObject)
            #adds the hit gameObject to the circuit list

        Instantiate(laserBeam, laser.position, Quaternion.Rotation(direction))
```

Data Dictionary: Generate			
Variable Name	**Variable Type**	**Purpose**	**Validation**
lasers	List of objects	A list containing all laser objects in the scene.	Default value set to () by the game, can be overridden if needed.

laser	object	The current laser object in the loop iteration.	n/a
direction	vector3	The direction of the current laser object.	Default value set to (0,0,0) by the game, can be overridden if needed.
maxDistance	int	The maximum distance that the raycast will go if it doesn't hit any objects	Default value set to 0 by the game, can be overridden if needed.
hit	n/a	The point where the raycast reaches an object	Default value by the game, can be overridden if needed.
switchObject	object	The lasers origin point / where it got generated from	n.a
laserBeam	object	The laser prefab to instantiate	n/a

Score

```
score = 10000

while True:
    if script.winCheck.win == False:
        score = score - 10
        time.seconds(1)
    else:
        win = true
```

If the winCheck script does not have win set to True, then this script constantly update score will decrease by 10 while win is not true

If win is true, it will stop

Data Dictionary: Score			
Variable name	**Variable Type**	**Purpose**	**Validation**
score	int	The score keeps decreasing per second until win is reached	Default value set to 10,000 by the game, can be overridden if needed.

win	boolean	Win boolean to check if the person has won	Default value set to False by the game, can be overridden if needed.

Win Check

```
while True:
    if levelType == 1: //3d level check order of gates
        for i = 0 to gates.length - 1:
            if correct < gates.length:
                if script.power.circuit[i] == gates[i]:
                    correct += 1
        if correct == gates.length:
            win = True:
    elif levelType == 2: //check order of things entered
        for i = 0 to inputs.length - 1:
            if correct < inputs.length:
                if inpOrder[i] == inputs[i]:
                    correct += 1
        if correct == inputs.length:
            win = True:
    elif levelType == 3: //match up
        procedure whenClicked()
            temp = this
            if temp != null:
                if temp.tag == this.tag:
                    correct += 1
        if correct == 3:
            win = True
    elif levelType == 4: // check if correct choice has been ticked
        procedure whenClicked()
            if clicked == False:
                temp = Instantiate(tickObj, this.position)
                clicked == True
            else:
                temp.Destroy()
                temp = temp = Instantiate(tickObj, this.position)

            if inpOrder == temp.name:
                win = True
```

LevelType 1 is checking if the gates part of the list (which keeps track of gates part of the circuit) are in the correct order by iterating through and comparing it with another hard coded list

LevelType 2 is filling in the gaps of a truth table or karnaugh map, it checks the win the win by checking if it's in the correct order by iterating through and comparing it with another hard coded list

LevelType 3 is the code for matching up gate names and gate symbols, this checks for the win by seeing if you click on two objects with the same tag 3 times, which means you have matched them up

LevelType 4 is the code for a multiple-choice question where you will tick the correct answer

This checks if you have won by checking if you have placed it in the correct place

Data Dictionary: WinCheck

Variable name	Variable Type	Purpose	Validation
gates	List of integers	List of the correct order of the gates by type	Default value set to () by the game, can be overridden if needed.
win	boolean	Win boolean to check if the person has won	Default value set to False by the game, can be overridden if needed.
levelType	int	The type of level that the current active scene is, if it is a level	Required value set by the game for each scene, and not intended to be changed at runtime. The value is predetermined based on the requirements of the specific scene
inputs	List of integers	List of inputs entered by the user	Default value set to () by the game, can be overridden if needed.
inpOrder	List of integers	The correct order of inputs	Default value set to () by the game, can be overridden if needed.
correct	Int	To check if you have the same order intended order by checking if the amount that is correct when comparing is the same as the length of the intended order	Default value set to 0 by the game, can be overridden if needed.

Outline Objects

```
outlineObj
selectInt = 0

while True:
    if physics.raycast(pos,dis,mask,hit):
        if selectPlaced != null:
            temp = selectPlaced

        if hit.tag == "Ground" and selectInt = 0:
            selectPlaced = Instantaite(outlineObj, Floor(hit.position),camera.rotation)
            selectInt = 1
            if temp != selectPlaced:
                hit = null
        if hit.position == 1 and selectInt = 0:
            selectPlaced = Instantaite(outlineObj, Floor(hit.position),camera.rotation)
            selectInt = 1
            if temp != selectPlaced:
                hit = null
        else:
            if selectInt == 1 and temp != hit.position:
                destroy(selectPlaced)
                selectInt = 0
```

Using physics.raycast it checks if the raycast hits a solid object allowing me to instantiate an outline around the object

This code will instantiate the outline object in two circumstances:

1. If you are looking at the ground and there is not already an outline object instantiated, I use the variable SelectInt to keep track if it has been instantiated
2. If the solid object is on a y level of 1 which means it has been placed or is preplaced to be a part of the level

However, it will also check if the selectInt is equal to one meaning it has been instantiated and the new position you are looking at is different; if so then it will delete the outline object, this is so that whenever you look away the outline object deletes reducing performance due to less objects in the scene

Data Dictionary: OutlineObject

Variable name	Variable Type	Purpose	Validation
outlineObj	object	The outline object prefab to be instantiated	n/a
selectInt	int	The status of the outlineObj/ if there is an instantiated outline object	Default value set to 0 by the game, can be overridden if needed.

pos	Vector3	Position from where the line from the raycast will spawn	Default value set to (0,0,0) by the game, can be overridden if needed.
dis	int	The maximum distance that the raycast will go if it doesn't hit any objects	Default value set to 5 by the game, can be overridden if needed.
mask	mask	The objects that have this mask applied, will either be ignored or accepted as being hit	Default value by the game, can be overridden if needed.
hit	n/a	The point where the raycast reaches an object	Default value by the game, can be overridden if needed.
selectPlaced	Object	The current outlined object	n/a
camera	Camera	Player Camera. This determines where to instantiate the object since the player is using the camera to look around	n/a
temp	Object	The previous outline objects. Used to delete itself when a player looks somewhere else, so it looks smooth.	n/a

Sandbox

```
placePos
placed


while True:
   if Physics.Raycast(pos,dis,mask hit):
       if input.button = "LeftMouseButton"
           placePos = Vector3 (Round(hit.point.x),Round(hit.point.y),Round(hit.point.z))
           if placePos.y == 1:
               placed = Instantiate(script.inventory.hotbar[script.inventory.index].GameObject,placePos,camera.rotation
       elif input.button = "RightMouseButton":
           hit.GameObject.Destroy()
```

This script uses Raycast to see if the player is looking at a solid object

Depending on the mouse button clicked it will do one of two actions

Left = This will round all the values of where you are looking at so that when you place the block it will be placed in whole number integers, effectively creating a grid system due to all blocks being 1x1x1, the code checks if the place pos y axis is equal to one and the instantiates the selected block from the inventory script in the Place pos position variable and using the same rotation as the camera so that it can be easy to rotate the gates to your liking

Right = Destroys the object that you are looking at

Data Dictionary: Sandbox

Variable name	Variable Type	Purpose	Validation
placePos	Vector3	Position of where you will place your block	Default value set to (0,0,0) by the game, can be overridden if needed.
placed	object	The instantiated object	n/a
pos	Vector3	Position from where the line from the raycast will spawn	Default value by the game, can be overridden if needed.
dis	int	The maximum distance that the raycast will go if it doesn't hit any objects	Default value set to 5 by the game, can be overridden if needed.
mask	mask	The objects that have this mask applied, will	Default value by the game, can be overridden if needed.

		either be ignored or accepted as being hit	
hit	n/a	The point where the line reaches an object	Default value by the game, can be overridden if needed.
camera	Camera	Player Camera. This determines where to instantiate the object since the player is using the camera to look around	n/a

Sign Up

```
procedure signUp():
    if password == confirmpassword:
        file = sqlite3.connect("dbName")
        if student == True:
            file.execute("CREATE TABLE IF NOT EXISTS Student(StudentID INTEGER, ClassID VARCHAR(4), Username VARCHAR(20), Password VARCHAR(20),Sensitivity FLOAT, Graphics INTEGER, PRIMARY KEY (StudentID AI))")
            file.execute("CREATE TABLE IF NOT EXISTS LevelProgress(StudentID INTEGER, LevelNum INTEGER, Score VARCHAR(20), PRIMARY KEY (StudentID, LevelNum))")
            file.execute("INSERT INTO Student(ClassID,Username, Password, Sensitivity, Graphics) VALUES ({},{},{},{},{});".format(code,username,password,2.5,0))
        if teacher == True:
            file.execute = ("CREATE TABLE IF NOT EXISTS Teacher(TeacherID INTEGER, Username VARCHAR(20), Password VARCHAR(20),PRIMARY KEY (TeacherID AUTOINCREMENT))")
            file.execute = ("CREATE TABLE IF NOT EXISTS Class(ClassID VARCHAR(4), TeacherID INTEGER, className VARCHAR(20),PRIMARY KEY (ClassID))")
            file.execute = ("INSERT INTO Teacher(Username, Password) VALUES ({},{})".format(username,password))



procedure teacher():
    Screen.Load(TeacherSL)
    teacher = True


procedure student():
    Screen.Load(StudentSL)
    student = True
```

This code allows the user to select what type of user they are and will send them to the corresponding screen furthermore it will set a Boolean to allow the code to differentiate what user is using the program that will have input boxes to allow them to sign up. This will also create tables based on the type of user and if those tables do not exist yet. After it will initialise the tables with the variables in the Database Design above the algorithms section. The tables created for students is the Student table and the LevelProgress table. It will also insert the newly created student into the table with their variables also initialising the setting variables (sensitivity and graphics quality) to their default values. If a teacher is signing up, then it will create a Teacher table and a Class table if they do not exist. It will then insert the new teacher that is signing up into the teacher table.

Data Dictionary: SignUp			
Variable name	**Variable Type**	**Purpose**	**Validation**
username	string	The username that the user will enter into an input box	Default value set to “” by the game, can be overridden if needed.

password	string	The password that the user will enter into an input box and hashed when stored in the database	Default value set to “” by the game, can be overridden if needed.
confirmpassword	string	The password that the user will re-enter into an input box, to confirm their password in case they mistyped the first time	Default value set to “” by the game, can be overridden if needed.
code	int	The code that the student will enter into an input box, that is generated by the teacher and will act as a primary key for classes	Default value set to 0 by the game, can be overridden if needed.
student	boolean	A boolean to determine which type of user is logging in	Default value set to False by the game, can be overridden if needed.
teacher	boolean	A boolean to determine which type of user is logging in	Default value set to False by the game, can be overridden if needed.
file	Database	Allows to connect and execute SQL commands on the database that is being connected to	n/a

Login

```
username
hashedpassword
bool teacher
bool student
sensitivity
graphics
level
score
class
errorMessage
errorMessage.visible = False

procedure whenClickedTeacherButton():
        teacher = True

procedure whenClickedStudentButton():
        student = True

while True:
    if teacher == True:
        file = sqlite3.connect("Teacher")
        teachCS = file.cursor
        teachername = teacherbox.retrieve
        password = passwordbox.retrieve
        teachCS.execute("SELECT * FROM Teacher WHERE {} == Name and {} == Password".format(teachername,password))
        fetched = teachCS.fetchOne()
        if fetched == null:
            errorMessage.visible = True
        else:
            Screen.Load(TeacherMenu)
    elif student == True:
        file = sqlite3.connect("Student")
        stuCS = file.cursor
        username = userbox.retrieve
        password = passwordbox.retrieve
        stuCS.execute("SELECT * FROM Student WHERE {} == Name and {} == Password".format(username,password))
        fetched = stuCS.fetchOne()
        if fetched == null:
            errorMessage.visible = True
        else:
            Screen.Load(StudentMenu)
```

This is the code for the login and start page.

If teacher is clicked till set teacher to true

In the login section it will check if the username and password that was typed into the input boxes are correct and then load the teacher menu

If the student is clicked it will set the student to true

It will then check the username and password that was used to login and load student menu after

If nothing is retrieved, then it shows an error message for both of them.

Data Dictionary: login			
Variable name	**Variable Type**	**Purpose**	**Validation**
username	string	The username that the user will enter into an input box	Default value set to “” by the game, can be overridden if needed.

hashedpassword	string	The password that the user will enter into an input box and hashed when stored in the database	Default value set to “” by the game, can be overridden if needed.
errorMessage	Object	An error message that will show if you have got the wrong password. Visibility set to false	n/a
teacher	boolean	A boolean to determine which type of user is logging in	Default value set to False by the game, can be overridden if needed.
student	boolean	A boolean to determine which type of user is logging in	Default value set to False by the game, can be overridden if needed.
graphics	int	The index position of which quality the students have chosen to play their game at	Default value set to 0 by the game, can be overridden if needed.
sensitivity	float	The sensitivity of the player used to adjust the mouse sensitivity to movement of the mouse	Default value set to 2.5 by the game, can be overridden if needed.
score	int	The score that the player has achieved for that level	Default value set to 0 by the game, can be overridden if needed.
file	Database	Allows to connect and execute SQL commands on the database that is being connected to	n/a

Updating and initialising values

```
sensSlider
graphicsDropDown

procedure whenClicked():
    file = sqlite3.connect("Student")
    stuCS = file.cursor
    stuCS.execute("SELECT sensitivity, graphics FROM Student WHERE {} and {}".format(script.login.username,script.login.password")
    fetched = stuCS.fetchOne()
    if fetched != null:
        script.LookAround.sens = fetched[4]
        sensSlider.value = fetched[4]
        graphicsDropDown.value = fetched[5]

procedure Adjusted():
    file = sqlite3.connect("Student")
    stuCS = file.cursor
    stuCS.execute("UPDATE Student SET sensitivity = {} WHERE {} and {}".format(sensSlider.value,script.login.username,script.login.password")
    stuCS.execute("UPDATE Student SET graphics = {} WHERE {} and {}".format(graphicsDropDown.value,script.login.username,script.login.password")
```

When the settings menu is accessed, it will initialise the graphics and sensitivity by retrieving them from the database and setting the UI to the correct values.

The code also checks if you make any changes to the settings when you do, it will update the values and change the graphics and sensitivity for when you change it

Data Dictionary: IntialisingValues			
Variable name	**Variable Type**	**Purpose**	**Validation**
sensSlider	Slider	The user will be able to interact with the slider to change their sensitivity which affects their mouse. Default slider value is set to 2.5	n/a
graphicsDropDown	Dropdown	The user will be able to interact with a drop-down menu to select the graphics option for 3D levels	n/a

Leaderboard

```
private string dbName = "URI=file:Databases.db"
list usernames
list scores

procedure whenClicked():
    file = sqlite3.connect(dbName)
    file.execute
    ("SELECT usernames, score
    FROM Class
    JOIN Student
    ON Class.ClassID = Student.ClassID
    WHERE Class.TeacherID = (SELECT TeacherID
                             FROM Teacher
                             WHERE name = {} and password = {}".format(script.login.teachername,script.login.password))

    fetched = file.fetchOne()
    for i = 0 in range(0,fetched.length - 1, 2)
        usernames[i].text = fetched[i]
        password[i].text = fetched[i+1]
```

When the leader board page is accessed, it connects to a file that has all the databases within them it then retrieves all the students' usernames and their score from the class of the teacher. It fetches

this and then goes through the text objects stored in usernames and password changing it to the fetched data from the database.

Data Dictionary: Leaderboard			
Variable name	**Variable Type**	**Purpose**	**Validation**
usernames	List of string	The list of all username's textboxes	Default value set to () by the game, can be overridden if needed.
scores	List of string	The list of all score textboxes	Default value set to () by the game, can be overridden if needed.
file	Database	Allows to connect and execute SQL commands on the database that is being connected to	n/a
fetched	Array of strings	An array that stores the fetched values from the database in an array	Default value set to [] by the game, can be overridden if needed.

Add and Remove Classes

```
dropDownBox
genNum = 0:
classNameInputBox

procedure whenDeleteClicked():

    file = sqlite3.connect("Class")
    file.execute("DELETE FROM Class WHERE className={};".format(dropDownBox.text))



procedure whenAddClicked():
    if genNum != 0:
        file1 = sqlite3.connect("Teacher")
        file1.execute("WHERE TeacherID = (
        SELECT TeacherID
        FROM Teacher
        WHERE name = {} and password = {}".format(script.login.teachername,script.login.password)
        teacherID = file1.fetchOne()
        file2 = sqlite3.connect("Class")
        file.execute("
        INSERT INTO Class (ClassID, TeacherID, className)
        VALUES ('{}', '{}', '{}'.format(genNum,teacherID[0],classNameInputBox.text))
        ")


procedure whenGenerateClicked():

    file = sqlite3.connect("Class")
    file.execute("
    SELECT ClassID
    FROM Class")
    fetched = file.fetchOne()
    flag = True
    while flag == True:
        genNum = generate()
        for i = 0 to fetched.length -1:
            if i == fetched.length -1:
                flag = False


procedure generate()
    random_number = random.randint(1, 9999)
    formatted_number = "{:04d}".format(random_number)
    return formatted_number
```

This script removes databases by accessing a dropdown box of all the classes the teacher currently has, once selected it will delete the class.

Adding classes is a bit more complex, the program has a procedure ran whenever the teacher tries to generate a new code for the class. This code acts as a primary key, so first the program checks if it is unique if not it will regenerate a new number and then starts the process of checking again. This code will be a unique 4-digit number if it is less than a 1000 it will lead with 0's. For example, 0001 or 0173.

Once generated the teacher can then fill in details for the class such as the name, then the code will connect to the teacher database to be able to access the TeacherID of the currently logged in teacher. This inserts a new data into the Class database with the generated number as the primary key and the teacherID and class name retrieved from an input box.

Data Dictionary: Class

Variable name	Variable Type	Purpose	Validation
file	Database	Allows to connect and execute SQL commands on the database that is being connected to	n/a
file1	Database	Allows to connect and execute SQL commands on the database that is being connected to	n/a
dropDownBox	Dropdown	The user will be able to interact with a drop-down menu to select the classes that the user wants to remove	n/a
genNum	string	The random number that is formatted to “0000”. It is also the primary key for the class table	Default value set to “” by the game, can be overridden if needed.
classNameInputBox	Input box	Input box from where the input will be retrieved	n/a
fetched	Array of strings	An array that stores the fetched values from the database in an array	Default value set to [] by the game, can be overridden if needed.
i	int	The index positon used to iterate through the fetched array	Default value set to 0 by the game, can be overridden if needed.
flag	Boolean	A flag used to determine if there is a repeated code in the database	Default value set to False by the game, can be overridden if needed.
random_number	int	A random number that is generated between 0 to 10,000	Default value set to 0 by the game, can be overridden if needed.

Screen Director

```
procedure MainMenu()
    Screen.Load(MainMenu)

procedure Settings()
    Screen.Load(Settings)

procedure Help()
    Screen.Load(Help)

procedure LoginS()
    Screen.Load(LoginS)

procedure LoginT()
    Screen.Load(LoginT)

procedure Load(ButtonName)
    Screen.Load(ButtonName)

while True:
    if input.key == "ESC":
        Screen.Load(PauseMenu)
```

This code is the procedures that will happen for each button whenever they are clicked. If the escape key is pressed, then the pause menu will show

I also made it easier to load levels or other screens by changing the name of the button using the load procedure.

Data Dictionary: Screen Director			
Variable name	**Variable Type**	**Purpose**	**Validation**
ButtonName	string	Loads the screen with the same name as the button.	Default value set to “” by the game, can be overridden if needed.

Updating the Score

```
while True:
    if script.Score.win == True:
        updateScore()
        Screen.Load(GameComplete)



procedure updateScore()
    file = sqlite3.connect("Student")
    file.execute("SELECT Score FROM Student WHERE StudentID = {}".format(readID))
    fetched = file.fetchOne()
    if fetched[0] == 0:
        file.execute("INSERT INTO LevelProgress(StudentID,LevelNum, Score) VALUES ({},{},{})".format(readID, level, script.Score.score))
    if fetched[0] < script.Score.score:
        file.execute("UPDATE LevelProgress SET Score = {}  WHERE StudentID = {} AND LevelNum ={}".format(script.Score.score,readID,level))
```

This script keeps checking if win is true from the Score script. If it is true, it will then go to the updateScore procedure. This will connect to the student database and select the score of the current student. Next it checks if it is empty if so, it will insert the new player and the new level into LevelProgress. If the score is higher, it will update the current level's score for that student.

After it will load the Level complete scene.

Data Dictionary: UpdateScore			
Variable name	**Variable Type**	**Purpose**	**Validation**
file	Database	Allows to connect and execute SQL commands on the database that is being connected to	n/a
readID	int	The integer that will be used as the StudentID field as a composite key along with level	Required value set by the game for each student, and not intended to be changed at runtime. The value is predetermined based on the value that it retrieves from the database when the student signs in or logs in
level	int	The integer that will be used as the level num field as a composite key along with readID	Required value set by the game for each scene, and not intended to be changed at runtime. The value is predetermined based

			on the requirements of the specific scene

Testing Tables

Gameplay				
Function / method being tested	**Scenario**	**Input**	**Testing Type**	**Expected results**
Movement	In game	W (default)	Valid	Player will walk forwards, towards where the player camera is looking
		Other (e.g., H)	Invalid	The player will not move forward
		A (default)	Valid	Player walks to the left, camera moves along with the player
		Other (e.g., Y)	Invalid	The player will not move to the left
		S (default)	Valid	Player will walk backwards, 180° where the player camera is looking
		Other (e.g., G)	Invalid	The player will not move backwards
		D (default)	Valid	Player walks to the right, camera moves along with the player
		Other (e.g., T)	Invalid	The player will not move to the right
Player Look	In game	Move the mouse	Valid	Camera rotation correlates to

				mouse movement
		Other (e.g., Mouse button)	Invalid	Camera will not rotate
Player Jump	In game	Space (default)	Valid	Player jumps along the y axis with the camera following the player
		Other (e.g., J)	Invalid	Player does not jump
Player Interacts with Switch	In game	E (default)	Valid	Laser Power turns on or off depending on the state of the Laser Power
		Other (e.g., R)	Invalid	The laser power's state will stay the same and so will everything connected to it
Player access Inventory	In game and out of game	Q (default)	Valid	Inventory screen is brought up, black opacified canvas is also brought to cover the players camera view. Movement and freeze camera rotation. Gameplay continues as usual after Q is pressed again
		Other (e.g., F)	Invalid	Player's inventory does not pop up. Player can move and look around as usual.
Access inventory slots	In game	Scroll wheel	Valid	Access slots with their corresponding numbers

		Other (e.g., F)	Invalid	Player's inventory does not pop up. Player can move and look around as usual.
Player Place block	In game	Right click	Valid	The selected block in the inventory array is placed from a prefab of that block
Player break block	In game	Left click	Valid	If the block being looked at can be broken (checked via tags or mask) Then block is destroyed turns off the power
		Other	Invalid	If block is highlighted and breakable inputs other than left click will not do anything, power stays on
Pause menu	In game	ESC	Valid	Open Pause Screen, movement and being able to look around is disabled
Laser Power/ switch	In game	-	Valid	Can be placed (right click), broken (left click) and interacted with by pressing E, when allowed. this then turns the current on or off (thus influencing other blocks). Generates a laser in the z direction
Laser	In game	-	Valid	Is generated based on the rotation of the Laser Power +

				Carries the correct signal output between logic gates. laser power changes the laser output
AND GATE	In game	-	Valid	IF both inputs are true this gate will output positive and both inputs are from the x axis of the block
OR GATE	In game	-	Valid	IF one or both inputs are true this gate will output positive and both inputs are from the x axis of the block
NOT GATE	In game	-	Valid	INVERSES the output (input = positive output = false, input false output positive) And the only input is from the opposite side of the output
XOR GATE	In game	-	Valid	IF only one input is true this gate will output true and both inputs are from the x axis of the block
NAND GATE	In game	-	Valid	SAME as AND GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block
NOR GATE	In game	-	Valid	SAME as OR GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block

XNOR GATE	In game	-	Valid	SAME as XOR GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block
Mirror	In game	-	Valid	When a laser hits the mirror depending on its rotation it will reflect the mirror 90°
PlayerScore	In game	-	Valid	Player score is updated in top right of gameplay in the overlay. Player score is measured by the time taken to complete the level

Other in-game features		
UI screens	**Action**	**Expected result**
Start menu	Teacher button	Redirects to the Teacher login screen
	Student button	Redirects to the Student login screen
	X Button	Exists to desktop
Teacher Sign Up and Log in	Username Input	Allows the user to enter an email
	Password Input	Allows the user to enter a password
	Confirm Password Input	Allows the user to enter a password and checks if it is the same as the original one.
	Enter button	These inputs are submitted and checked in the database if successful redirects to Teacher Menu Screen, if not error is displayed
	Back button	Returns to previous screen
	X button	Exists to desktop
Student Sign Up and Log in	Username Input	Allows the user to enter an email

	Password Input	Allows the user to enter a password
	Confirm Password Input	Allows the user to enter a password and checks if it is the same as the original one.
	Code Input	User can enter a code that will join the student to the class screen
	Enter button	These inputs are submitted and checked in the database if successful redirects to Teacher Menu Screen, if not error is displayed
	Back button	Returns to previous screen
	X button	Exists to desktop
Teacher Menu	Add Button	Redirects to Add class screen and creates a new class database
	Remove Button	Redirects to Remove class screen
	Leaderboard Button	Redirects to Leaderboard class screen
	X button	Exit to desktop
Teacher Add	Generate Button	Generates a random number in a 4-digit format that leads with 0s if its less than 1000.Also checks to see if its unique and regenerates
	Code text	Generated number shows up here
	Class name input	Teacher can enter the class name
	Submit button	Submits all data to the database and inserts the class into the database
	X button	Exit to desktop
Teacher Remove	Class Drop Down Menu pt1	Menu drops down with correct classes groups
	Class Drop Down Menu pt2	Select the class and it saves correctly this is then submitted and connects to the database
	Submit button	Submits all data to the database and deletes the selected class from the database
	X button	Exit to desktop

Leaderboard	Class Drop Down Menu pt1	Menu drops down with correct classes groups
	Class Drop Down Menu pt2	Select the class and it saves correctly this is then submitted and connects to the database
	Level Drop Down Menu pt1	Menu drops down with correct Levels from pre gcse – gcse and finally alevel
	Level Drop Down Menu pt2	Select the level and it saves correctly this is then submitted and connects to the database
	-	Shows the correct class, names and scores in the correct order and places
	Scroll Wheel	Moves the page down
	Scroll Slider	Updates depending on how far down the page you are and also moves the page down when held and dragged down
	Back button	Returns to previous screen
	X button	Exit to desktop
Main Menu	Level Selection Button	Redirects you to Level Selection Screen
	Settings Button	Redirects you to Settings Screen
	Help Button	Redirects you to Help screen
	X button	Exit to desktop
Settings	Sensitivity slider	Student can move the slider around and this will change the sensitivity in the 3D levels in game
	Graphics Drop Down Menu pt1	Menu drops down with correct classes groups
	Graphics Drop Down Menu pt2	Select the class and it saves correctly this is then submitted and connects to the database
	Back button	Returns to either Pause Menu or Main Menu depending on the previous scene
	X button	Exit to desktop
Pause Menu	Main Menu button	Redirects you to Main Menu Screen
	Settings button	Redirects you to Settings Screen
	Resume button	Returns to previous screen
	X button	Exit to desktop

Help	Content Image	Contains an image of how to play and the keybinds
	Back button	Returns to previous screen
	Scroll Wheel	Moves the page down
	Scroll Slider	Updates depending on how far down the page you are and moves the page down when held and dragged down
	X button	Exit to desktop

Database Inputs and Outputs					
UI screens	**Action**	**Input**	**Input Data**	**Testing Type**	**Expected result**
Creating Tables	Pressing Student button in the Start Menu	User clicks the student button	n/a	Valid	Redirects to Student sign up or login in screen and creates the student table if it does not exist
	Pressing Teacher button in the Start Menu	User clicks the Teacher button	n/a	Valid	Redirects to Teacher sign up or login in screen and creates the student table if it does not exist
	Pressing Level Selection button in the Main Menu	User clicks the Level Selection button	n/a	Valid	Redirects to PRE-GCSE Level selection and creates the LevelProgress table if it does not exist
	Pressing Add Class button in the Teacher Menu	User clicks the Add Class button	n/a	Valid	Redirects to Teacher sign up or login in screen and creates the Class table if it does not exist
	A student trying to log in without signing up beforehand /	User clicks the submit button on the log in screen	n/a	Erroneous	No actions should be performed to access the

	Student table does not exist yet				database since it does not exist, and text should display telling the user to sign up first
	A teacher trying to log in without signing up beforehand / teacher table does not exist yet	User clicks the submit button on the log in screen	n/a	Erroneous	No actions should be performed to access the database since it does not exist, and text should display telling the user to sign up first
Sign Up	Pressing submit with something not filled in	User clicks the submit button on the sign-up screen with nothing inputted in any of the fields	Username = “” Password = “” Confirm Password = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.
	Password is 8 characters long or less	User clicks the submit button on the sign-up screen with the password being less than 8 characters	Password = “passwor”	Erroneous	Text is shown to the user to enter a longer password, the code stops since the validation has failed.
	Password does not contain a special character	User clicks the submit button on the sign-up screen with the password not containing a special character	Password= “password”	Erroneous	Text is shown to the user to enter a longer password, the code stops since the validation has failed.
	Pressing submit with password and	User clicks the submit	Password = “password”	Erroneous	Text is shown to the user to

	confirm password not matching	button on the sign-up screen with the password and confirm password inputs not matching	Confirm Password = “possword”		say that the passwords do not match, the code stops since the validation has failed.
	Pressing submit without typing in the class code (student sign up only)	User clicks the submit button on the sign-up screen with nothing inputted in any of the fields	Code = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.
	Pressing submit when entering non numbers into the class code (student sign up only)	User clicks the submit button on the sign-up screen with letters inputted in the class code	Code = “091H”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.
	Pressing submit when entering code that is not 4 digits in length (student sign up only)	User clicks the submit button on the sign-up screen with the length of the inputted class code is not equal to 4 digits	Code = “01234”	Erroneous	Text is shown to the user to enter a longer code or the correct code, the code stops since the validation has failed.
	Entering every field, reaching every validation requirement(student) and pressing submit	User clicks submit when all validation is reached	Username = “naglis” Password = “password@” Confirm Password = “password@” Code = “0123”	Valid	Redirects to main menu while also inserting the data as a new entry into the student table
	Entering every field, reaching every validation requirement	User clicks submit when all validation is reached	Username = “flain” Password = “teachers@”	Valid	Redirects to teacher menu while also inserting

	(teacher) and pressing submit		Confirm Password = “teachers @”		the data as a new entry into the teacher table
Login In	Pressing submit with something not filled in	User clicks the submit button on the Log in screen with nothing inputted in any of the fields	Username = “” Password = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.
	Pressing submit and the Password and Username match not found within the database	User clicks the submit button on the Log in screen however their entry does not exist in the database since they have not signed up or their details are incorrect.	Username = “Josh” Password = “wrongpassword”	Erroneous	Text is shown to the user to enter the correct details, the code stops since the validation has failed.
	Entering every field, reaching every validation requirement (Student) and pressing submit	User clicks submit in the log in screen and passes all validation	Username = “naglis” Password = “password@”	Valid	Redirected to Main Menu
	Entering every field, reaching every validation requirement (teacher) and pressing submit	User clicks submit in the log in screen and passes all validation	Username = “flain” Password = “teachers@”	Valid	Redirected to Teacher Menu
Student Settings	Graphics drop down new option selected	User selects new option from the drop-down section	Graphics Drop Down Option = “Low”	Valid	Edits the graphics field data for the student that is changing the graphics so that it can be retrieved from the database when loading

					a 3d level to initialise the graphics settings
	Change in sensitivity slider	User slides the slider to pick a new sensitivity	Sensitivity slider value = 3.4	Valid	Edits the sensitivity field data for the student that is changing the sensitivity so that it can be retrieved from the database when loading a 3d level to initialise the player sensitivity
Student Score	Student completes a level for the first time	When win for that level is true.	n/a	Valid	It will check if there is already an entry since there is not, it will enter a new entry into the LevelProgress database table putting in the correct data.
	Student completes a level again but with a better score	When win for that level is true.	n/a	Valid	It will check if there is already an entry since there is, it will edit the entry from the LevelProgress database table changing the data to its new value only if the new value is

					larger than the old value in the database
	Student completes a level again but with a worse score	When win for that level is true.	n/a	Valid	It will check if there is already an entry since there is, however since the new value is lower than the value in the database nothing updates or changes.
Teacher Add Class	Pressed submit button with no inputs filled	Click submit on the add screen with no code generated or class name inputted	Code = “” Class Name = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.
	Pressed submit with only the class name filled	Click submit on the add screen with no code generated	Code = “” Class Name = “Year 13”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.
	Pressed submit with no class name and only a code	Click submit on the add screen with no class name inputted	Code = “5126” Class Name = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.

	Reaching every validation requirement and pressing submit	Click submit with all validation reached	Code = “5126” Class Name = “Year 13”	Valid	It will add the class into the database. It will also tell the user that their entry has been submitted
Teacher Remove Class	Initiating the drop-down Menu	When first loading in the screen, the drop-down menu must be initiated with the correct values /classes	n/a	Valid	Will retrieve all classes from the database for that teacher and to initialise the drop-down options
	Pressing Submit	Click submit on the teacher remove screen	Class Drop down option = “Year 13”	Valid	Will delete the class entry from the database and will also update the drop-down menu
Leader board	Initiating the drop-down Class Menu	When first loading in the screen, the drop-down menu must be initiated with the correct values/classes	n/a	Valid	Will retrieve all classes from the database for that teacher and to initialise the drop-down options
	Pressing Submit	Pressing submit on the leader board screen	Class Drop down option = “Year 13” Level Drop down option = “A-Level 1”	Valid	Will retrieve all scores of students from that class and all their scores from the level and format them into the leader board from highest score to lowest

Post Development Testing

For post development testing I will use beta testing to test the game. I will do this testing through two ways:

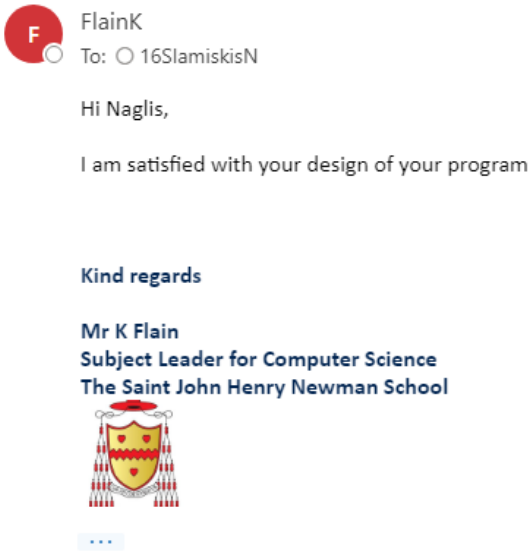
1: Ask Mr Flain to beta test the game and interview Mr Flain around thoughts on functionality and usability.

2: Take the game into a year 9 lesson and ask several students to beta test the game. You will hold a focus group with these students and gain comments on usability and functionality.

3: I will also manually test the inputs and outputs of each game trying to find bugs that I have not accounted for.

Therefore, I will have done beta testing with my client who will use the game in class and other clients who will play the game in class, I will also be part of the testing, testing the program myself furthermore when there is a bug, I will be able to try figure out why and maintain the program.

Client Sign Off



FlainK
To: 16SlamiskisN

Hi Naglis,

I am satisfied with your design of your program

Kind regards

Mr K Flain
Subject Leader for Computer Science
The Saint John Henry Newman School

Section 3: Development

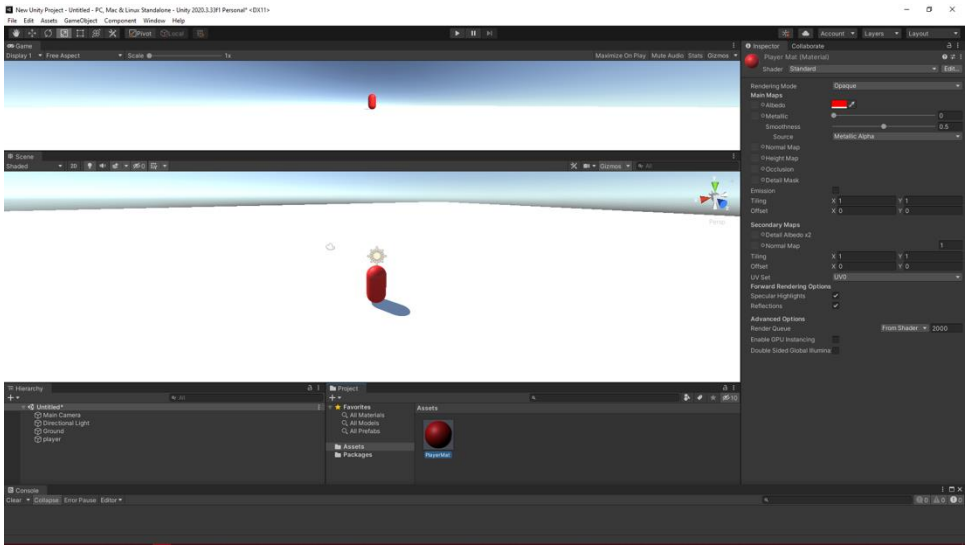
I chose object-oriented approach to develop my game and the solution to the problem of the new young people trying to get into Computer Science and creating a fun game to teach them about logic gates and develop their logical thinking skills. Objected -oriented programming will help code the game as it allows me to create logic gates to act as objects that the player can interact with, and these logic gates will contain methods and attributes. Unity is a game engine that is OOP by default and is made to cater towards OOP implementation and is also user-friendly which is why I chose to use it.

I began to develop the gameplay of my game and will then move on to developing the screens where I will also start to program the database section of my game. I developed the game section of the game first as I believe it is the most complex part of the game to get right and I needed to make sure I had enough time to allow to fix any game breaking bugs so that user can have the most satisfying and immersive experience and allow the students to get a good grasp on logic gates and how they work instead of being side-tracked by bugs and unfixed glitches.

Gameplay

Player Controls

I started off by creating a capsule object and calling it player. I chose red as the material for my player as it stands out from the floor that I also made. The floor was made by using the cube object and stretching it out to cover a good portion of the void.



I then started to code the players camera. This script will take the players mouse as an input and adjust the camera rotation accordingly

```
public Transform player; // player object assigned here as a transform so the rotatin is able
public float mouseSens = 2f; // mouse sensitivity, this allows the player to customise the spe
float cameraVertRot = 0f; // camera vertical rotation
```

The transform player gets the player object as a transform allowing it to rotate. Since the camera is a child of the player object it follows the player, and the parent object player will rotate and so will the child camera. The variable is public to allow the player object to be assigned

mouseSens variable is a variable to affect the speed of the players ability to look around and will be able to be set by the player in their settings. The variable is public so that it can be accessed and changed, this is an example of encapsulation where I am setting variables to public so that they can be accessed by other scripts. It is also a float as I decided it would give the player more options and hence the player can get closer to a sensitivity that suits them.

cameraVertRot is the camera's vertical rotation hence the name as vert and rot are shorthand for vertical and rotation. This variable is a float and because I want the camera to feel as smooth as possible and thus needed to get the most precise value to achieve this.

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    float inputX = Input.GetAxis("Mouse X") * mouseSens; // get the input of the mouses x axis and multiply this by the mouse sensitivty
    float inputY = Input.GetAxis("Mouse Y") * mouseSens; //  get the input of the mouses y axis and multiply this by the mouse sensitivty


    cameraVertRot -= inputY;// change the camera's rotation depending on the Y input from the mouse
    cameraVertRot = Mathf.Clamp(cameraVertRot, -90f, 90f); // limit the camera's y rotation between 90 and -90
    transform.localEulerAngles = Vector3.right * cameraVertRot; // apply the Y roation sn


    player.Rotate(Vector3.up * inputX); // change the player's rotation based on X input

}
```

inputX is a float variable that is defined in the update function which means it gets updated every frame and based on the input of mouse X it then multiplies this by the player's mouse sensitivity

inputY is a float variable that is defined in the update function which means it gets updated every frame and based on the input of mouse Y it then multiplies this by the player's mouse sensitivity

Both inputs are floats due to needing to get the accuracy of inputs allowing for smoother gameplay

The cameraVertRot is adjusted by the Y input and is then limited to –90 and 90 which then disallows the player to look up or down and 'overlook' which causes a very buggy perspective. Hence, I used clamp to limit the vertical rotation. This variable is then applied to the camera

The player's rotation is then changed based on the mouses X input.

```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    Cursor.visible = false; // make the cursor not visible
    Cursor.lockState = CursorLockMode.Locked; // lock the cursor in the centre of the screen
}
```

This function is called in the first frame and disables the cursors visibility and then centres the cursor to the middle of the screen.

This next script will allow the player to move and jump.

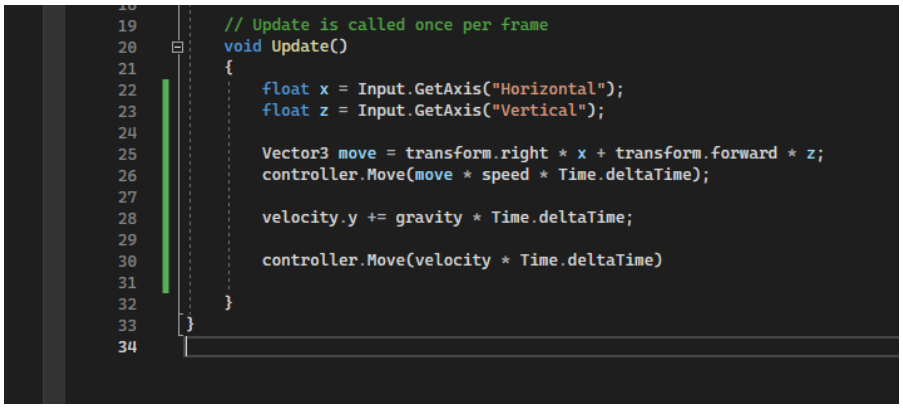
```
public class playerMovement : MonoBehaviour
{

    public CharacterController controller;
    public float speed = 12f;
    public float gravity = 9.81f;
    Vector3 velocity;
```

Character controller is an inbuilt unity script that allows the player to move and jump however you still need to code to let the player move and jump. I set it to public to allow the character controller to be accessed by the script.

The speed variable is public so that I can change it during testing. It is also a float variable to allow me to change it to a more suitable number which feels natural.

Gravity is the force acting down on the player. The variable is public so that I can change it during testing.

Velocity is a vector 3 as it is a vector quantity in real life and not a scalar. This direction for velocity allows me to jump and fall.



```
19      // Update is called once per frame
20      void Update()
21      {
22          float x = Input.GetAxis("Horizontal");
23          float z = Input.GetAxis("Vertical");
24
25          Vector3 move = transform.right * x + transform.forward * z;
26          controller.Move(move * speed * Time.deltaTime);
27
28          velocity.y += gravity * Time.deltaTime;
29
30          controller.Move(velocity * Time.deltaTime)
31
32      }
33  }
34
```

x is a float variable that is defined in the update function which means it gets updated every frame and based on the input of W and S

z is a float variable that is defined in the update function which means it gets updated every frame and based on the input of A and D

Both of these inputs are floats due to them being used to create the Vector 3 move

The Vector3 move variable is made of up of moving forward and right based on x and z

This is then sent in as a parameter along with speed and Time.deltaTime which is the time passed between the current and last frame update into the controller move functions which takes these in and moves the player.

The controller then moves the player based on gravity along the y axis and Time.deltaTime.

However, this code does not work as the jump still does not work properly for many reasons:

- The velocity will keep being added onto itself and does not reset
- The player cannot jump as there is nothing checking if the space bar is being pressed
- You do not check if the player is on the ground
- The player will fly up due to gravity not being negative

```
public CharacterController controller;
public float speed = 12f;// player speed and how fast they move
public float gravity = -9.81f; // gravity to affect player jump
public float jHeight = 3f; // how high the player can jump

public Transform groundCheck; // an empty object the checks if the player is on the ground
public float groundDistance = 0.1f; // how far the player has to be from the ground to be checked
public LayerMask Ground; // check if the object is the Ground

Vector3 velocity;
bool isGrounded; // boolean to check if the player is on the ground
```

The updated attributes are shown above.

The gravity variable has changed to the negative version

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{

    isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, Ground); // checks if its on the ground

    if (isGrounded && velocity.y < 0) // resets the velocity for jumping
    {
        velocity.y = -2f;
    }

    float x = Input.GetAxis("Horizontal"); // input from ws
    float z = Input.GetAxis("Vertical"); // input from ad

    Vector3 move = transform.right * x + transform.forward * z; // turn for the inputs into a vector 3
    controller.Move(move * speed * Time.deltaTime); // move along the x or z axis

    if (Input.GetButtonDown("Jump") && isGrounded) // if player presses space bar and is on the ground
    {
        velocity.y = Mathf.Sqrt(jHeight * -2f * gravity); // jump height times -2 times gravity = how high player jumps
    }

    velocity.y += gravity * Time.deltaTime; // changes the velocity when the player is falling

    controller.Move(velocity * Time.deltaTime); // applies the change in velocity



}
```

Added groundCheck, groundDistance and Ground mask for the check sphere. The isGrounded Boolean variable is changed depending on the check sphere. The new code restarts the velocity and also lets the player jump as long as they are on the ground.

My client is content with the functionality of the player controls.

Interaction

Next, I had to code the interaction between the switch / Laser Power and the player. I need the player to be able to turn the switches on and off. I also need a prompt to appear for the player to know how to interact with the switch.

```
public class Interactor : MonoBehaviour
{
    [SerializeField] private Transform interactionPoint;
    [SerializeField] private float interactionPointRadius = 0.5f;
    [SerializeField] private LayerMask interactables;

    private readonly Collider[] col = new Collider[3];
    [SerializeField] private int numFound;
```

The code above initialises the attributes and sets them to private since these variables do not need to be accessed.

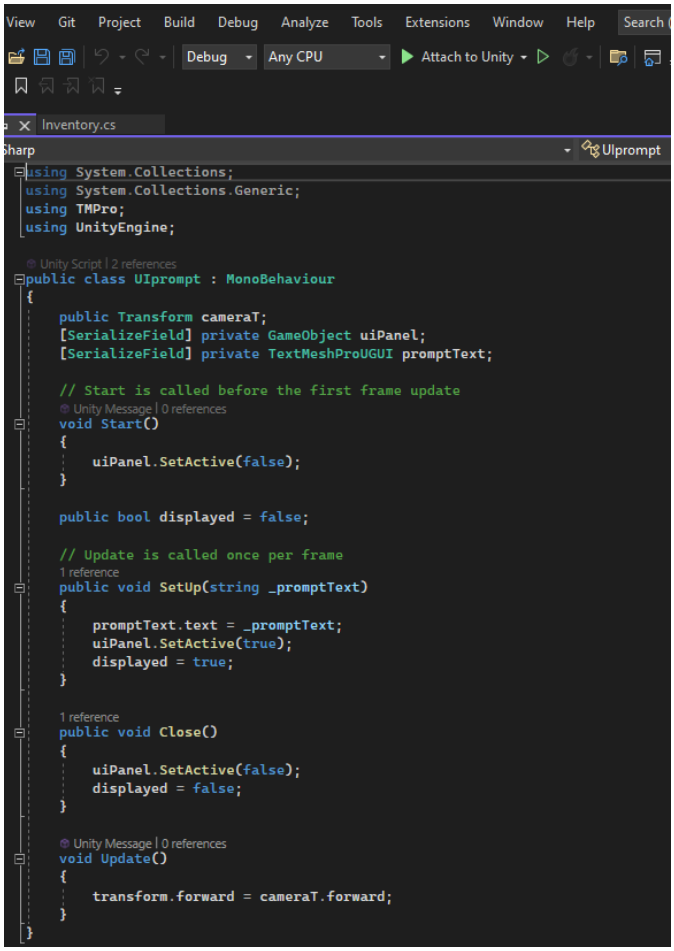
The interactionPoint is a transform variable that acts as a point on the player that is able to interact with interactable objects.

The interactables mask is the mask to allow the code to be able to correctly identify objects that are interactable as they will be under this mask

The col array is an array to store all collisions between interactable objects and the player. This has a max size of 3 since it is first of all unlikely to have 3 collisions with interactable objects and also to not complicate the code and cause issues.

```
Unity Message | 0 references
private void Update()
{
    numFound = Physics.OverlapSphereNonAlloc(interactionPoint.position, interactionPointRadius,col, interactables);

    if(numFound > 0)
    {
        _interactable = col[0].GetComponent<power>();

        if(_interactable != null)
        {
            if (!uiPrompt.displayed) uiPrompt.SetUp("Press E to turn ON and OFF");

            if (Keyboard.current.eKey.wasPressedThisFrame) _interactable.Interact(this);
        }



    }
    else
    {
        if (_interactable != null) _interactable = null;
        if (uiPrompt.displayed) uiPrompt.Close();
    }
}

Unity Message | 0 references
private void OnDrawGizmos()
```

This code updates every frame since it is part of the update function. Every time it is ran it checks for how many items that are colliding with the sphere made at the interactionPoint with the interactionPointRadius as the radius of the sphere and are also part of the interactables layer mask. Col is to store the output of the function. If it has collided with something the code will get the component for the interface and will check to make sure it is not null and then also checks if the E key is pressed. If this is true it, then calls goes to the interact references and preforms the actions that are part of it. Else if the UI is currently displayed it will close and set the interactable to null to reset it.



This code is for the UI prompt interface. A getter method to access the interactionPrompt and also reference the interactor

L Pinteraction (Mono Script)

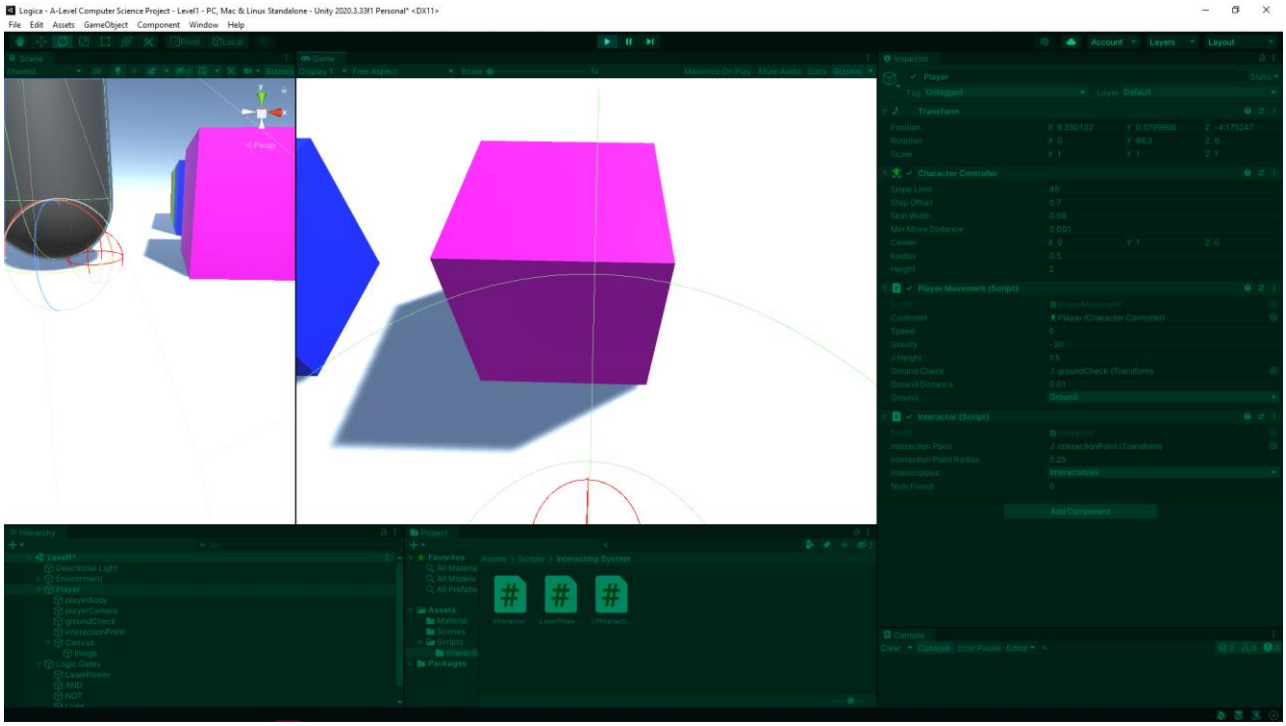
Assembly Information

Filename Assembly-CSharp.dll

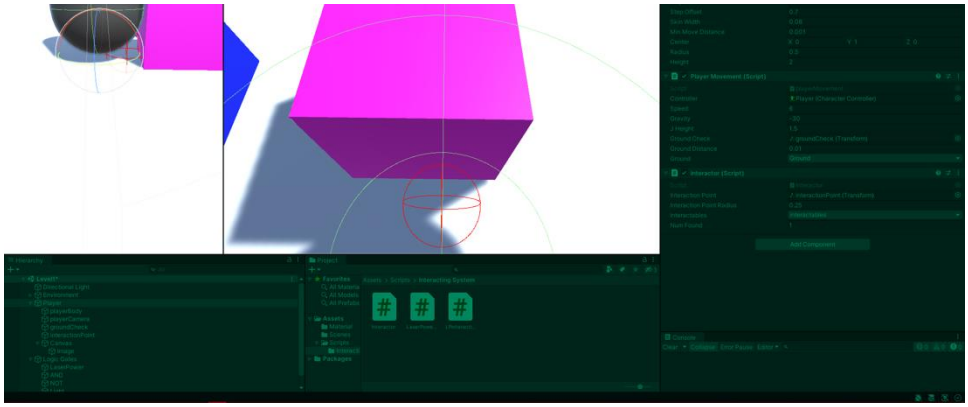
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LPinteraction : MonoBehaviour, LaserPowerUI
{

  [SerializeField] private string prompt;

  public string interactionPrompt => prompt;
  public bool Interact(Interactor interactor)
  {
    Debug.Log(message:"Turning On the power...");
    return true;
  }
}
```

All the previous code that has referenced the function Interact will be directed to here and it will return true and debug the message.

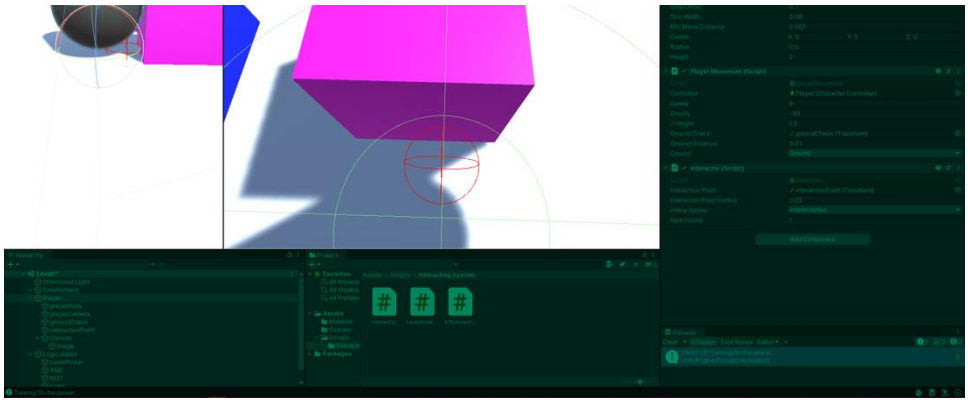
Below is a couple of print screens showcasing the code working



As you can see, the player is not colliding with the switch and thus numFound in the inspector is 0 and also no debug message is shown



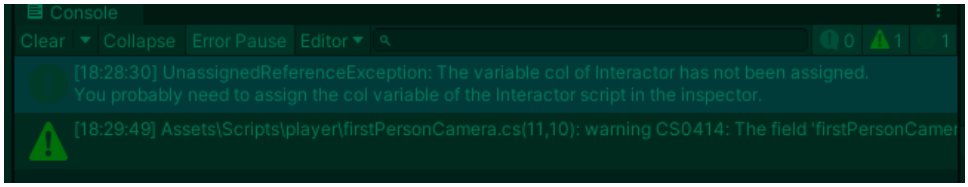
NumFound is 1 since it is colliding with interactable switch



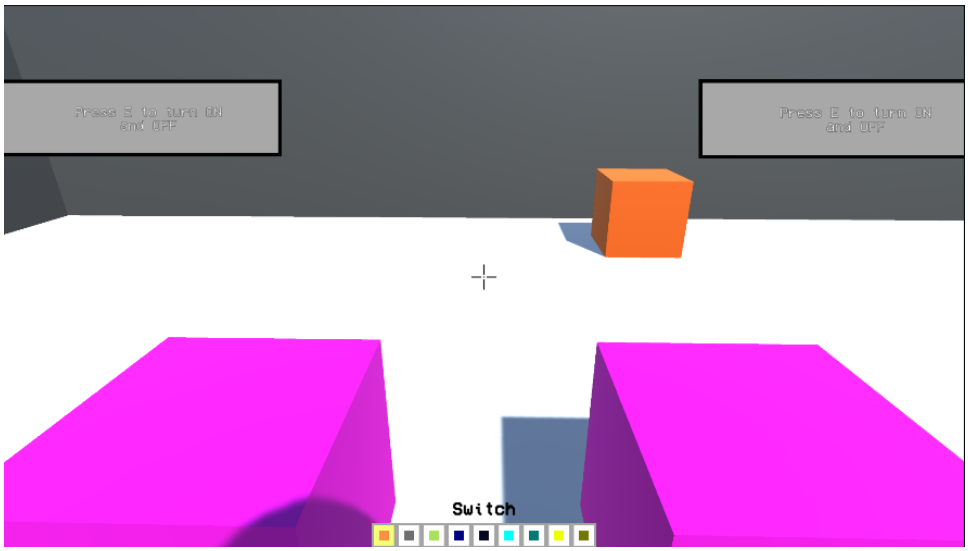
It also now debugs the message

And also shows the UI prompt below

Later on, in the project during my attempt at trying to fix another problem I ran into a problem with the interaction code.



I found an error when testing that whenever you walked next to two switches and both their pop ups appear it would give an error when you tried to interact



I decided to adjust the script to try and delete / close both UI displayed whenever you walk away however you will only turn on one of them whenever you interact

My first approach was to create a foreach that would iterate though the array for every object and do the original code but this time accessing all the variables using c which chosen since it is a collider.

```
19      private void Update()
20      {
21          numFound = Physics.OverlapSphereNonAlloc(interactionPoint.position, interactionPointRadius,col, interactables); // checks for interactrable object
22          foreach (Collider c in col)
23          {
24              if (c != null)
25              {
26                  if (numFound > 0)
27                  {
28                      if (!c.gameObject.GetComponentInChildren<UIprompt>().displayed) c.gameObject.GetComponentInChildren<UIprompt>().SetUp("Press E to turn ON and OFF"
29
30                      if (Keyboard.current.eKey.wasPressedThisFrame) _interactable.Interact(this);
31                  }
32                  else
33                  {
34                      if (c.gameObject.GetComponentInChildren<UIprompt>().displayed) c.gameObject.GetComponentInChildren<UIprompt>().Close();
35                  }
36              }
37          }
38      }
```

However, this did not work as some colliders stayed in the array and did not change

To fix this I instead made a for loop to iterate through each collider in the oldCol array and close them before it goes through the current Col array and sets it up. This resets the switch's pop ups back to closed. I also made pressed E a variable so that I can use that as a branch from my code. This also makes it easier to read for other people since the line before did the same thing however it was all in one line making it harder to read

```
public class Interactor : MonoBehaviour
{
    [SerializeField] private Transform interactionPoint;
    [SerializeField] private float interactionPointRadius = 0.5f;
    [SerializeField] private LayerMask interactables;
    [SerializeField] private UIprompt uiPrompt;
    Collider[] col = new Collider[5];
    Collider[] oldCol = new Collider[5];
    [SerializeField] private int numFound;
    public bool pressed;

    Unity Message | 0 references
    private void Update()
    {
        pressed = Input.GetKeyDown("e");
        col = new Collider[5];
        numFound = Physics.OverlapSphereNonAlloc(interactionPoint.position, interactionPointRadius, col, interactables); // checks for interactrable object
        for (int i = 0; i < 5; i++)
        {
            if (oldCol[i] != col[i])
            {
                if (oldCol[i] != null)
                {
                    if (oldCol[i].gameObject.GetComponentInChildren<UIprompt>().displayed)
                    {
                        oldCol[i].gameObject.GetComponentInChildren<UIprompt>().Close();
                    }
                }
                oldCol[i] = col[i];
            }
        }
        foreach (Collider c in col)
        {
            if (c != null)
            {
                if (numFound > 0)
                {
                    if (!c.gameObject.GetComponentInChildren<UIprompt>().displayed) c.gameObject.GetComponentInChildren<UIprompt>().SetUp("Press E to turn ON and OFF");

                    if (pressed)
                    {
                        c.gameObject.GetComponent<power>().Interact();
                        Debug.Log("interacted");
                    }
                }
            }
        }
    }
```

Here is the fixed code above.

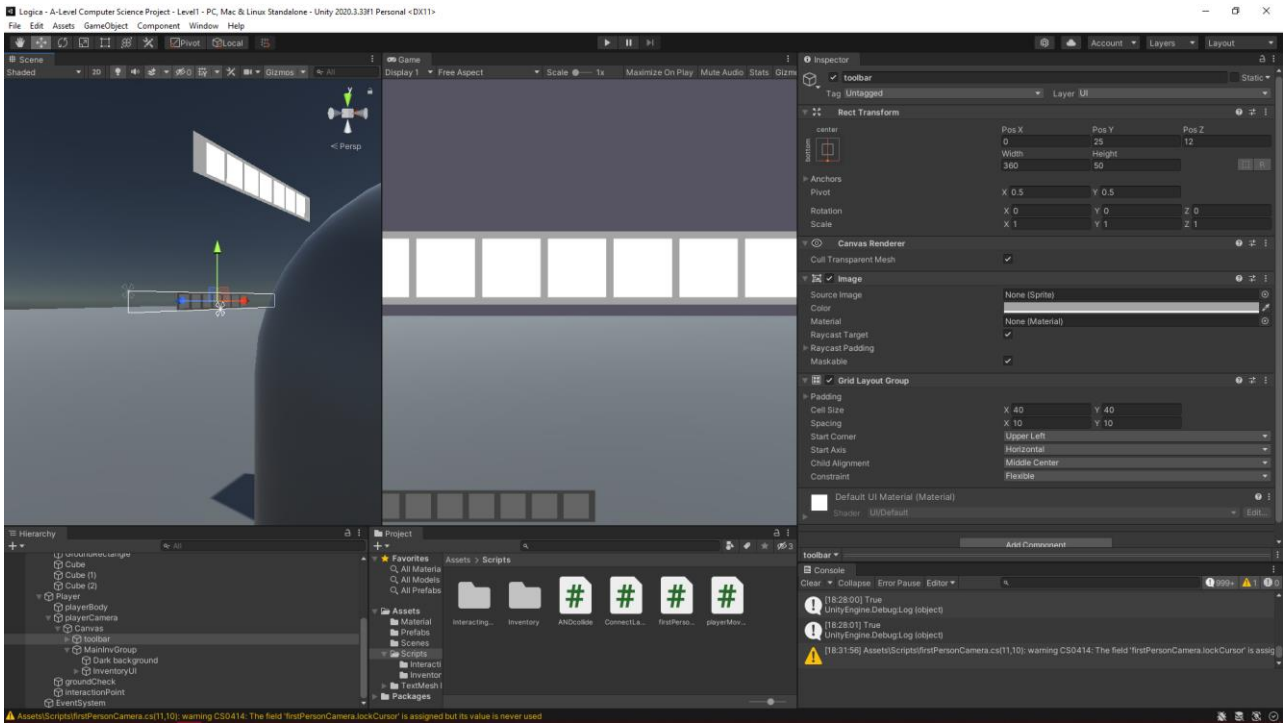
After further consultation, my client is content with the functionality of the interaction.

Inventory

I started off by making objects for the hot bar and inventory. The inventory will be an enlarged version of the hotbar and allows for players to move and switch items around to their liking.

The hotbar will act as a way to show the players what they are selecting and what they have in the inventory.

I made these objects by creating a canvas and then creating a grey image for the background then images for each slot and used the grid layout group to structure and order them. I did this for both hotbar and inventory.



I next started to create the openInv script which will open the inventory by setting the canvas as active allowing for the player to see it. It will also stop movement, ability to look around, being able to place and break blocks, turn the mouse cursor on so they are able to see what their mouse is on and also to darken the game so that the inventory pops out.

```
{
    public GameObject Inventory;
    public GameObject Player;
    public GameObject playerCam;
    public bool interfaceOn = false;
```

First, I declared all variables I will use. Some variables that I need to disable are currently missing since I have not coded them yet but will add later on. All the public Game Objects are to be able to access the correct scripts needed to disable and enable certain things. They are public so that I can assign the correct game object from the scene in the inspector.

The interfaceOn variable acts as a Boolean to check if the inventory is open or close so it is able to act accordingly when the open inventory key is pressed

```
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        Inventory.SetActive(false);
    }
```

This turns off the inventory canvas at the start of the game so that the UI does not get in the way while playing and let the player be able to see

```
// Update is called once per frame
Unity Message | 0 references
void LateUpdate()
{
    playerMovement move = Player.GetComponent<playerMovement>();
    firstPersonCamera cam = playerCam.GetComponent<firstPersonCamera>();

    if (Input.GetKeyUp("q") && interfaceOn == true)

    {
        interfaceOn = false;
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;
        move.enabled = true;
        cam.enabled = true;

    }

   else if (Input.GetKeyUp("q") && interfaceOn == false)

    {
        interfaceOn = true;
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
        move.enabled = false;
        cam.enabled = false;

    }
    Inventory.SetActive(interfaceOn);
}
```

This section of code inside the lateUpdates checks for whenever the player is pressing the open inventory button (Q) and if the inventory is already open or not. When the player is trying to open the inventory. It will turn of the ability to look around and move the player object around. It will also turn on the mouse. When you close the inventory, it reverses those changes so that you are able to play the game

My next script will be an attempt at making a drag and drop inventory system. I will do this by dragging the item and making it a child of the inventory and not the slot. This means that the item you are dragging will be on top of everything on the screen and will be set to be a child of a slot once you 'drop' it on a slot.

```
Unity Script | 2 references
public class DraggableItem : MonoBehaviour, IBeginDragHandler, IDragHandler,IEndDragHandler
{
    [HideInInspector] public Transform parentAfterDr
    public Transform tInv;
    Vector3 offset;
    public string destinationTag = "DropArea";
```

interface UnityEngine.EventSystems.IBeginDragHandler

The script above declares each variable type and the variable names.

The variable parentAfterDrag is the new parent assigned to the item you are dragging. This is public to be access the item to be able to use the item and access the slot.

The public tInv variable is a transform to be able to move the item and it is public so that you can assign the correct object from the scene.

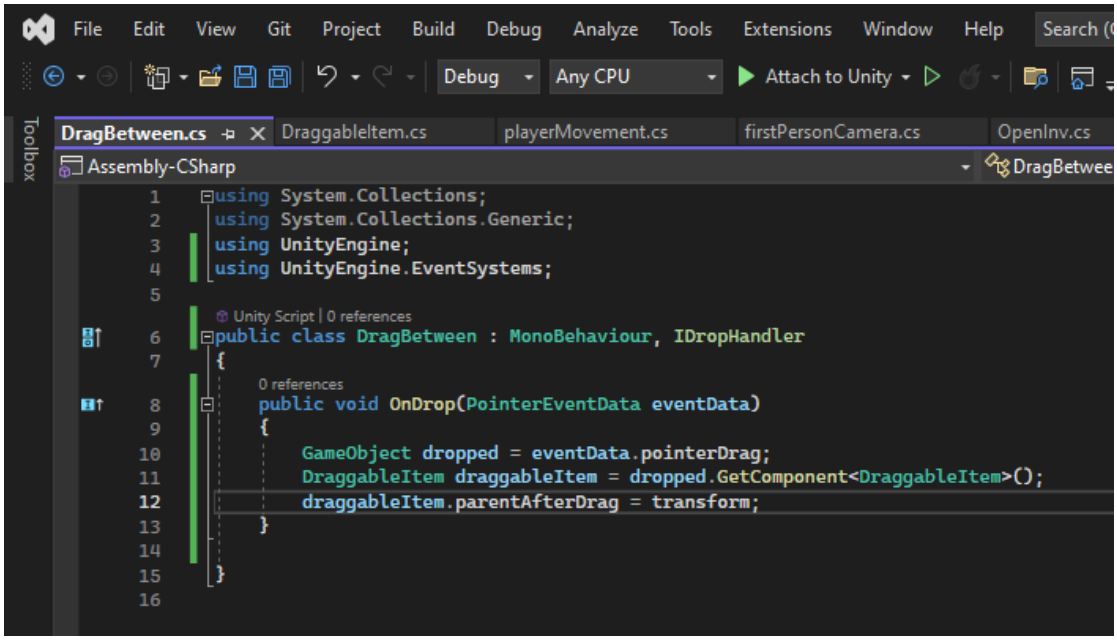
```
1 reference
Vector3 MouseWorldPosition()
{
    var mouseScreenPos = Input.mousePosition;
    mouseScreenPos.z = Camera.main.WorldToScreenPoint(transform.position).z;
    return Camera.main.ScreenToWorldPoint(mouseScreenPos);
}

0 references
public void OnBeginDrag(PointerEventData eventData)
{
    Debug.Log(message: "Begin Drag");
    parentAfterDrag = transform.parent;
    transform.SetParent(tInv);
    transform.SetAsLastSibling();
}

0 references
public void OnDrag(PointerEventData eventData)
{
    Debug.Log(message: "Dragging");
    transform.position = MouseWorldPosition() + offset;
}
0 references
public void OnEndDrag(PointerEventData eventData)
{
    Debug.Log(message: "End Drag");
    transform.SetParent(parentAfterDrag);
}
```

The code above sets a new Vector3 as the mouse z position and translates that as accordingly to the world position and camera position.

Next whenever the player begins to drag it sets the parentAfterDrag to the current parent. And the set the parent of the item we are moving to as the inventory and not the slot and then puts the item at the bottom of the hierarchy to show above all the other objects and images.

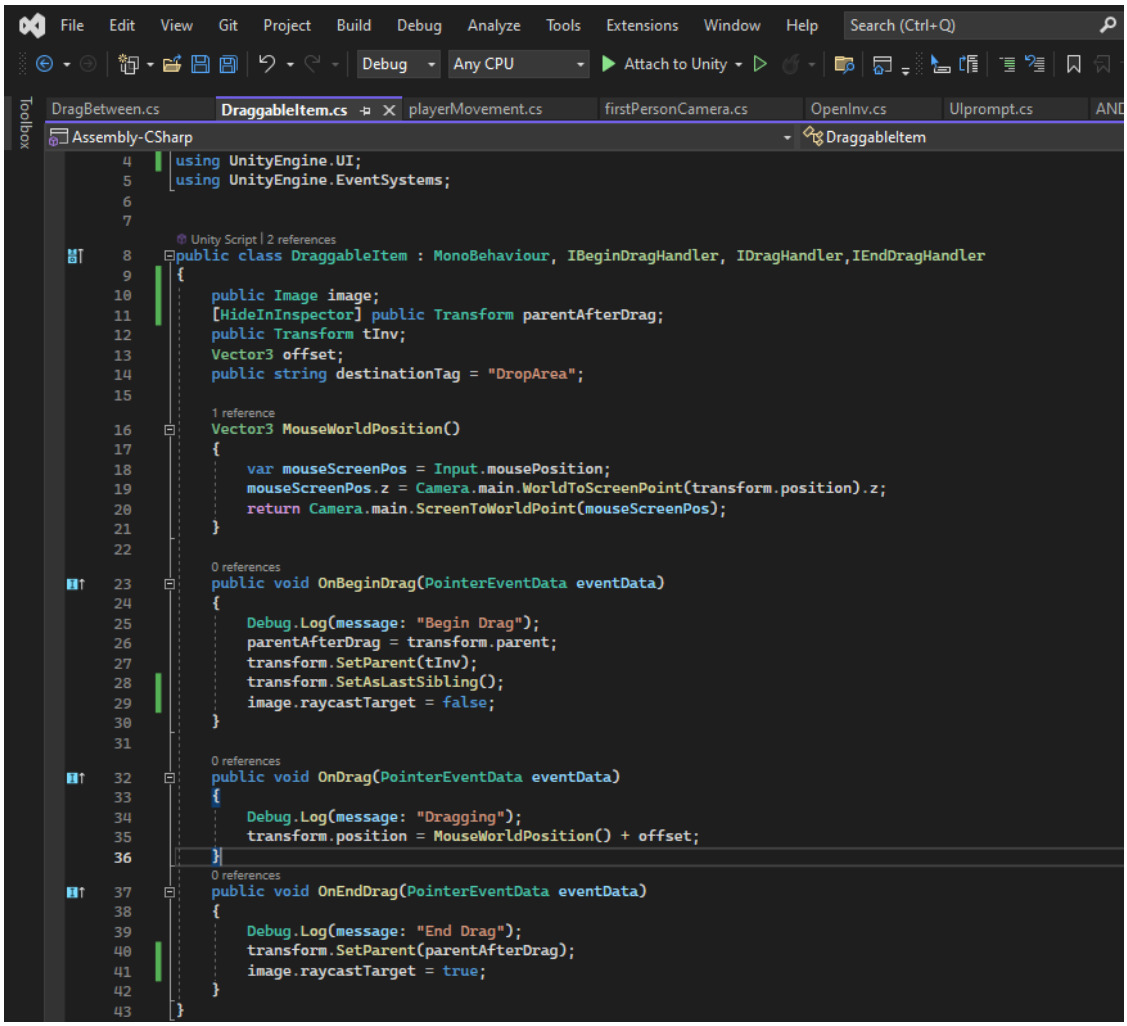
While the player is dragging the item it will move the item based of the mouse position and the offset.

The code will finally set the item as the parentAfterDrag which will be changed to the new slot in another script.

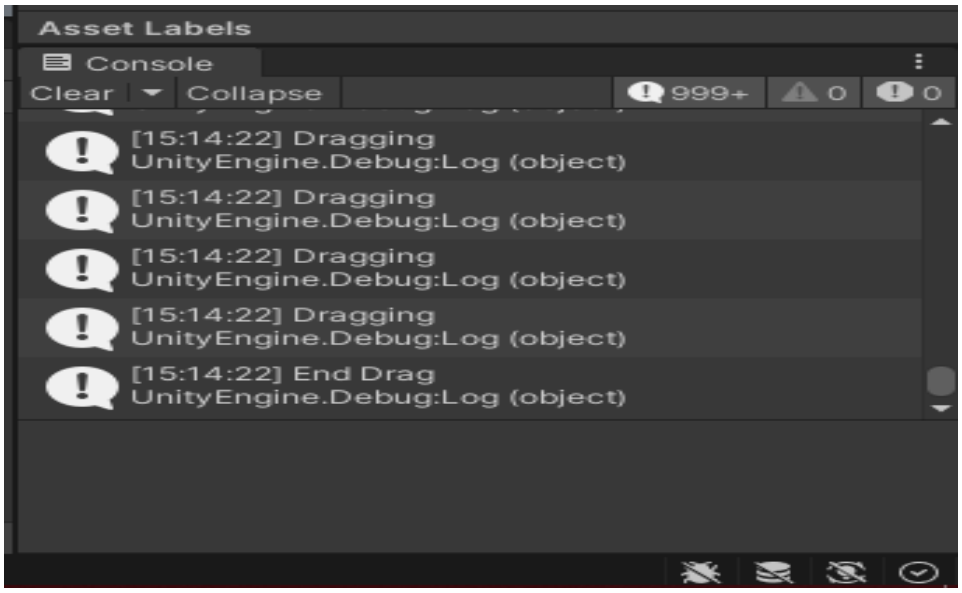


This code will set the new parentAfterDrag to the new item that the item is being dropped on. It does this by setting dropped game object that the mouse is on. This then gets the draggable item

component and access' the variable that is public due to encapsulation. This then allows the script to set the new parentAfterDrag to the new slot. However, the code is not finished due to the image we are dragging must ignore ray cast so that it doesn't cause issues. I fixed this issue below by creating a public image and setting the raycastTarget to false or true depending on if it is being dragged or not.

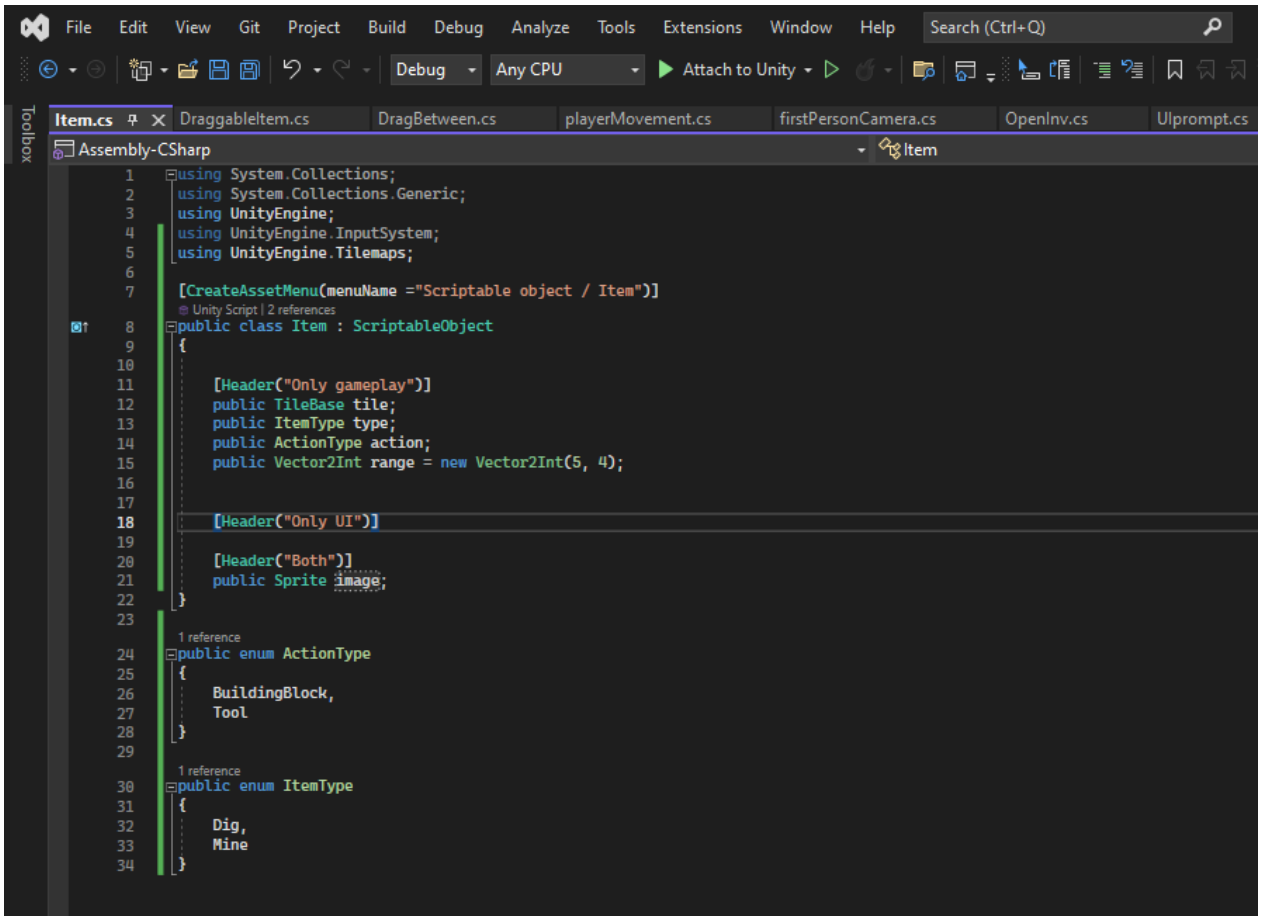


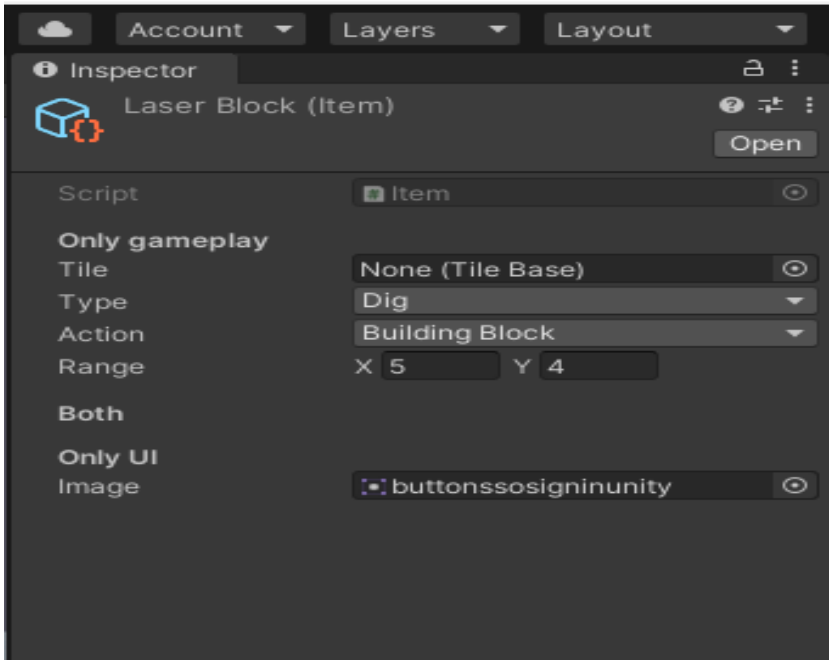
This shows the Inventory drag and drop working as expected in the console with the debug logs.



Next, I must start to add functionality to the items.

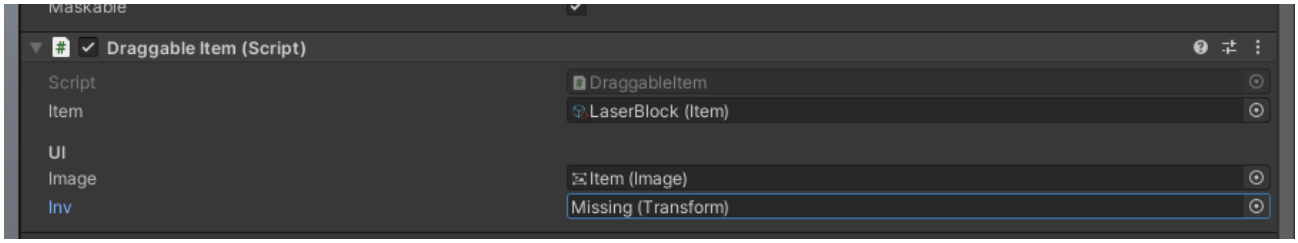
I did this by making a scriptable object called Item so that I can attach game objects to them whenever I am placing the items down. I described what action it does and what type of item it is. I also declared the images key information needed about the item.





This shows the Item scriptable object information in the inspector.

However, when I turned my items and slots into prefabs, I can no longer assign MainInvGroup into Inv



If I created MainInvGroup as a prefab, then Unity does not allow me set MainInvGroup as a parent because of it being a prefab and it can cause errors.

Since I did not know how to fix this, I tried a new approach to coding the inventory this time I will store the items in a list and switch their positions. This allows me to easily access the items.

```
Unity Script (1 asset reference) | 2 references
public class Inventory : MonoBehaviour
{
    public List<int> inv;

    public List<Item> items;

    [SerializeField] List<Image> slotImgs;
    [SerializeField] List<Image> barImgs;
    [SerializeField] List<RectTransform> slotRTs;
    [SerializeField] List<RectTransform> barRTs;

    Item item;

    [SerializeField] int dragItem;

    public Vector2 mPos;

    [SerializeField] Image dragImg;
    [SerializeField] RectTransform dragRT;
    [SerializeField] RectTransform selectedRT;
    float temp;

    public int select; // the index of my hotbar
```

The inv variable is a list of numbers and each number represents an item in the items list

I first declare the list of items and also the transforms of all the slots and item images.

I then declare a dragImg variable which is acts as an invisible image to switch the position when off when dragging an item.

SelectedRT is the image to represent what item you have selected in your hotbar.

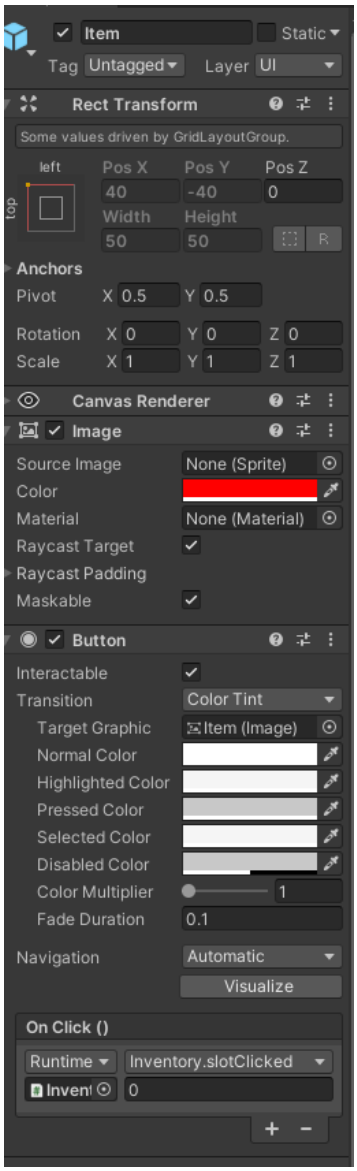
```
void Update()
{

    //slotRender
    for (int i = 0; i < inv.Count; i += 1)
    {
        if (inv[i] == 999)
        {
            slotImgs[i].color = new Color32(255, 255, 255, 0);
            barImgs[i].color = new Color32(255, 255, 255, 0);
        }
        else
        {
            slotImgs[i].color = new Color32(255, 255, 255, 255);
            barImgs[i].color = new Color32(255, 255, 255, 255);

            item = items[inv[i]];
            slotImgs[i].sprite = item.sprite;
            barImgs[i].sprite = item.sprite;
        }
    }

    mPos = Input.mousePosition - new Vector3(960, 540, 0);

    if (dragItem != 999)
    {
        dragImg.color = new Color32(255, 255, 255, 255);
        dragImg.sprite = items[dragItem].sprite;

        dragRT.anchoredPosition = mPos;
    }
    else
    {
        dragImg.color = new Color32(255, 255, 255, 0);
    }

    temp = Input.GetAxis("Mouse ScrollWheel");


}


0 references
public void slotClicked(int j)
{
    int a = inv[j];
    inv[j] = dragItem;
    dragItem = a;
}
```

The code above sets each of the images' alpha to 0 when iterating through to check if they have been switched with the empty object (dragItem)

The first part loops through each item in the inventory (stored in a list called "inv") and checks if it has a value of 999. If it does, it sets the corresponding slot and bar images' alpha value to 0, effectively hiding them. If the item is not 999, it sets the alpha value to 255, showing the slot and bar images. It then gets the corresponding item object from an array called "items" using the index in the inventory list and sets the slot and bar images' sprite to the sprite of that item.

The second part updates the position and appearance of a dragged item. It checks if a variable called "dragItem" has a value of 999, which indicates that no item is currently being dragged. If an item is

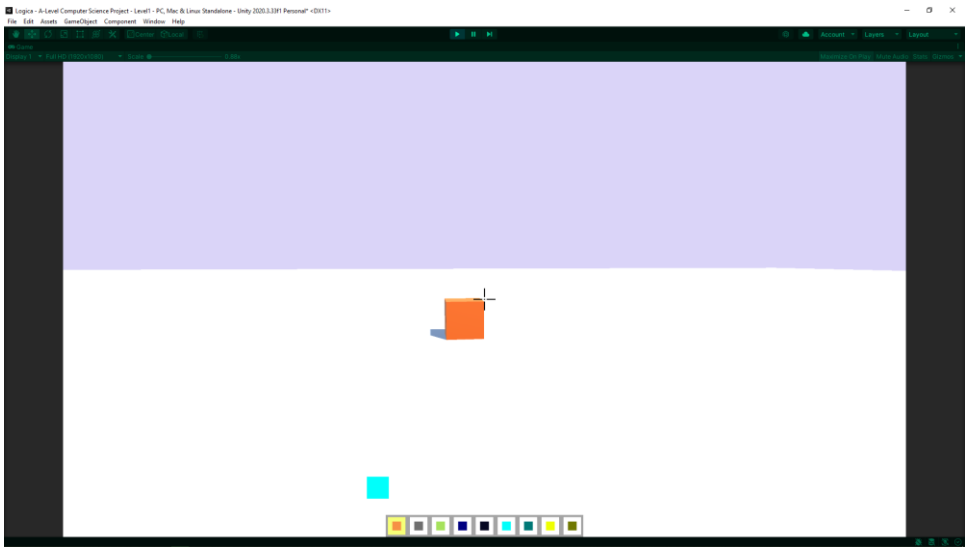
being dragged, it sets the alpha value of the drag image to 255, showing it on the screen, and sets its sprite to the sprite of the dragged item. It also sets the anchored position of the RectTransform component attached to the drag image to the current mouse position relative to the centre of the screen. If no item is being dragged, it sets the alpha value of the drag image to 0, effectively hiding it.



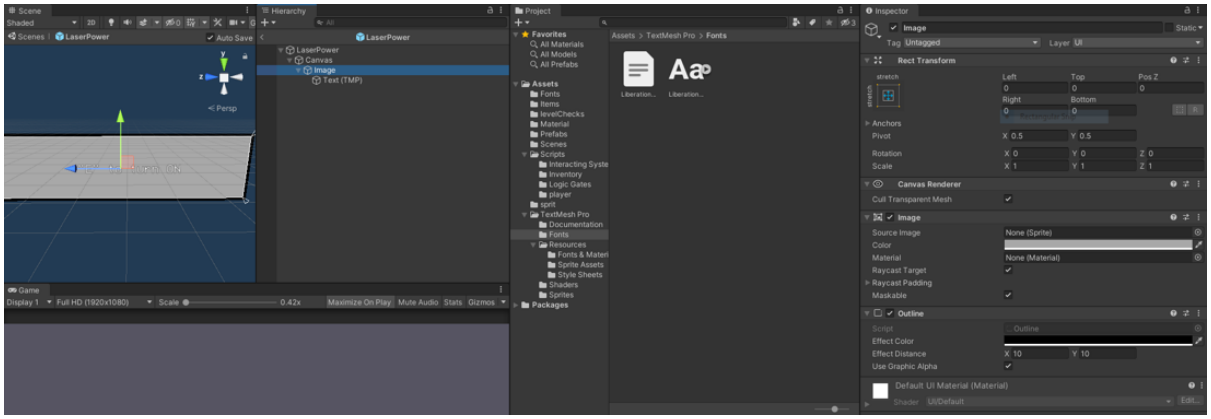
The procedure slotClicked is accessed by the button with its corresponding item passed in and this function swaps the item in the clicked slot with the currently dragged item, allowing the player to drag and drop items within their inventory.

```
    {
        dragImg.color = new Color32(255, 255, 255, 0);
    }

    temp = Input.GetAxis("Mouse ScrollWheel");
    if (temp == 0.1f)
    {
        select -= 1;
    }
    if (temp == -0.1f)
    {
        select += 1;
    }

    if (select > 8)
    {
        select = 0;
    }

    if (select < 0)
    {
        select = 8;
    }

    selectedRT.anchoredPosition = barRTs[select].anchoredPosition;
}
```

This code scrolls through the hot bar and sets the selectedRT position to the same as the items position in the inventory.



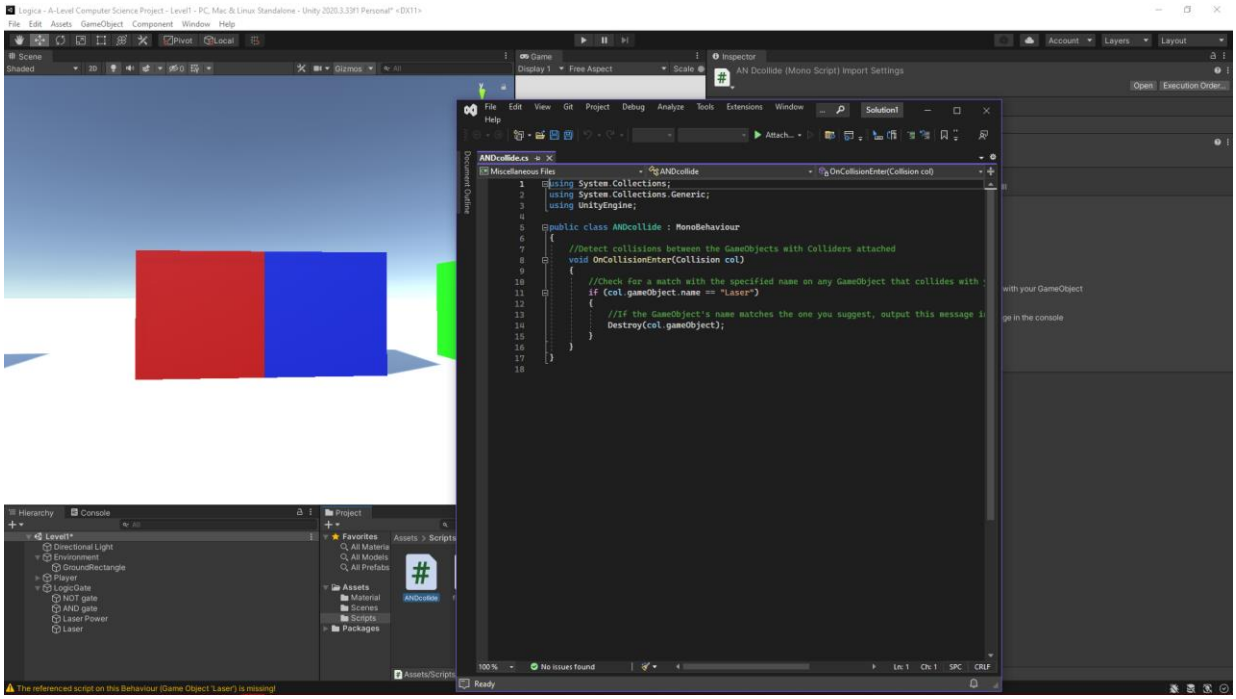
After some consolation with my client, they suggested to redesign the colours for my pop Up to not be so vibrant and an eyesore since it was very orange, and the colours did not match at all. I ended up going with grey and black and the client agreed and said it was much better.



My client is content with the functionality of the inventory within the program.

Generating Lasers

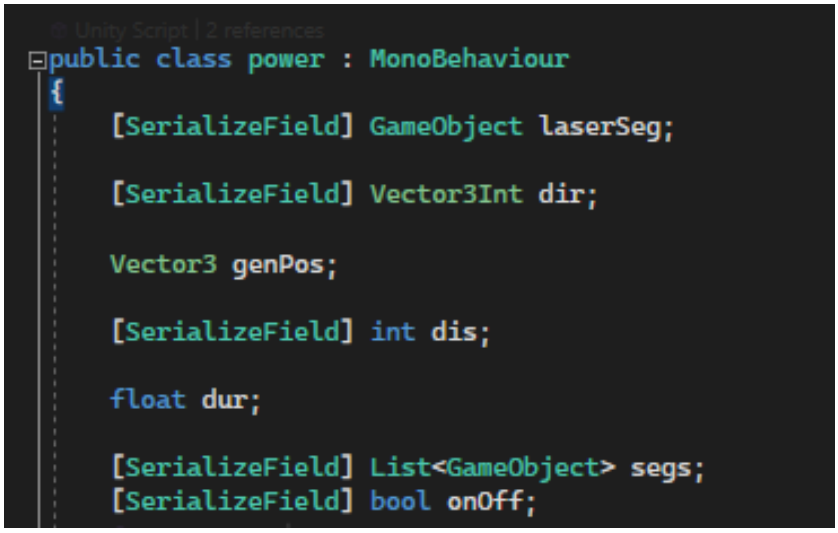
First, I decided to use onCollisionEnter to check when lasers collide with blocks and use collisions to determine when they should be affected etc. This was my approach using pseudocode.



I initially wanted to follow the pseudocode for this part. However, I thought of another way to create the logic gates so that is more immersive, and more efficient.

By using ray casts that can act as lasers I will be able to create a logic gate system more effectively

I needed to write the script to generate the lasers and check for how different lasers should generate based on the angle it is placed at and also code the lasers bouncing off of the mirrors. I will also have to check for any updates to a circuit when placing a block.



The laserSeg is the laser segment object

Dir is the direction that laser will be generated in; this is in Vector3 due to the world space being in 3D

The genPos variable is the position where the laser will be generated and is a vector3 due to the world space being 3D

dis is the distance that the laser is generated

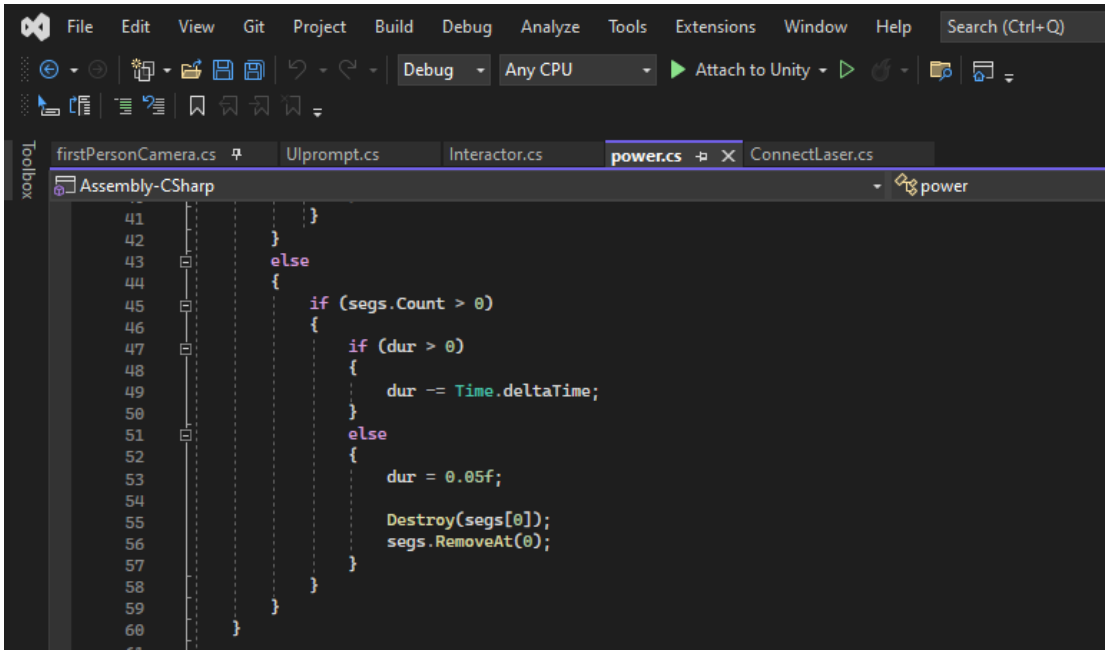
The dur variable is the duration / speed that the lasers are generated at.

The segs variable is the list of all the laser segments generated

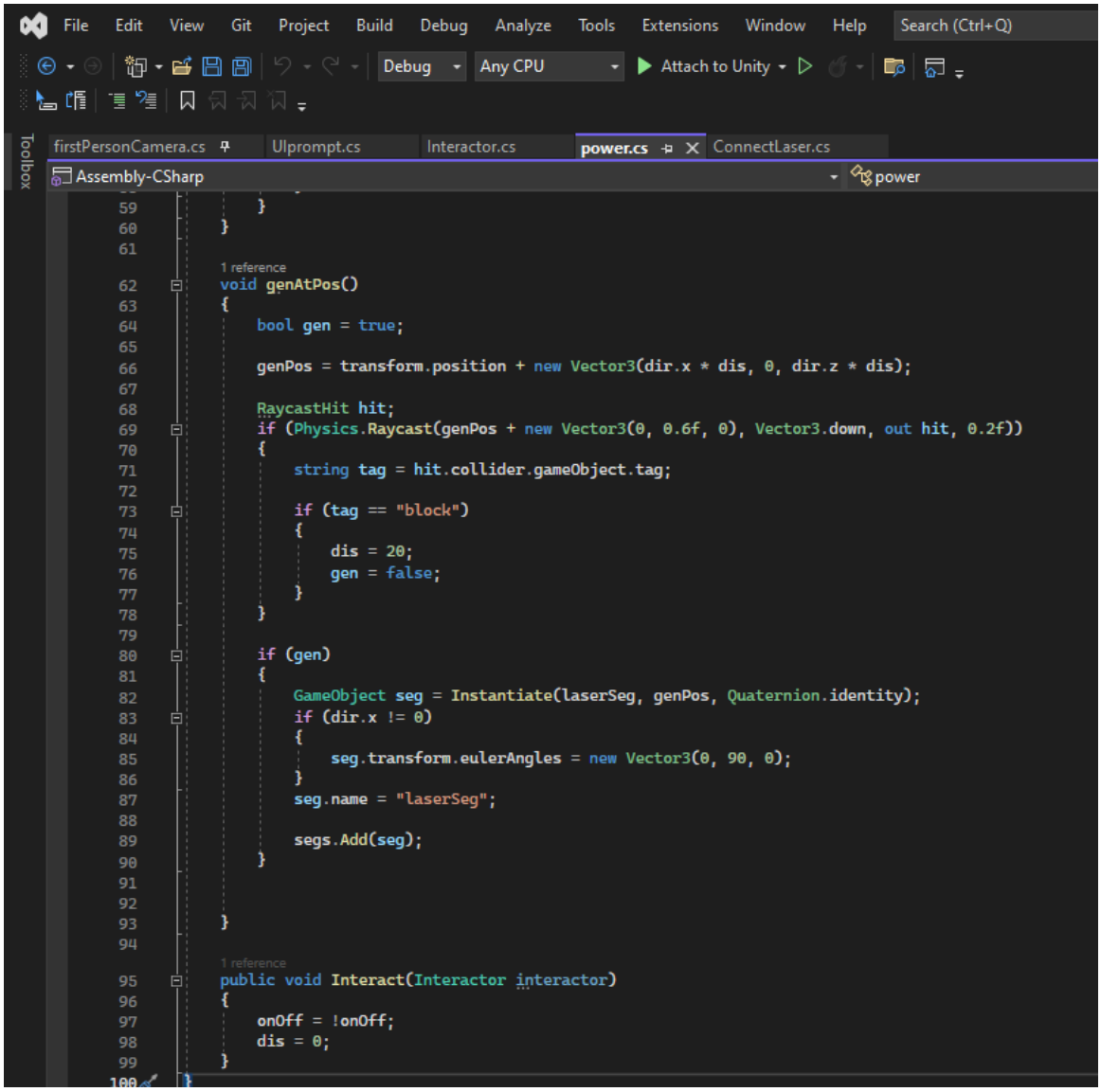
OnOff is a boolean variable that is the current of the lasers.

```
Unity Message | 0 references
void Start()
{
    dis = 0;
}

Unity Message | 0 references
void Update()
{
    if (onOff)
    {
        if (dis < 20)
        {
            if (dur > 0)
            {
                dur -= Time.deltaTime;
            }
            else
            {
                dur = 0.05f;
                dis += 1;
                genAtPos();
            }
        }
    }
    else
    {
        if (segs.Count > 0)
        {
            if (dur > 0)
```

The start function initialises the distance (dis) value to 0. This is to avoid any bugs when generating laser. The code then checks if the current is true, then checks if the distance is less than 20 and then checks if the duration is more than 0 which means it is currently generating. This then decreases the duration so eventually it will stop generating and reach the else statement which is where the duration is set to 0.05 and the dis is set to 1 which affects the direction later on. Then genAtPos is called



The else statement checks if a current is off, this does the same thing as the code above however it removes the lasers from the segs list and also destroys the lasers, this allows the lasers to un-generate whenever their source block is broken or removed.

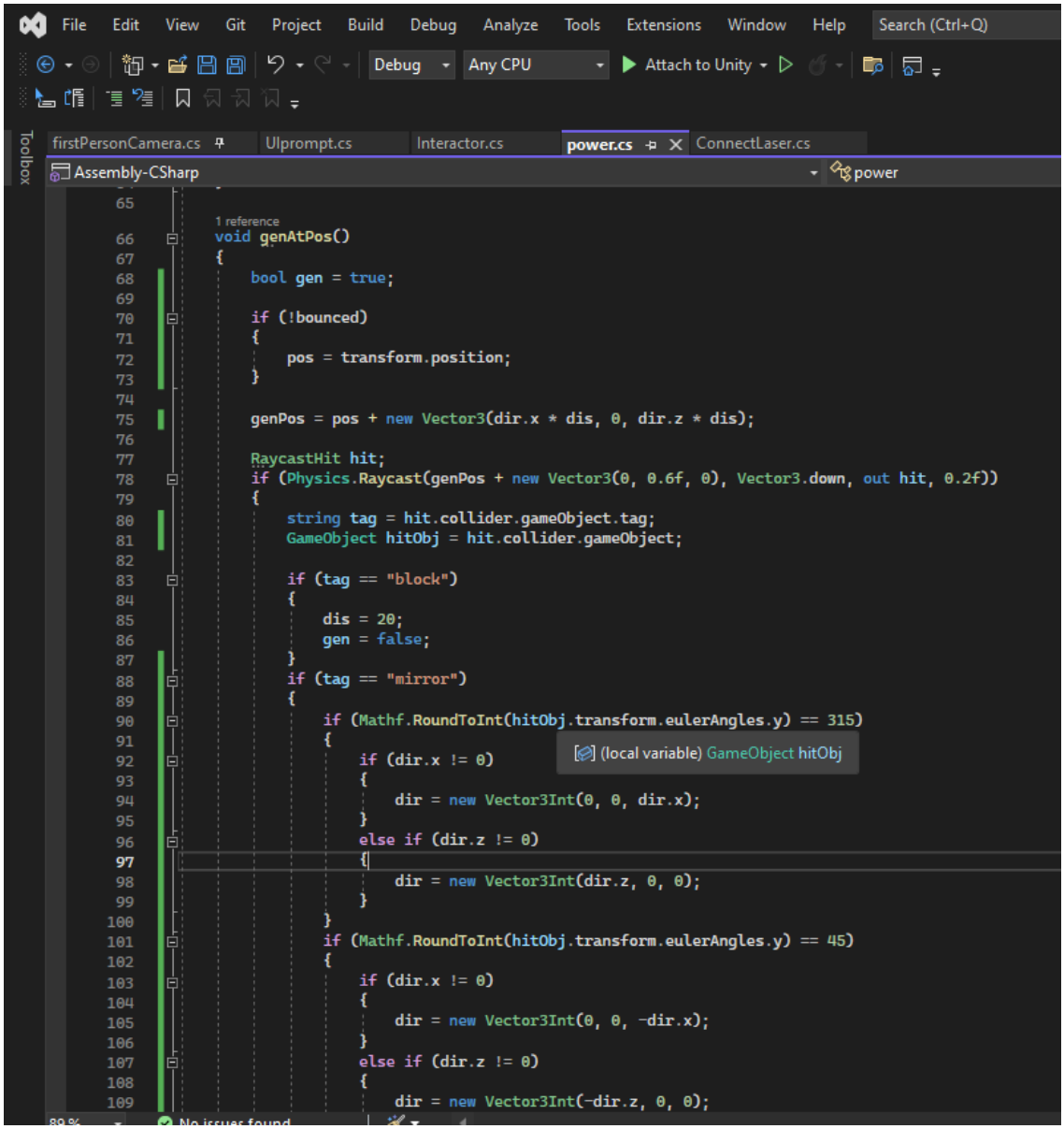


The void genAtPos is a function that generates the lasers based on direction. The direction being multiplied by dis acts to check if the lasers should be generating. If dis is one, then when multiplied by the direction it won't affect the direction and allow the laser to generate. However, if dis is 0 then it won't generate because there is no direction. It also creates a Boolean called gen which is set to true this is used later in the function to generate the lasers.

Line 69 checks if there's something where it wants to generate and makes the variable "hit" equal to what it hits. This allows us to check what the laser is hitting and thus can act depending on what the laser hits. For example, if it is a mirror, a block / wall or a gate.

Next the code checks if gen is true if it is it instantiates a laser prefab into the scene at the generation position and the default rotation for blocks. The code then changes the angle to make sure that the segment generates in the correct direction. The interact void below is to turn the switch on and off and set the distance to 0

The code also gets the tag of the object that has been hit to check how the laser should respond. Line 73 checks if it is a bloc. This will the stop laser from generating. The block tag will be applied to the walls that surround the level room.



I now added an if statement to check if it has bounced (from a mirror) and change pos to equal transform.position of the mirror

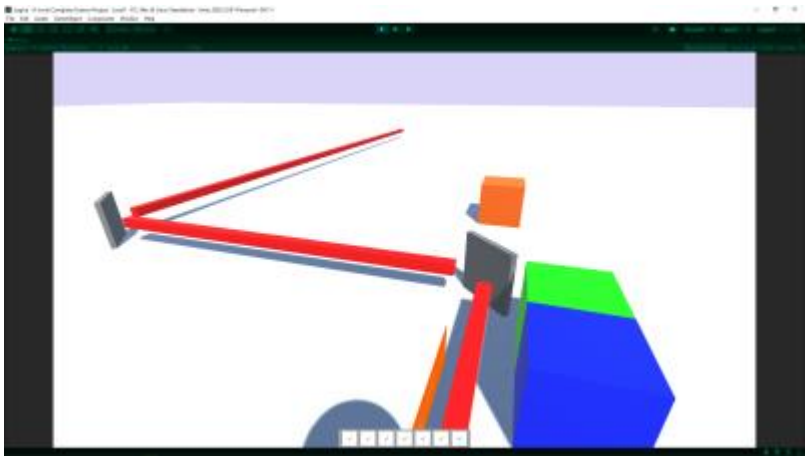
I created a new variable in the if statement for hitObj which is the object that has been hi. I now check if the tag of the object that is hit is a mirror. I coded an if statement to check if the rounded rotation to is equal to 315 which will generate it to the right or left depending on if the z or x is set to 0. The lines below check the same thing but if the objects rotation is 45

```
101             if (Mathf.RoundToInt(hitObj.transform.eulerAngles.y) == 45)
102             {
103                 if (dir.x != 0)
104                 {
105                     dir = new Vector3Int(0, 0, -dir.x);
106                 }
107                 else if (dir.z != 0)
108                 {
109                     dir = new Vector3Int(-dir.z, 0, 0);
110                 }
111             }
112             dis = 0;
113             pos = hitObj.transform.position;
114             bounced = true;
115             gen = false;
116         }
```

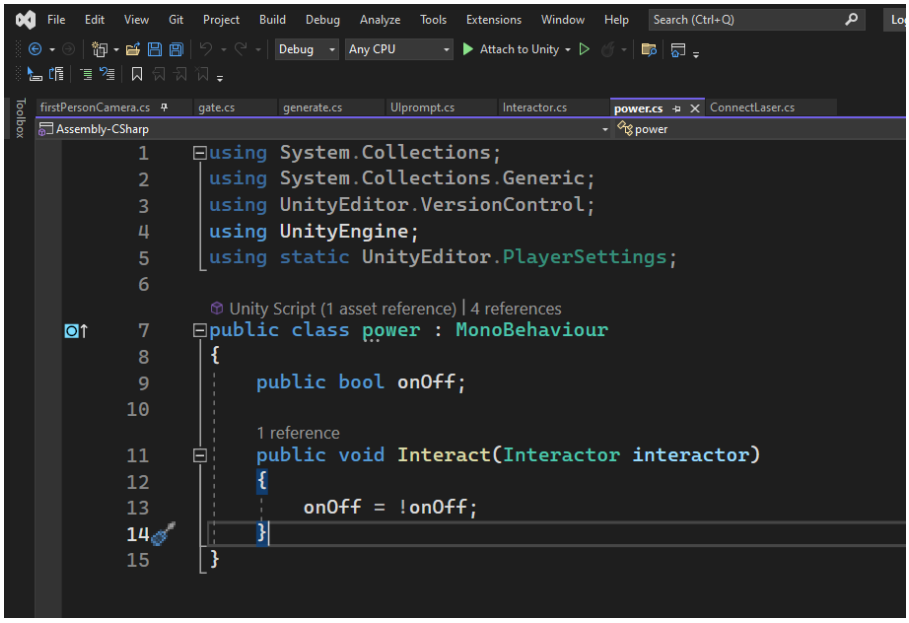
I then reset the values for distance, gen and set bounced to true.

```
133
134     public void Interact(Interactor interactor)
135     {
136         onOff = !onOff;
137         dis = 0;
138         if (onOff)
139         {
140             bounced = false;
141             dir = genDir;
142         }
143     }
144 }
```

In the interact void I check if the switch is on and the set bounce to false and direction equal to generation direction



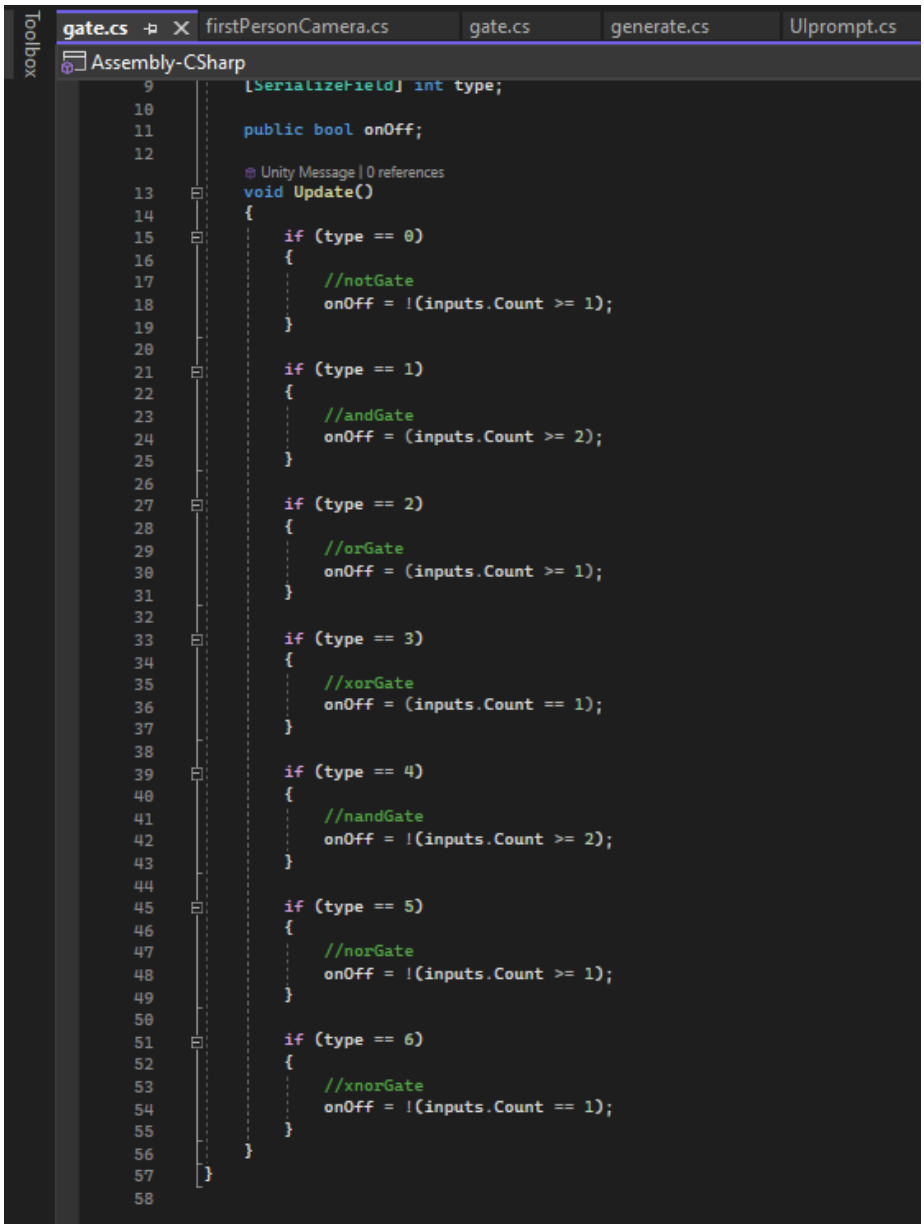
Here is a photo of me testing the lasers and them working as intended



I decided to create a new script called generate and move all the code to generate and have only the code above for power as power acts as the switch, and generate will be a script for multiple components thus making it more efficient due to not needing to have the Interact void in a generate script that will have other components that won't use this void.

After further consultation, my client is content with the current functionality of the lasers generating.

Logic Gates



I created a new gate script to have a list of inputs and origins. Inputs are all the objects hitting the current object with a laser. Origins are the switches that laser that has hit the object originates from. Each gate will have this script attached with the int which is set in the inspector this will allow me to check what gate it is and depending on the type how that affects the onOff variable which acts like a current. I decided to do this instead of tags is because I already use tags to distinguish switches, mirrors and gates in general otherwise that will have made the code far more inefficient. For the not gate which is type 0, it checks to make sure that if the inputs are not more than or equal to one it'll set the onOff to true. If the current that is going into the not gate is turned off, it will then be set to true since the inputs are more than or equal to one so then it returns false and the '!' sets it to true.

This is the same for all the gates and depending on the type of gate it will change the requirements for the onOff to be set to true or false. This code will also be used in the generate script to check for certain gates.

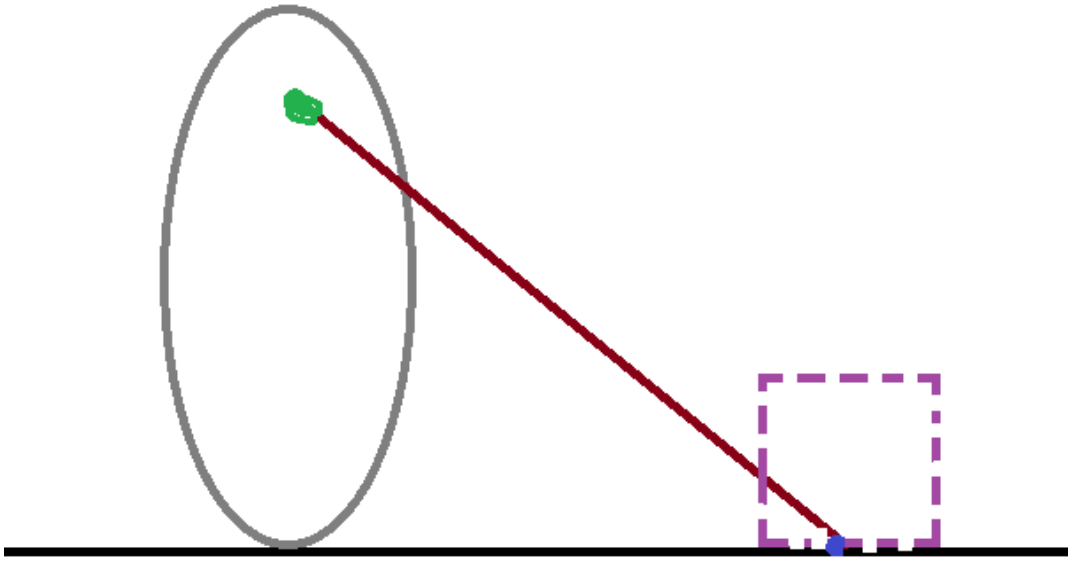
```
if (tag == "gate")
{
    if (hitObj.GetComponent<gate>().type == 0)
    {
        if (hitObj.transform.eulerAngles.y == 0 && dir.z == 1 || hitObj.transform.eulerAngles.y == 180 && dir.z == -1 || hitObj.transform.eulerAngles.y == 90 && dir.x == 1 || hitObj.transfor
            hitObj.GetComponent<gate>().inputs.Add(gameObject);
            gate = hitObj;
            dis = 100;
            gen = false;
        }
    }
    else
    {
        if (hitObj.transform.eulerAngles.y == 90 && dir.z != 0 || hitObj.transform.eulerAngles.y == 270 && dir.z != 0 || hitObj.transform.eulerAngles.y == 0 && dir.x != 0 || hitObj.transfor
            hitObj.GetComponent<gate>().inputs.Add(gameObject);
            gate = hitObj;
            dis = 100;
            gen = false;
        }
        else
        {
            dis = 100;
            gen = false;
        }
    }
}
```

This code checks if it is a not gate and depending on that it only checks for the input from one direction as a not gate only has one input compared to the other gates as they have two inputs. I coded to check more directions and object rotation

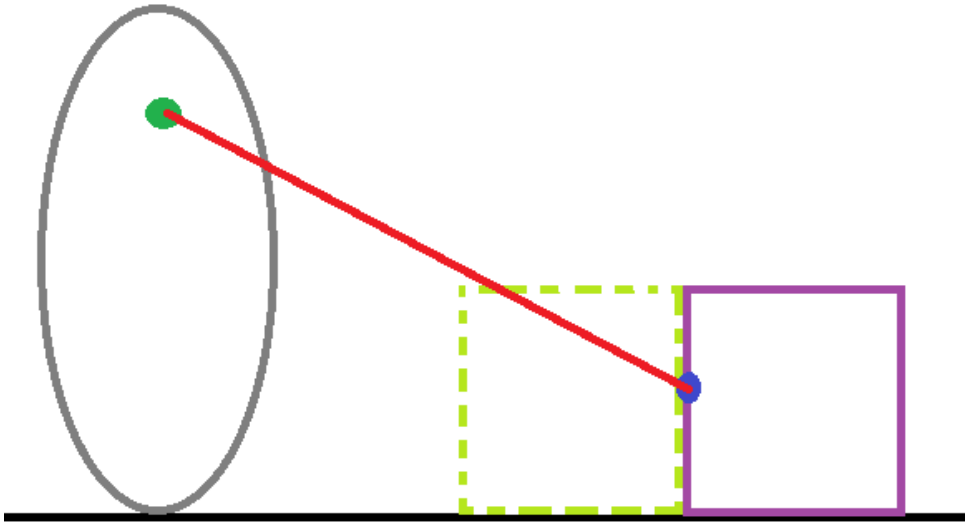
After further consultation, my client is content with the implementation of the logic gates.

Placing and breaking blocks

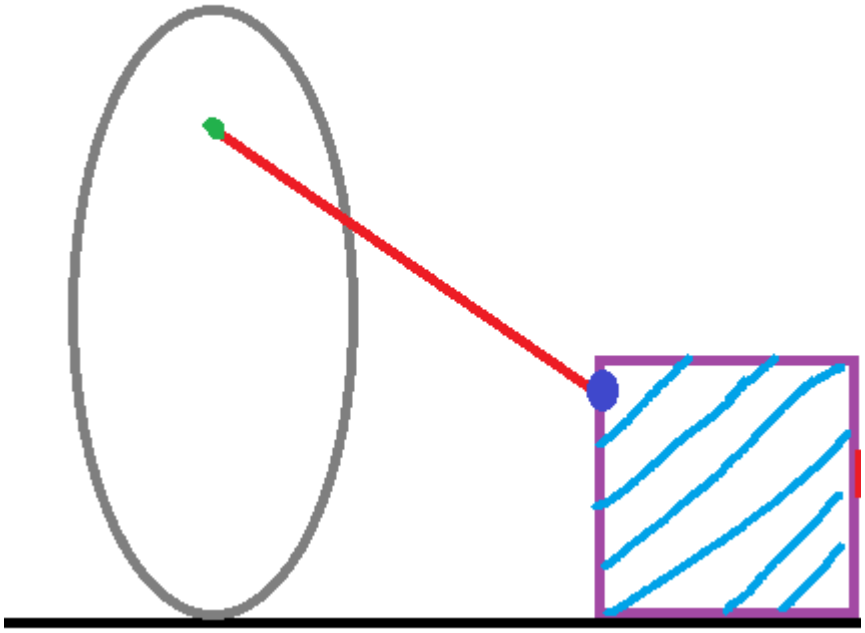
For placing and breaking blocks, I will use a raycast that will spawn an invisible line, visualised in the image below using the red line from the players camera which is where the green dot is and if / once it hits an object and it will stop and create a hit point and output that hit point as a variable to use, this is showcased below as the blue dot. I will then round the hit point so that the position of the placed block is only on whole numbers hence creating a grid format to follow by. The use of the mask will also be beneficial to get certain objects to ignore raycast such as the player object / capsule.



I will also have to make sure that whenever a block is placed, the block does not go inside one another but instead goes to the side of the block. My end goal is show in the drawing aswell. The purple cube is showing the already placed block that I am looking at and the dashed line cube is where the block should go if I decide to place it.



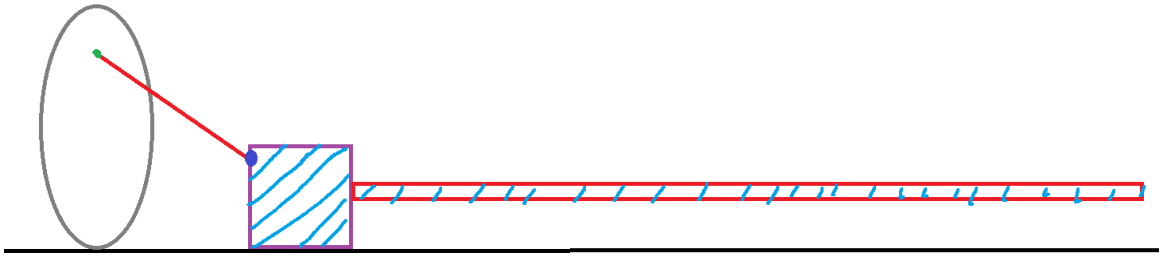
I will use the same function when breaking blocks, for example when I look at this block and the raycast detects and outputs it via the hit variable I will be able to access this object and destroy it using the destroy function in built within unity (‘Destroy (Object)’)



However, I will also need to check if I am generating lasers with the block

This will then also have to delete the lasers that are generated by the block being broken

I can do this by accessing the segs list attached to that object and contains all the lasers that were generated by that block. This will also remove the segments from the list as you delete them.



I started off by initialising the variables for this script.

```
6   public class sandbox : MonoBehaviour
7   {
8       Inventory inventoryC;
9       Item selected;
10      GameObject selectObj;
11      [SerializeField] Transform cameraT;
12      Vector3 posDistance;
13      Vector3 placePos;
14      [SerializeField] LayerMask mask;
15
        Unity Message | 0 references
```

The inventoryC variable is to allow the code to access the Inventory script since you need to access the item that is currently selected

The selected variable is of type Item, this variable will be the current item selected for the player to place.

selectObj is the object version of the current item selected.

The placePos variable is the position at which to place the currently selected item down. This is a Vector3 as the position of the blocks will be on a 3-dimensional plane.

The mask variable is a mask for the ray cast to hit, if an object is not part of this mask the ray cast will go through it.

```
Unity Message | 0 references
private void Awake()
{
    inventoryC = GetComponent<Inventory>();
}

Unity Message | 0 references
private void Update()
{
    selected = inventoryC.items[inventoryC.inv[inventoryC.select]];
    selectObj = selected.obj;



    if (Input.GetMouseButtonDown(0))
    {
        RaycastHit hit;

        if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask))
        {
            posDistance = hit.collider.transform.position - hit.point;
            placePos = new Vector3(Mathf.Round(hit.point.x), Mathf.Round(hit.point.y), Mathf.Round(hit.point.z));
            Debug.Log(placePos);
            if (placePos.y == 1)
            {
                Instantiate(selectObj, placePos, Quaternion.identity);
            }
        }


    }
}
```

In the awake function this sets the inventoryC variable to the players inventory script.

Next, we access the items list and the integer inv variable that represents an item, and we get the selected item of the inv list which means we get the item too.

I then set selectObj to the object that the selected variable represents.

After I check for if the player is pressing the left mouse button which is represented by 0 within Unity.

I then check using the in=built Physics Raycast function if I am looking at a solid surface. This makes sure I will not be able to place blocks randomly in the air. I check this by creating a straight line from the camera starting at the same position as the player camera and the same rotation hence checking if they can place in the same place that their camera is facing. The object that is hit is assigned in the ‘out hit’ part of the raycast. 5 is the distance of the line that is created, and the mask is for anything with those masks it will recognise as being hit. This needs to be used so that the line does not accidentally hit the players body object.

I set posDistance to the position of the collider of the object that the raycast hit takeaway from the exact point where the raycast hit. This was going to be used to find how close it is to the player to place it to make sure it is not too close to be placed (e.g., inside the player) and not too far away to be placed.

I then create a place position where I round the x y z of the hit point to create a new vector 3. I then make sure that placePos is on y level 1. This is due to making sure the player does not leave the perimeter and then instantiate the selected object with the placePos that I previously set.

However, placepos did not work as it did not as sometimes the blocks would go inside of itself

```
if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask))
{
    posDistance = hit.collider.transform.position - hit.point;
    pos = hit.point - cameraT.forward * 0.01f;
    placePos = new Vector3(Mathf.RoundToInt(pos.x), Mathf.RoundToInt(pos.y), Mathf.RoundToInt(pos.z));
    Debug.Log(placePos);
    if (placePos.y == 1)
    {
        Instantiate(selectObj, placePos, Quaternion.identity);
    }
}
```

To fix this I added “pos = hit.point - cameraTforward * 0.01f”. This fixed the issue of blocks placing within itself as it will minus the hit point by the direction that the player is looking in and multiply this by a constant this then makes sure to place the blocks next to each other rather than inside of one another.

Next, I coded the breaking blocks with the other mouse button.

```
51          if (Input.GetMouseButtonDown(1))
52          {
53              if(hit.collider.transform.position.y == 1)
54              {
55                  Debug.Log(hit.collider.gameObject.GetComponent<generate>().segs);
56                  foreach(GameObject seg in hit.collider.gameObject.GetComponent<generate>().segs)
57                  {
58                      delSegs.Add(seg);
59                  }
60                  Destroy(hit.collider.gameObject);
61              }
62          }
63
64      }
65
66      if(delSegs != null)
67      {
68          if (delSegs.Count > 0)
69          {
70              if (dur > 0)
71              {
72                  dur -= Time.deltaTime;
73              }
74              else
75              {
76                  dur = 0.05f;
77
78                  Destroy(delSegs[0]);
79                  delSegs.RemoveAt(0);
80              }
81
82
83
```

This checks for an input from the right mouse button. And destroys the block however it also must destroy the lasers that came out of the block.

It does this by adding all the current lasers into a separate list while also removing it in the segs list that it is accessing from a different script. After it goes through the list and destroys each laser segment while also removing it from the list.

```
Unity Message | 0 references
void Start()
{
    if (Mathf.RoundToInt(transform.eulerAngles.y) == 0)
    {
        genDir = new Vector3Int(0, 0, 1);
    }
```

I then also had a problem where when you place an object at a certain angle it would not generate, therefore I rounded the Y rotation of the object to check if it was close to 0. This fixed the issue.

Later in my program I had to add a lot of checks to assign different variables for the placed objects or add them to different lists so that I can update circuits.

I added things such as changing the rotation of mirrors since they should be rotated at a 45-degree angle and not 90.

I also had to assign the camera through the script as I could not assign it in the inspector because the powerBlock / switch is a prefab.

Below shows most conditions I had to check and what action to do on that object after placing down an object.

```
temp = (Mathf.RoundToInt(playerT.eulerAngles.y / 90))*90;
placeDir = new Vector3(0, temp, 0);
if (selected.name == "mirrorBlock")
{
    temp = (Mathf.RoundToInt(playerT.eulerAngles.y / 45)) * 45;
    if (temp % 90==0)
    {
        if (playerT.eulerAngles.y < temp)
        {
            temp = 315;
        }
        else if(playerT.eulerAngles.y > temp)
        {
            temp = 45;
        }
        else
        {
            Debug.Log("easter egg lol");
        }
    }
    placeDir = new Vector3(0, temp, 0);

}
GameObject placed = Instantiate(selectObj, placePos, Quaternion.Euler(placeDir));
if(placed.transform.childCount != 0)
{
    mirrorChild = placed.transform.GetChild(0).gameObject;
    mirrorChild.transform.localRotation = Quaternion.Euler(new Vector3(0, 45, 0));
}
placed.name = selectObj.name;
if (selected.name == "powerBlock")
{
    placed.GetComponentInChildren<UIprompt>().cameraT = cameraT;
    winCheck.powers.Add(placed.GetComponent<power>());
}
if(placed.GetComponent<generate>() != null)
{
    generates.Add(placed.GetComponent<generate>());
}
foreach(generate gen in generates)
{
    gen.genUpdate();
}
```

This code also adds to remove objects along with removing those objects from other lists and that would impact other lists that would need some lasers removing and there are a bunch of checks and conditions to ensure robustness of the code and to make sure that the circuit gets update accordingly visually and the lists also get updated correctly. The code shown below is not all the checks that I have done but it it's the first portion. Explaining each of the checks will be very repetitive since they all do the same thing but for different lists and objects.

The general idea of each condition is to remove the removed object from each list that it is within by checking each possible list to see if it is in any of the lists and then removing itself from those lists before it is deleted.

```
if (Input.GetMouseButtonDown(0))
{
    delObj = hit.collider.gameObject;
    if (delObj.tag != "block" && delObj.tag != "light")
    {
        if (hit.collider.transform.position.y == 1)
        {
            if (delObj.GetComponent<generate>() != null)
            {
                foreach (GameObject seg in delObj.GetComponent<generate>().segs)
                {
                    delSegs.Add(seg);
                }
                for (int i = 0; i < 2; i++)
                {
                    GameObject hitObj = null;
                    if (i == 0 )
                    {
                        hitObj = delObj.GetComponent<generate>().gate;
                    }
                    if (i == 1)
                    {
                        hitObj = delObj.GetComponent<generate>().unGate;
                    }
                    if (hitObj != null)
                    {
                        hitObj.GetComponent<gate>().inputs.Remove(delObj); // gets the gate component from the generate script of the object we are dest
                        if (hitObj.GetComponent<gate>() != null)
                        {
                            while (hitObj.GetComponent<gate>().origins.Count > 0)
                            {
                                hitObj.GetComponent<gate>().origins.RemoveAt(0);
                            }
                        }
                        else
                        {
                            if (hitObj.GetComponent<gate>().origins.Contains(delObj))
                            {
                                hitObj.GetComponent<gate>().origins.Remove(delObj);
                            }
                        }
                    }
                }
                if (delObj.GetComponent<gate>() != null)
                {
                    if (delObj.GetComponent<gate>().origins != null)
                    {
                        foreach (GameObject origin in delObj.GetComponent<gate>().origins)
                        {
                            if(origin != null)
                            {
                                int x = getIndex(delObj, origin.GetComponent<power>().circuit);
                                for (; x < origin.GetComponent<power>().circuit.Count;)
                                {
                                    origin.GetComponent<power>().circuit.RemoveAt(x);
```

When I did some testing, I found that when deleting a gate object that has origins it did not remove itself from its origins circuit lists.

My approach to try to fix this was to use the getIndex procedure which I explain below to get the index of the deleted Object and try to remove it by removing at its index position.

```
    }
    if (delObj.GetComponent<gate>() != null)
    {
        if (delObj.GetComponent<gate>().origins != null)
        {
            foreach (GameObject origin in delObj.GetComponent<gate>().origins)
            {
                int x =getIndex(delObj, origin.GetComponent<power>().circuit);
                Debug.Log(x);
                Debug.Log(origin.GetComponent<power>().circuit.Count);
                for (; x < origin.GetComponent<power>().circuit.Count;)
                {
                    origin.GetComponent<power>().circuit.RemoveAt(x);
                }

            }
        }
```

This fixed the issue.

```
    if (delay > 0)
    {
        delay -= Time.deltaTime;
    }
    else
    {
        if (delay != -0.1f)
        {
            foreach (generate gen in generates)
            {
                gen.genUpdate();

            }
            delay = -0.1f;
        }
    }


int getIndex(GameObject obj, List<GameObject> list)
{
    int i = 0;
    foreach (GameObject lObj in list)
    {
        if (lObj == obj)
        {
            return i;
        }
        i++;
    }
    return 99;
```

The code above goes through each object that generates a laser (stored in gens) and calls the genUpdate procedure for that object. My first approach was to do just a foreach without using a delay before the foreach is called, but this caused issues since when I placed an object, the void Start in the generate script attached to the object I am placing down, this would call the genUpdate procedure at the same time as the sandbox script did which caused some more issues that I did not know how to solve, to fix this I added a delay before the foreach.

The purpose of the getIndex procedure is to return the index position of an object, if the object is not within the list, it'll return 99. It does this by iterating through the passed in list and comparing it

to the object that the code wants to find. This is the obj parameter that has been passed through. While iterating it compares the obj variable to the lObj and if it is the same it returns 'i', if not it increments 'i' by 1 and continues until it reaches the end of the list. Then it will return 99 since nothing has been found.

```
        }
    }
    if(Input.GetMouseButtonDown(0)|| Input.GetMouseButtonDown(1))
    {
        delay = 0.01f;
        foreach(generate gen in generates)
        {
            if (gen.gameObject.GetComponent<gate>() != null)
            {
                gen.gameObject.GetComponent<gate>().inputs.Clear();
                gen.gameObject.GetComponent<gate>().origins.Clear();
            }
        }

    }
}

if(delSegs != null)
```

The code again checks for an input by the left or right mouse button if it does happen then delay is set to 0.01 which triggers the code that I have just talked about. This then does the same thing by going through each object that can generate a laser and clearing their inputs and origins so that they can be filled back in when re-generating. This was originally inside each of separate mouse button if statements, but I eventually changed my approach to be this because I needed the delay to update the circuit so that it would not cause errors

```
if (delay > 0)
{
    delay -= Time.deltaTime;
}
else
{
    if (delay != -0.1f)
    {
        foreach (power pow in winCheck.powers)
        {
            pow.gameObject.GetComponent<generate>().genUpdate();
        }
        foreach (GameObject obj in updateObjs)
        {
            obj.GetComponent<generate>().genUpdate();
        }
        delay = -0.1f;
    }
}
```

I then had to update the delay process because it did not check gates that had pre generated lasers such as the not gate. To fix this I created a new list of game Objects called updateObjs in which I

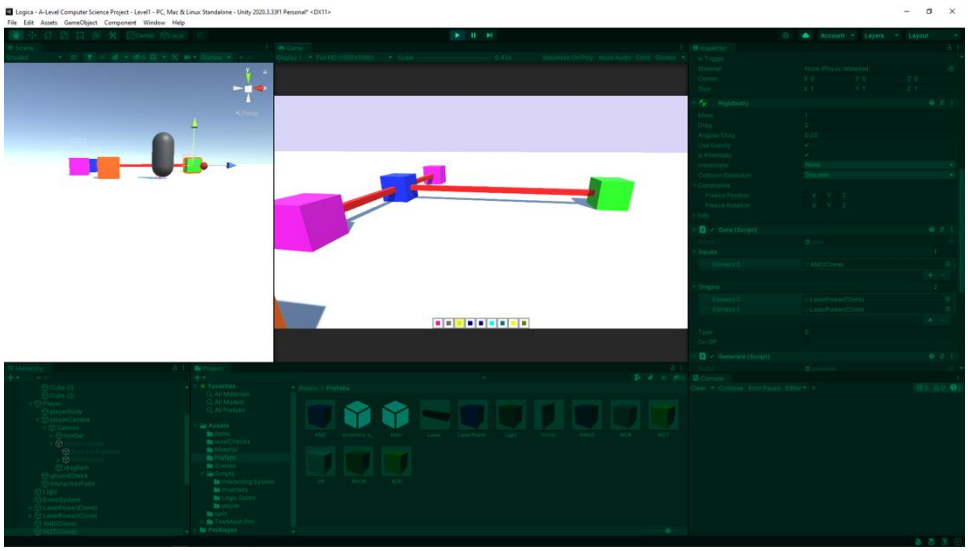
added all not gates that were placed into the list. I also removed the deleted object from the list when a not gate got broken.

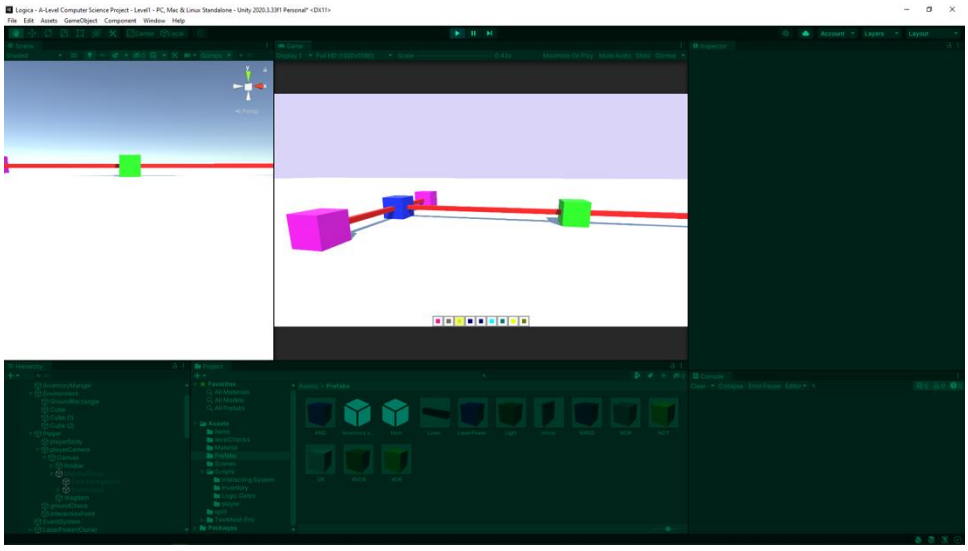
After further consultation, my client is content with the functionality of the placing and breaking blocks.

Updating Circuits

Next, I tried to make the game more immersive and enjoyable by updating the circuits so that the student will not have to turn off the entire circuit and turn the switches back on.

Below are some screenshots showing that the circuits are not updating when a block is added.





I tried to create a circuit list, anticipating using this list in the future when checking for the order of the gates.

In the sandbox code I also tried to make it so that the game regenerates the lasers every time a new object is placed or broken hence updating the lists that need updating such as circuit and the circuit visually in-game.

The circuits did not add my gates correctly and kept updating incorrectly whenever I placed a block When I Debugged the code below it kept giving me the correct index, but it did not remove it from the circuit at all.

```
Unity Message | 0 references
void Update()
{
    if (GetComponent<power>() != null)
    {
        onOff = GetComponent<power>().onOff;
        onGate = false;
    }
    if (GetComponent<gate>() != null)
    {
        onOff = GetComponent<gate>().onOff;
        onGate = true;
    }


    if (coords.Count != segs.Count)
    {
        if (coords.Count > segs.Count)
        {
            for (int i = segs.Count; i < coords.Count; i++)
            {

                Vector3 rot = new Vector3(0, 0, 0);
                if (dirs[i].x != 0)
                {
                    rot.y = 90;
                }
                GameObject placed = Instantiate(laserSeg, coords[i], Quaternion.Euler(rot));
                placed.name = "laser";
                segs.Add(placed);
            }
        }
        if (coords.Count < segs.Count)
        {
            Destroy(segs.Last());
            segs.RemoveAt(segs.Count - 1);
        }
    }

    if (onOff != oldOnOff)
    {

        genUpdate();
        oldOnOff = onOff;
    }
}
```

To fix this I started off by changing the generate code. I added the top two if statements initialise onGate and onOff variables to their correct state depending on if it is a switch or a gate.

The if statement below checks that the visual segments in are in the correct spot when updating.

To do this it first checks if the length of coords (coordinates) which is a list of coordinates where the segments should be placed comparing it to the current positions of the lasers currently generated from the game object. I then check if coords length is more than the length of the segments list, after I iterate through the segments list and I create a new local variable inside the for loop called rot which will be used to reset the rotation. After this, I then check the direction that the laser should be facing based on the rotation of its source block and make sure that this is not equal to 0. I do this by accessing the dirs list which is a list of directions that the segments should be facing, and then I change the rotation of the segment's y axis to 90. I do this because in the other case this is already 0 hence, I need to change the rotation to 90 to connect to the source block correctly.

The code then instantiates the laser segments using the coords list for the coordinates as to where they should be placed and then the rotation of the coordinates to make sure they connect neatly.

I then add the segments to the segs list. I also check the other case which is if coords is less than segs in which case it will delete the last segments until coords and segs are equal so that the condition is not being satisfied anymore.

After, I check if onOff is different to the oldOnOff and then I call genUpdate. At the end of this procedure, I change onOff to the oldOnOff. Whenever a gate or switch is turned on or off it generates the laser again so that it can delete the correct laser segments and other gates have the correct inputs and origins.

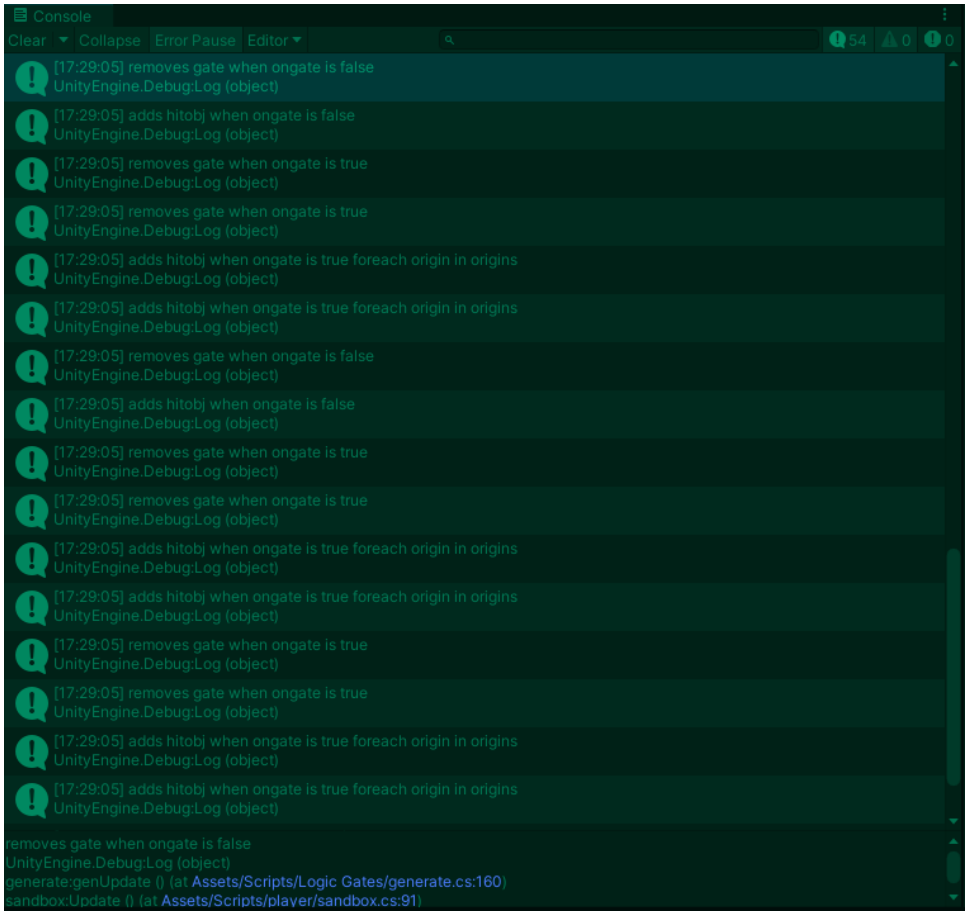
```
            }
        }
        gate = null;
    }

    if (onOff)
    {
        dir = genDir;
        bounced = false;

        pos = transform.position;


        for (genDis = 1; genDis < 25; genDis++)
        {
            if (!bounced)
            {
                pos = transform.position;
            }
            genPos = pos + new Vector3(dir.x * genDis, 0, dir.z * genDis);
            gen = true;
            checkAtPos();
            if (gen)
            {
                coords.Add(new Vector3Int(Mathf.RoundToInt(genPos.x), Mathf.RoundToInt(genPos.y), Mathf.RoundToInt(genPos.z)));
                dirs.Add(dir);
            }
        }
    }

}
```

I also separated the genAtPos procedure into checkAtPos and genUpdate. This is because it adds more modularity into the code to improve maintenance and debugging of the code to find errors.

However, I ran into a logic error.

Whenever an object was placed the light object would move up by two, light is supposed to be at the bottom of the circuit list since it is hit last.

I debugged a bunch of messages in my code to trace the values and see what is happening



The logic error was whenever I placed a block to update the lasers by quickly turning them on and off, I found out that it added the objects by the order that they were placed rather than the order that they were hit by the laser. So, it would not be in the correct order.

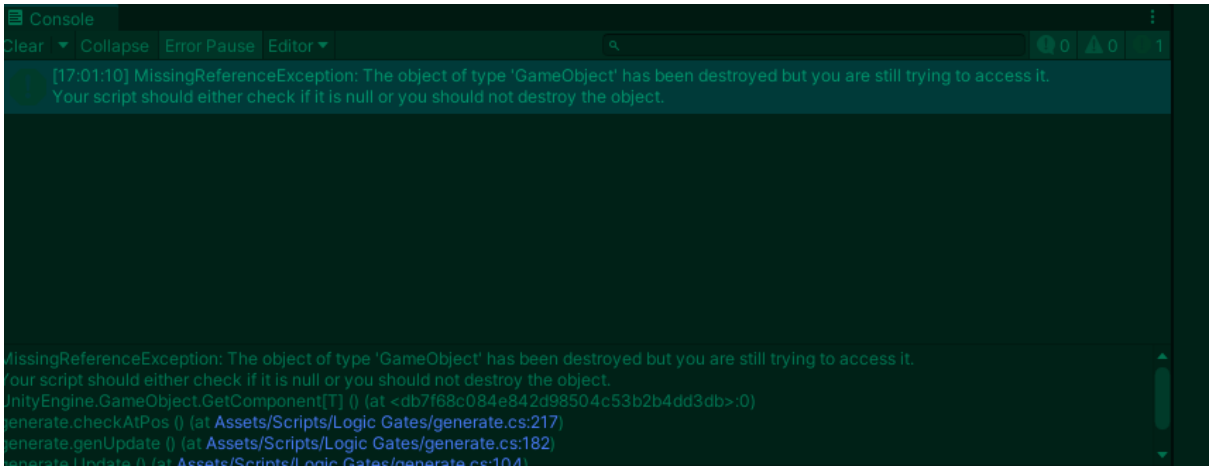
Within checkAtPos I added a line of code to make sure that the object that has been hit via the raycast / laser will also start to generate, this fixed it by instead of generating all the lasers in a circuit at the same time, it works like a chain reaction where the lasers get generated in the order of the circuit

The code then also has an else statement which is a way to stop lasers from endlessly generating. This is primarily implemented to save processing power.

```
            hitObj.GetComponent<generate>().genUpdate();
        }
        else
        {
            genDis = 25;
            gen = false;
        }
    }
}
```

After I made these changes, I got an error that I am trying to access a destroyed object

I tried debugging to see where the error occurred however it still gave me the error before the debug appeared.



My first approach was to check if circuit was null for this error

```
if (origin.GetComponent<power>().circuit != null)
{
    if (!inList(hitObj, origin.GetComponent<power>().circuit))
    {
        origin.GetComponent<power>().circuit.Add(hitObj);
    }
```

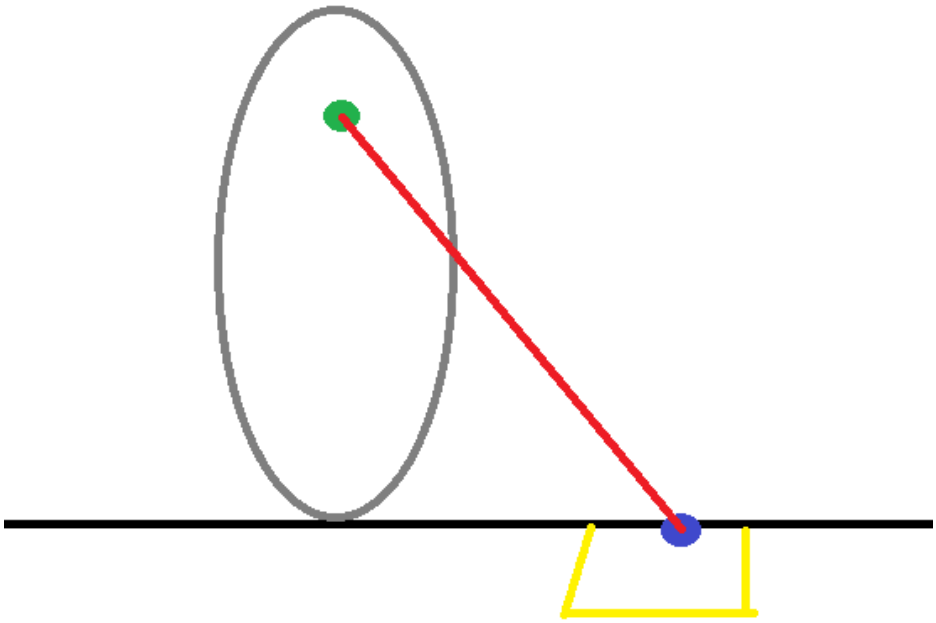
I then checked to see if the actual origin was null, and this solved the error it was giving me

```
if(origin != null)
```

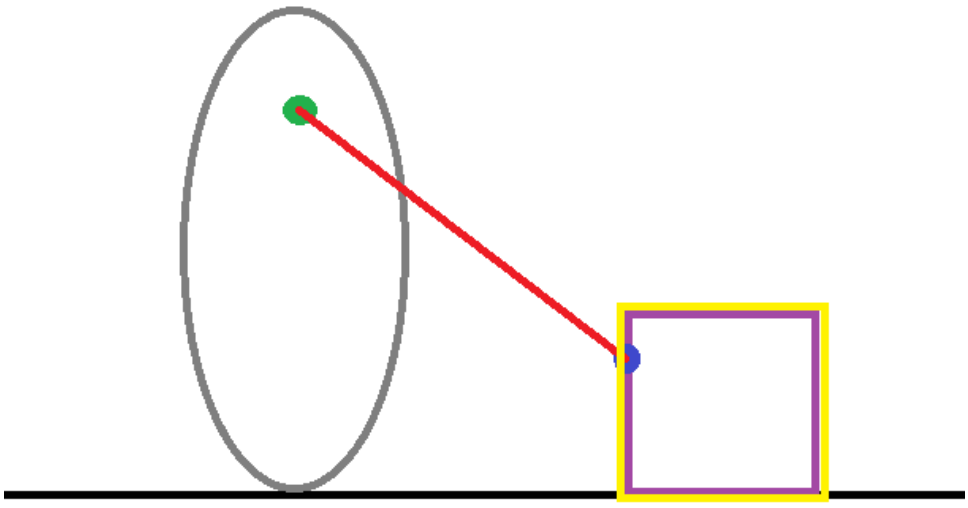
After further consultation, my client is content with the functionality of the generating and updating lasers.

Object Outline

The outline of objects will use a similar way of instantiating the outline object but instead I will have to instantiate it on the floor. I will be able to do this by again rounding the hit point but making sure that is in the floor and not above



I will also have to instantiate the outline object inside the block that the raycast hits. This will use the object that it hits position to instantiate the outline object.



This will allow the user to better see what they are looking. This is used in Minecraft and is extremely user friendly

I started of the script by initialising the variables

```
Unity Script | 0 references
public class select : MonoBehaviour
{
    [SerializeField] Transform cameraT;
    Vector3 posDistance;
    Vector3 placePos;
    GameObject placed;
   [SerializeField] LayerMask mask;
    public GameObject outlineObj;
    RaycastHit hit;
```

I first started off by initialising the values of the selectObject class

The variable called cameraT is the transform of the camera. This is so that I can access the position and rotation more efficiently than passing it as a GameObject.

The Vector3 called posDistance is the position of the hit object.

The Vector3 called placePos is the position where the outline will be instantiated in that position

The GameObject placed is the instantiated outline.

The mask is used within the raycast function that is being used

OutlineObj is the prefab that is being instantiated and raycast hit is where the raycast hits

```
Unity Message | 0 references
private void Update()
{
    if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask))
    {
        posDistance = hit.collider.transform.position;
        placePos = new Vector3(Mathf.RoundToInt(posDistance.x), Mathf.RoundToInt(posDistance.y), Mathf.RoundToInt(posDistance.z));
        if (placePos.y == 0)
        {
            GameObject placed = Instantiate(outlineObj, placePos, Quaternion.identity);
        }
    }
    else
    {
        if(placed != null)
        {
            GameObject.Destroy(placed);
        }
    }
}
```

I first started off by coding the instantiation of the object by using a raycast making sure I pass in 5 so it has the same distance has been able to place and break blocks for convenience and user friendly – ness.

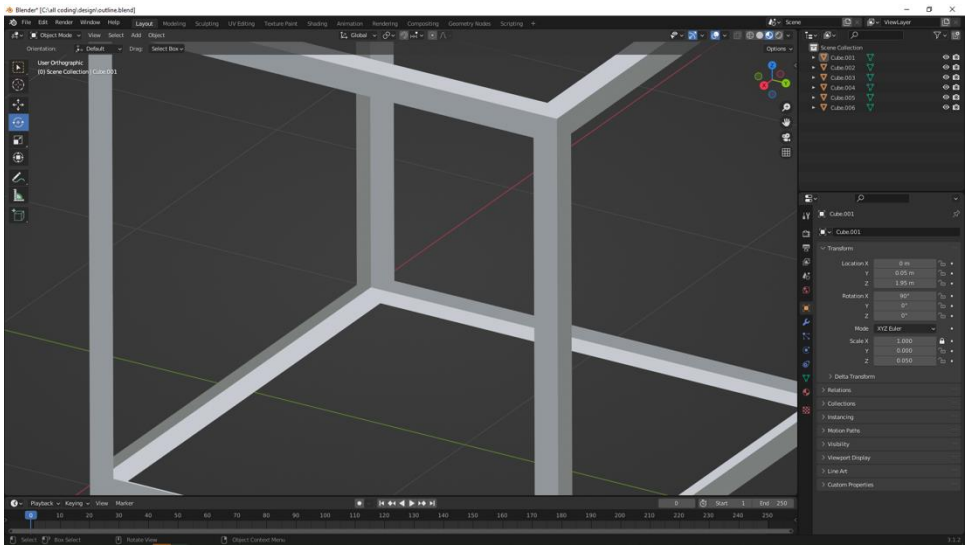
I then used posDistance to find the position of the game object that has been hit and then round it and then set that to placePos

After I checked if the y axis of the position to be placed is equal to 0.

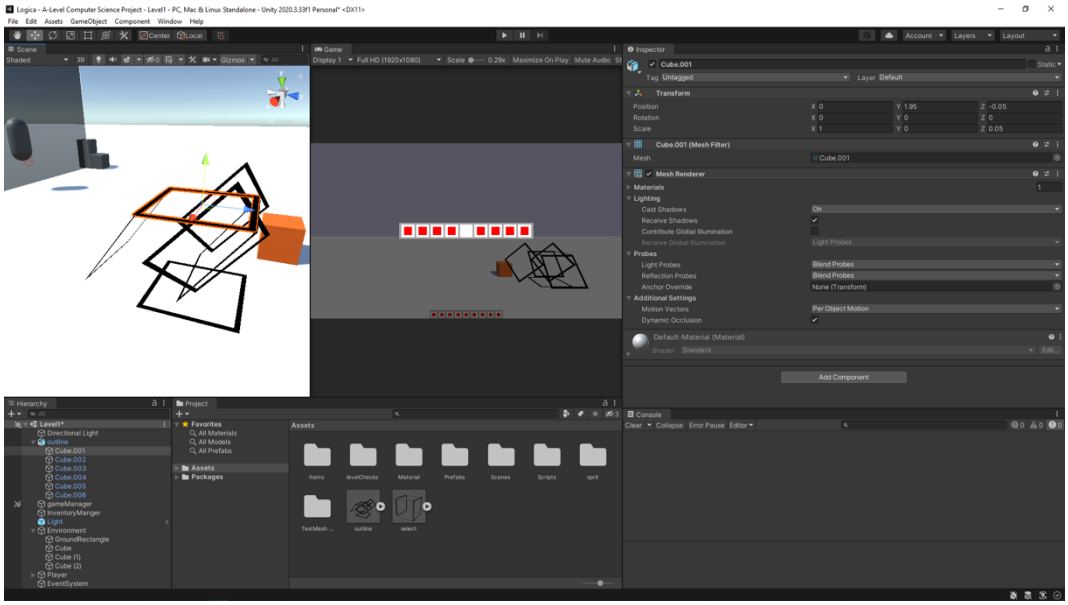
If so, then it can be instantiated

However, if the player is looking somewhere else and it does not hit an object e.g., the sky or an object is more than 5 tiles away it will delete the placed object.

I then started to work on the model for the outline in blender. Since the object had to be bigger than the Unity version of 1x1x1 I Had to make sure that it was not small enough so that the outline will be inside the object and clip and had to make sure it was big enough so that it nicely fits around the 1x1x1 block in Unity.

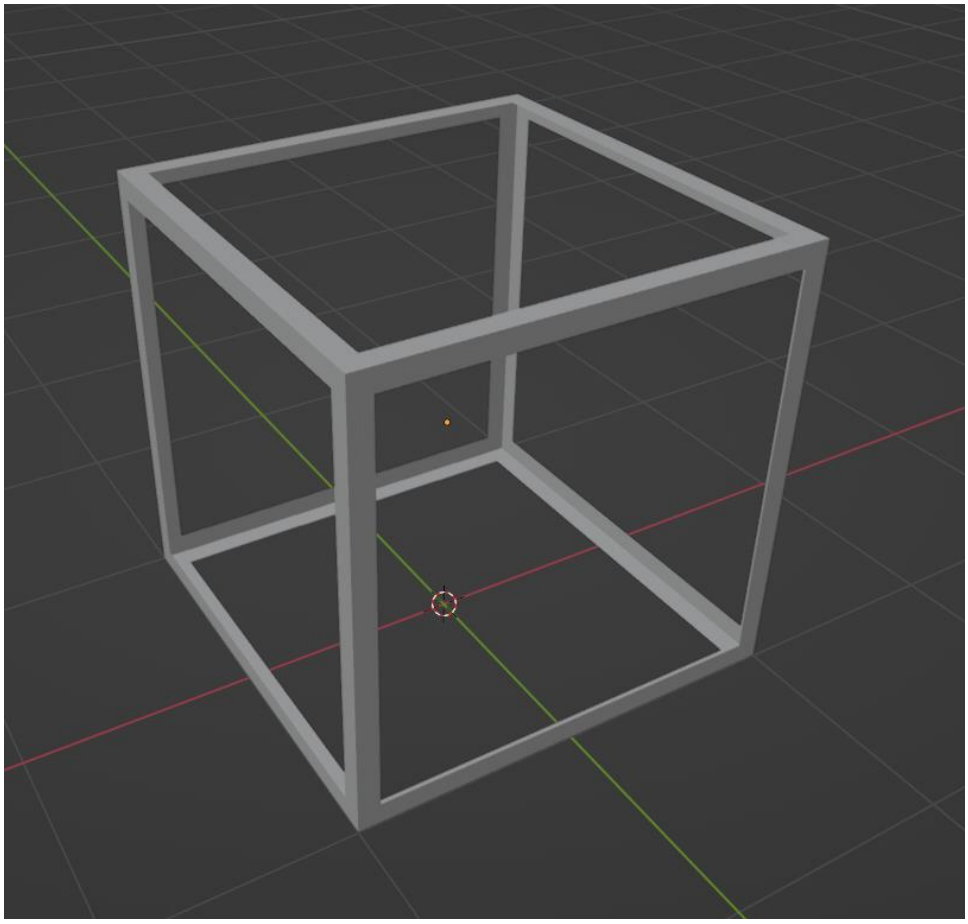


After I created the model, I imported it to Unity and there was a problem.

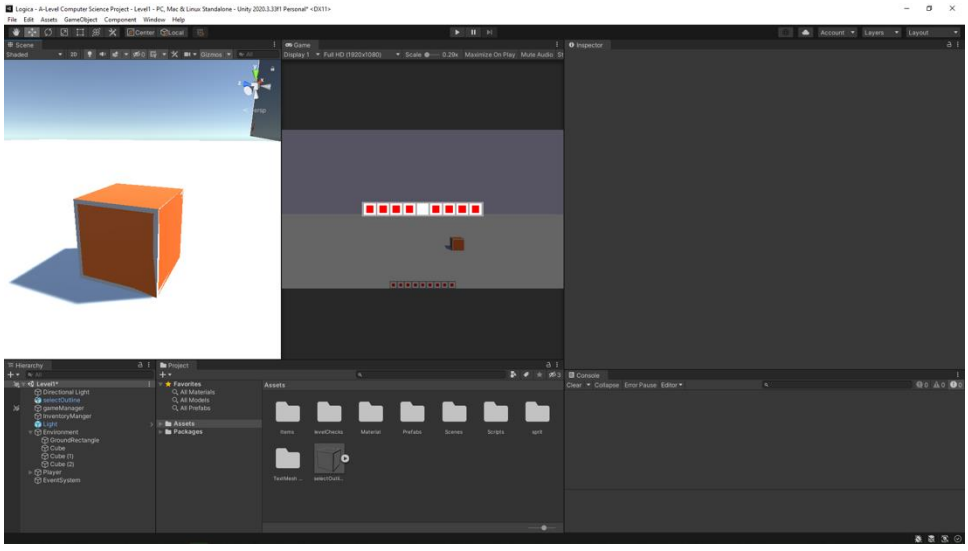


The object did not import correctly due to me using multiple objects and connect them as once which created this problem

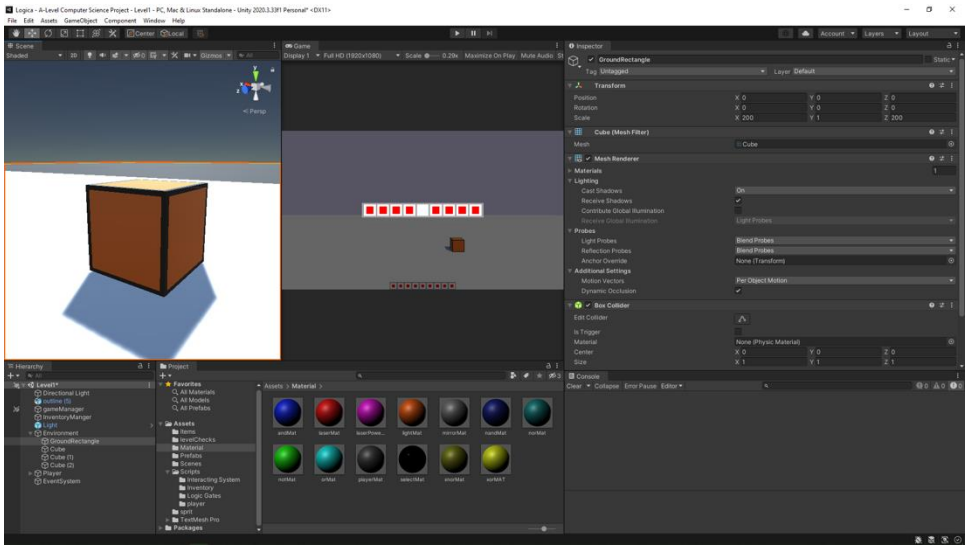
I solved this by creating the outline as one object and not consists of multiple parts and cutting out a single cube using Boolean modifiers



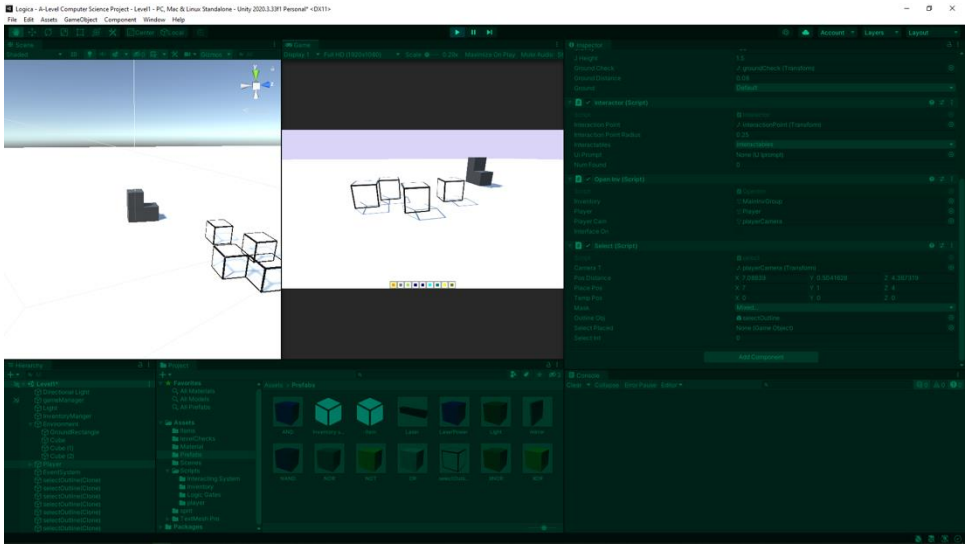
I next imported this to Unity, and it worked as intended however I did have to go into the blender settings to change the rotation and scale of the object and setting the origin to the geometry.



Once imported I created a material that was fully black. This contrasts the white ground and the colourful gates and makes it easy to identify what you are looking at



When I tested this in game, The outline Object spawned multiple times and above the ground.



I decided to fix this by taking some inspiration from my pseudocode and implement the selectInt to keep track of where the player is looking and also adding an extra position called tempPos to act to keep track of where the player last looked previously and check if they have changed where they have looked so that I can move the delete and instantiate a new outline Object. I first check if the object that was hit by the raycast is not null in case it has been deleted by the player. I then check if placePos is not null so that tempPos is only set when placePos has been declared. Then I set tempPos to placePos so now I store a temporary variable to check for the previous position of the outline object.

```
Unity Message | 0 references
private void Update()
{
    RaycastHit hit;
    if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask)) {

        posDistance = hit.point - cameraT.forward * 0.01f;
        if (placePos != null)
        {
            Vector3 tempPos = placePos;
        }
        placePos = new Vector3(Mathf.RoundToInt(posDistance.x), Mathf.RoundToInt(posDistance.y), Mathf.RoundToInt(posDistance.z));
        if (placePos.y == 1 && selectInt == 0)
        {
            GameObject selectPlaced = Instantiate(outlineObj, placePos, Quaternion.identity);
            selectInt = 1;
        }
    }
    else
    {
        if(selectInt == 1 || tempPos != placePos)
        {
            Destroy(selectPlaced);
            selectInt = 0;
        }
    }
}
}
```

However, the destroy part did not work as intended I then added more if statements to branch the code more for different conditions that could happen

The code now uses an else if statement instead of another If since if it satisfies both conditions it will do both. This will reduce further errors. I next checking to make sure hitObj is not null and that the collider of the object's position from where I am instantiating is also at y level 1 to reduce any possible errors in the future. I then instantiate the object and I reset hitObj to null and assign 1 to selectInt meaning the outline object is present

```
Unity Message | 0 references
private void Update()
{
    RaycastHit hit;
    if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask)) {
        posDistance = hit.point - cameraT.forward * 0.01f;
        if (placePos != null)
        {
            Vector3 tempPos = placePos;
        }
        placePos = new Vector3(Mathf.RoundToInt(posDistance.x), Mathf.RoundToInt(posDistance.y), Mathf.RoundToInt(posDistance.z));
        if (placePos.y == 0 && selectInt == 0)
        {
            GameObject selectPlaced = Instantiate(outlineObj, placePos, Quaternion.identity);
            selectInt = 1;
        }
        else if (placePos.y == 1 && selectInt == 0)
        {
            hitObj = hit.collider.gameObject;
            if (hitObj != null)
            {
                if (hit.collider.transform.position.y == 1)
                {
                    GameObject selectPlaced = Instantiate(outlineObj, hit.collider.transform.position, Quaternion.identity);
                    selectInt = 1;
                    hitObj = null;
                }
            }
        }
        else
        {
            if (selectInt == 1 || tempPos != placePos)
            {
                Destroy(selectPlaced);
                selectInt = 0;
            }
        }
    }
}
}
```

71 % No issues found

However, this still did not fix my code, but it did make it more efficient and have less risks of errors crashing the game.

I assumed that the reason it did not delete still is that the condition under when to delete it was wrong, so I changed the if statement to be an AND statement and not an OR statement.

Next, I made tempPos to be initialised outside of any procedures. This is because I will be using in the else statement tempObj does not exist yet, so it gave me an error.

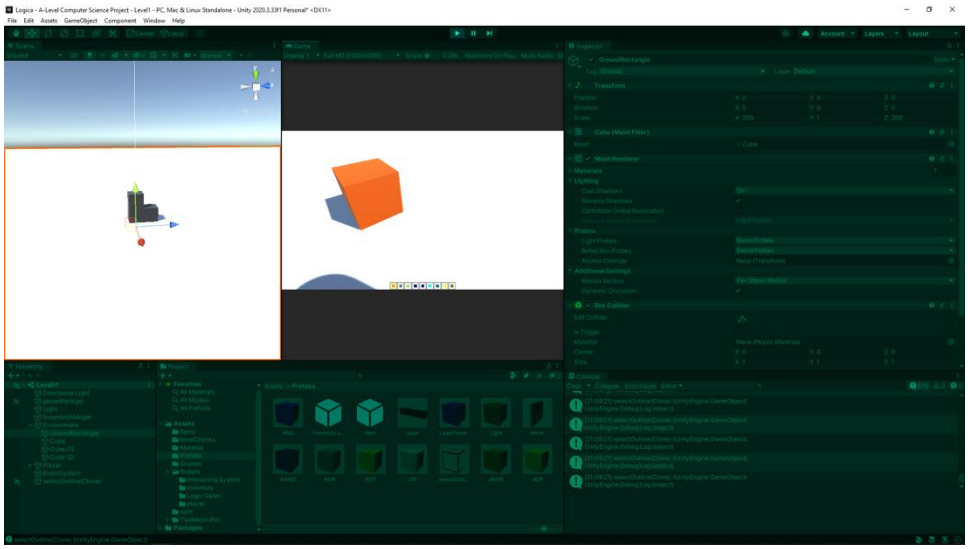
```
Unity Message | 0 references
private void Update()
{
    RaycastHit hit;
    if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask)) {
        posDistance = hit.point - cameraT.forward * 0.01f;
        if (placePos != null)
        {
            tempPos = placePos;
        }
        placePos = new Vector3(Mathf.RoundToInt(posDistance.x), Mathf.RoundToInt(posDistance.y), Mathf.RoundToInt(posDistance.z));
        if (Mathf.RoundToInt(posDistance.y) == 0 && selectInt == 0)
        {
            GameObject selectPlaced = Instantiate(outlineObj, placePos, Quaternion.identity);
            selectInt = 1;
        }
        else if (placePos.y == 1 && selectInt == 0)
        {
            hitObj = hit.collider.gameObject;
            if (hitObj != null)
            {
                if (hit.collider.transform.position.y == 1)
                {
                    selectPlaced = Instantiate(outlineObj, hit.collider.transform.position, Quaternion.identity);

                    selectInt = 1;
                }
                if(tempPos != placePos)
                {
                    hitObj = null;
                }
            }
        }
        else
        {
            Debug.Log(selectPlaced);
            if (selectInt == 1 && tempPos != placePos)
            {
                Destroy(selectPlaced);
                selectInt = 0;
            }
        }
    }
}
```

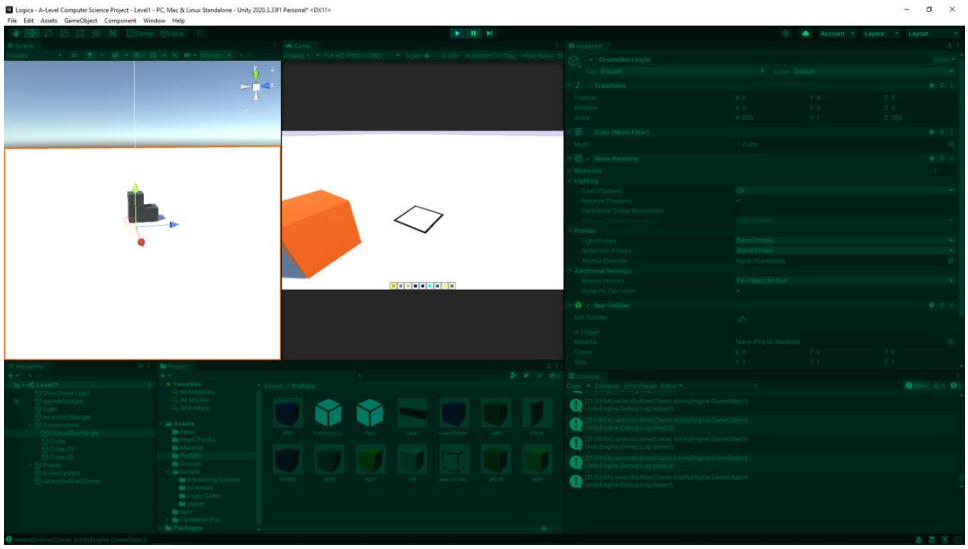
64 % No issues found

However, the code still did not work as they outline object kept flickering in the ground (deleting itself and re instantiating) despite me trying to fix it by rounding it in the if statement.

This was due to the place Pos rounding to 1 which meant it had to be deleted and the hitObj was defined only when you are placing something at the Y level 1 which caused more inefficient code

Below is showing the outlineObj flickering when testing



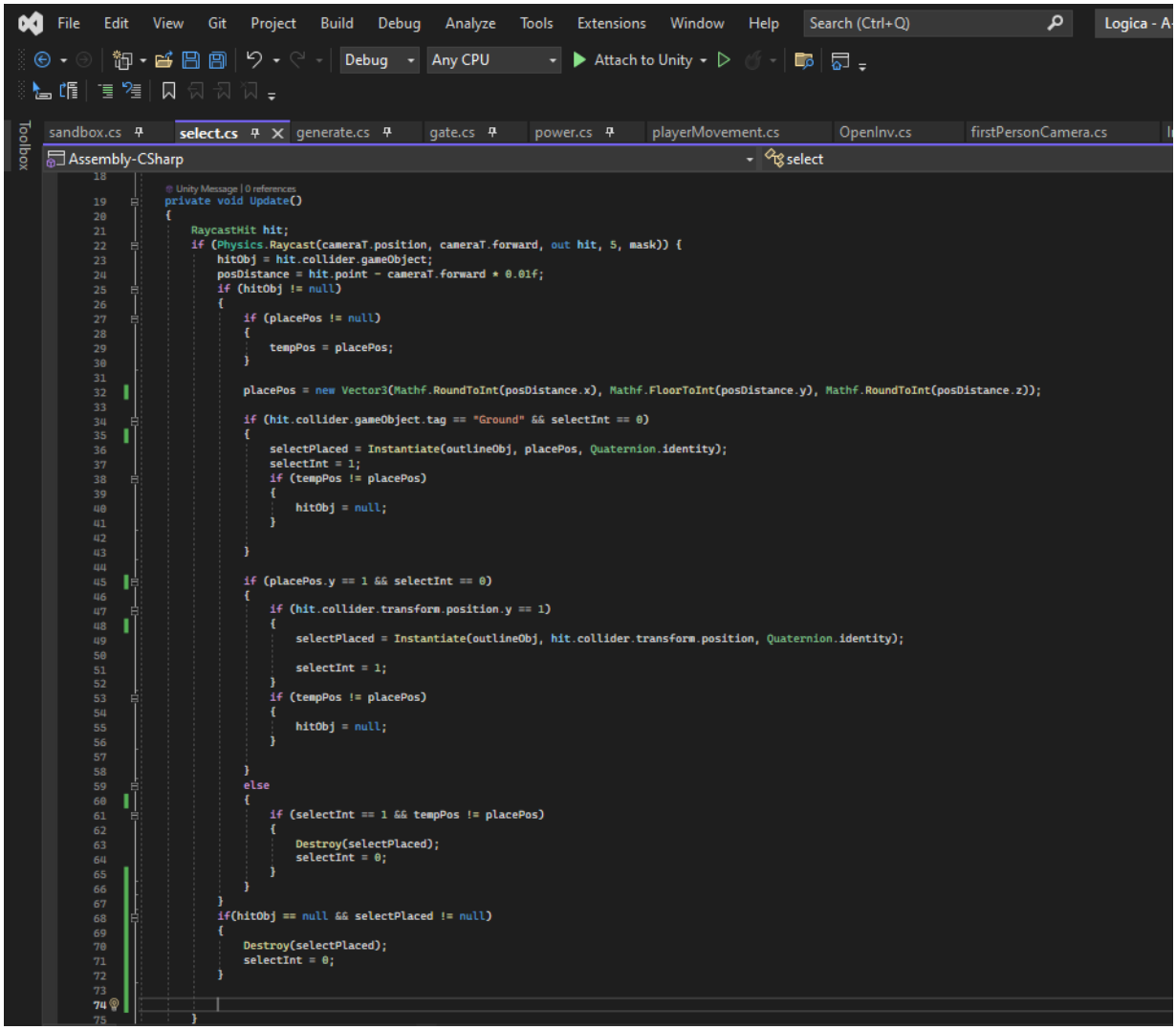


To fix this I floored the y axis of placePos instead of rounding it. This made the outline objects always be 0 or 1.

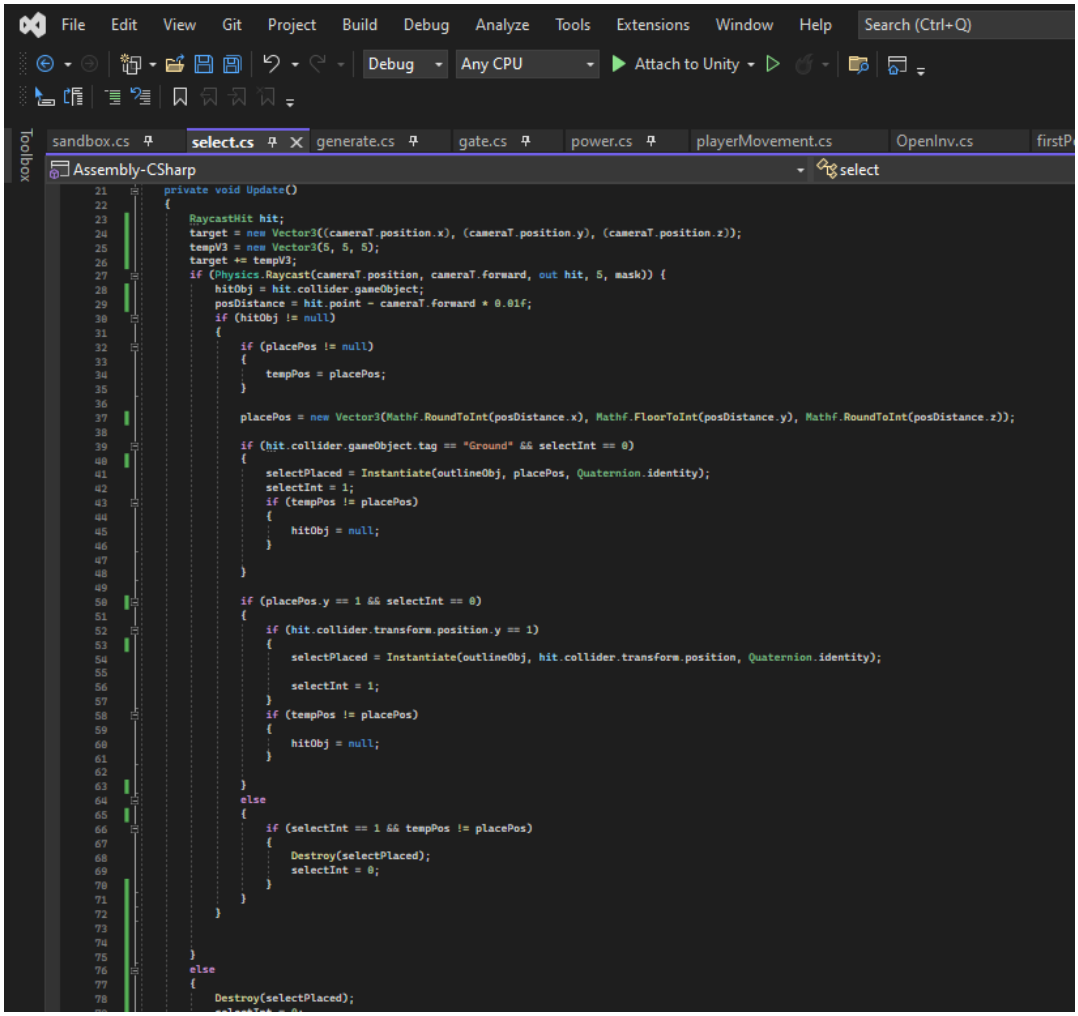
```
31
32          placePos = new Vector3(Mathf.RoundToInt(posDistance.x), Mathf.RoundToInt(posDistance.y), Mathf.RoundToInt(posDistance.z));
33
34          if (hit.collider.gameObject.tag == "Ground" && selectInt == 0)
35          {
36              placePos.y = 0;
37              selectPlaced = Instantiate(outlineObj, placePos, Quaternion.identity);
38              selectInt = 1;
39              if (tempPos != placePos)
40              {
41                  hitObj = null;
42              }
43
44          }
45
46          else if (placePos.y == 1 && selectInt == 0)
47          {
48              if (hit.collider.transform.position.y == 1)
49              {
50                  selectPlaced = Instantiate(outlineObj, hit.collider.transform.position, Quaternion.identity);
51
52                  selectInt = 1;
53              }
54              if (tempPos != placePos)
55              {
56                  hitObj = null;
57              }
58
59          }
60          else
61          {
62              Debug.Log(selectPlaced);
63              if (selectInt == 1 && tempPos != placePos)
64              {
65                  Destroy(selectPlaced);
66                  selectInt = 0;
67              }
```

However, this still had an error where the outline object will not delete if you are not looking at an object.

My first approach was to test if hitObj was null and if selectPlaced had a value.



However, this did not work and instead my next approach was to do an else statement to the if raycast check since this will only happen if the player is not looking at an object.



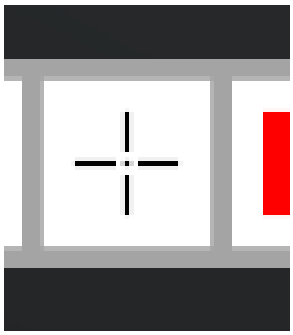
This fixed the issue and outline objects now works as intended.

After further consultation, my client is content with the functionality of the outlining objects.

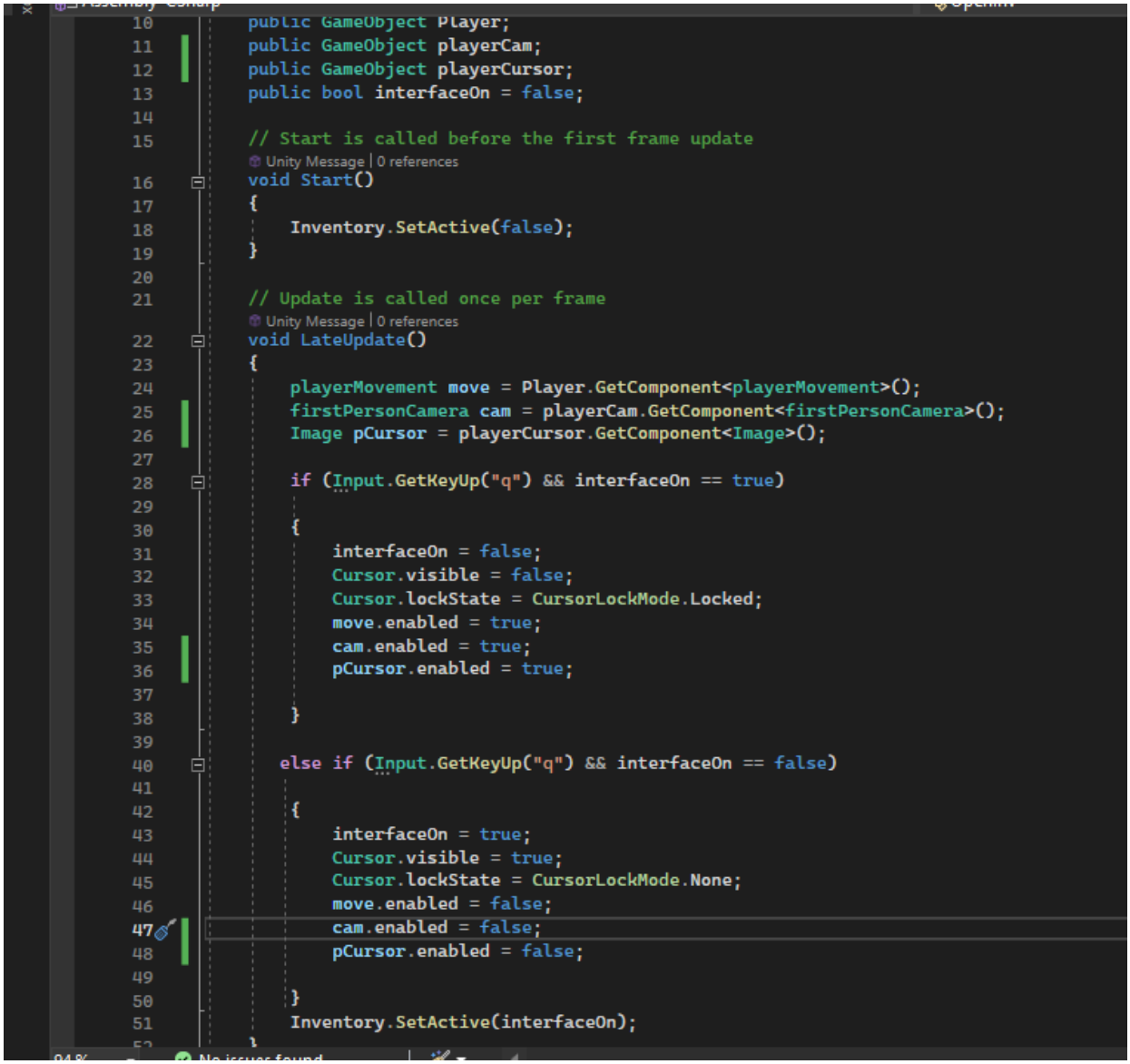
Cursor

I added a cursor to improve user usability and interaction with the game

Since it is at the centre of the screen this makes it easier for the player to see where their player object is looking and at what since it is at the centre of the screen



The black contrasts the most blocks, the sky and the floor.

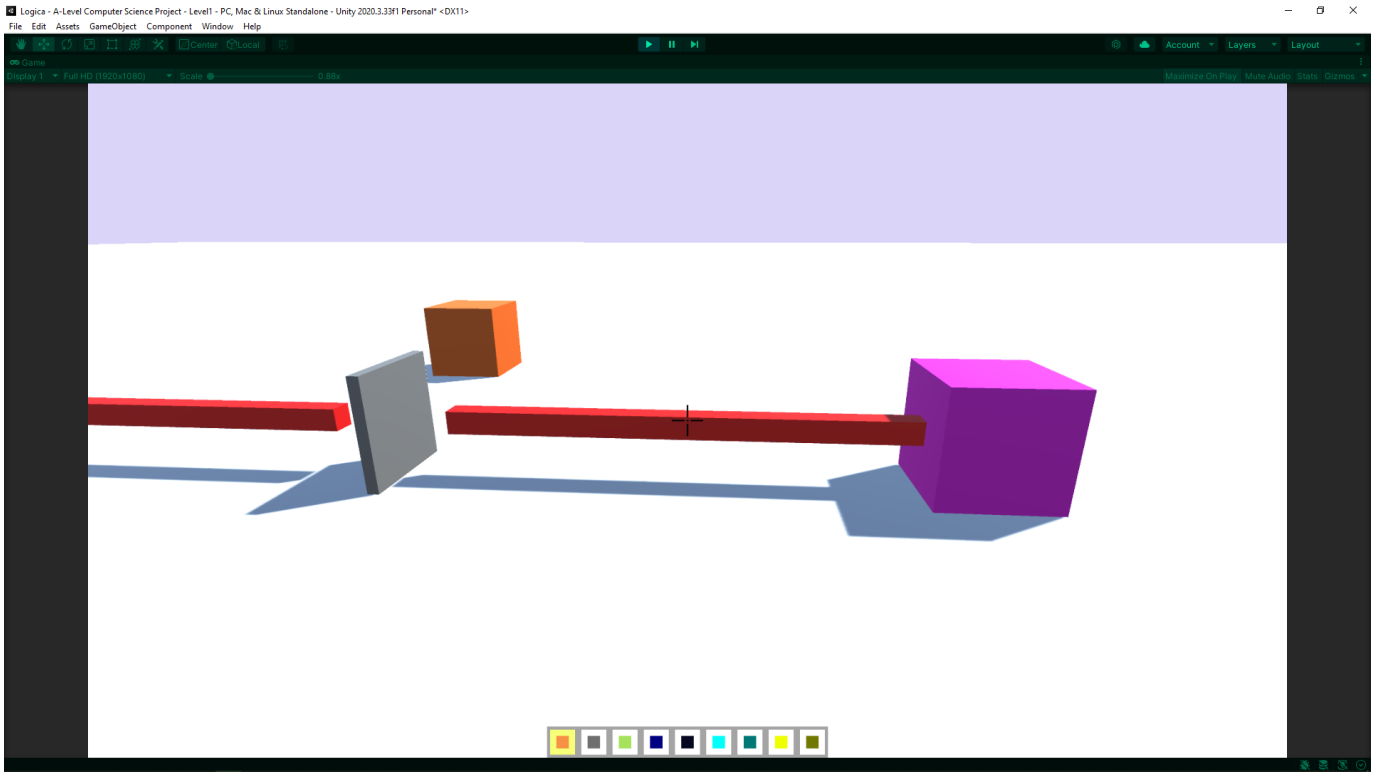


```
public GameObject Player;
public GameObject playerCam;
public GameObject playerCursor;
public bool interfaceOn = false;

// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    Inventory.SetActive(false);
}

// Update is called once per frame
Unity Message | 0 references
void LateUpdate()
{
    playerMovement move = Player.GetComponent<playerMovement>();
    firstPersonCamera cam = playerCam.GetComponent<firstPersonCamera>();
    Image pCursor = playerCursor.GetComponent<Image>();

    if (Input.GetKeyUp("q") && interfaceOn == true)

    {
        interfaceOn = false;
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;
        move.enabled = true;
        cam.enabled = true;
        pCursor.enabled = true;

    }

   else if (Input.GetKeyUp("q") && interfaceOn == false)

    {
        interfaceOn = true;
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
        move.enabled = false;
        cam.enabled = false;
        pCursor.enabled = false;

    }
    Inventory.SetActive(interfaceOn);
}
```

However, when opening the inventory, I had to disable the cursor, so the user inventory screen looks better aesthetically and fits my initial design more.

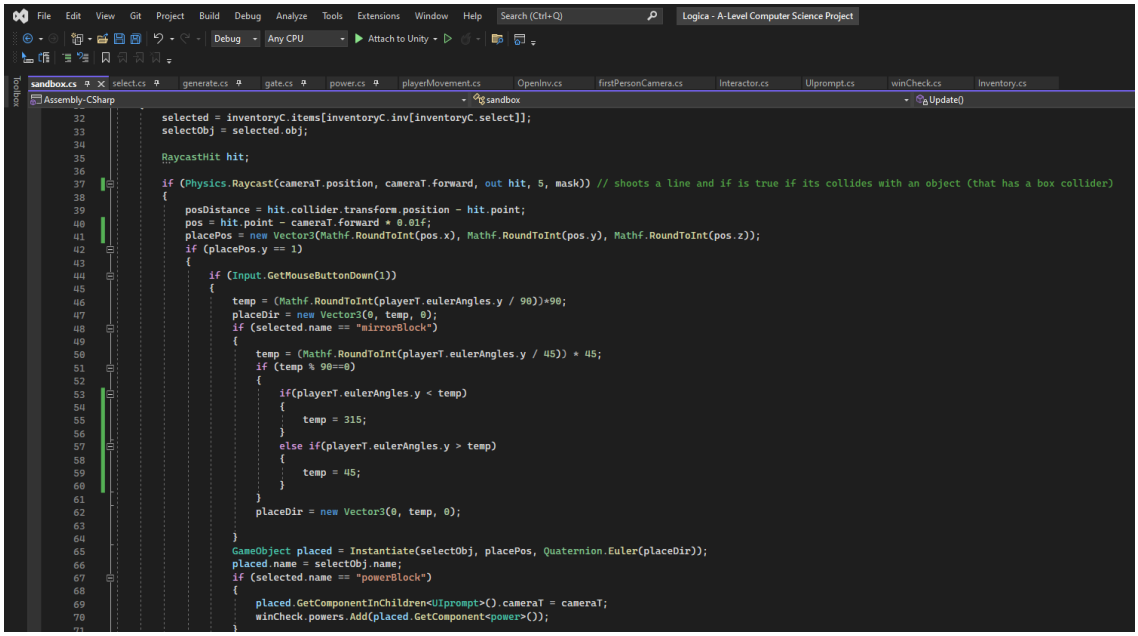
After further consultation, my client is content with the design of the in-game cursor.

Mirrors



A problem with mirrors is that they can be placed straight like this and not at an angle.

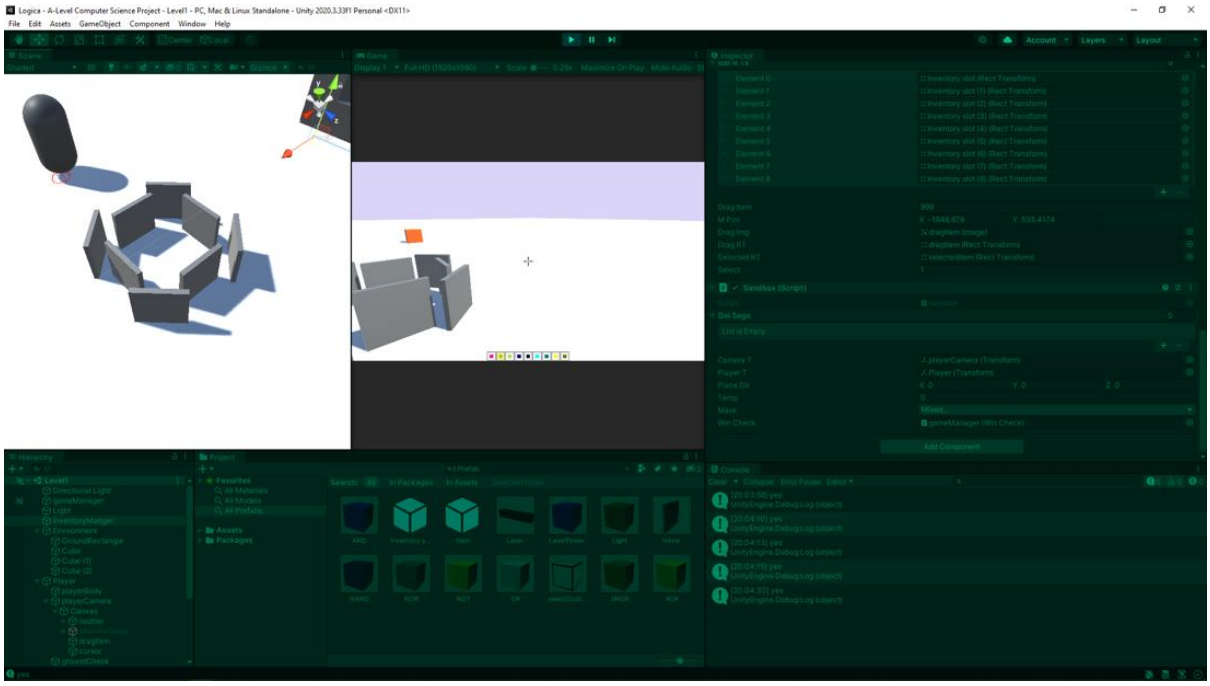
To fix this my first approach was to make a separate section where instead of making temp based on 90 degrees since it's a mirror it's based on 45. This allows me to place the mirrors at the correct angle for the laser generation to work.



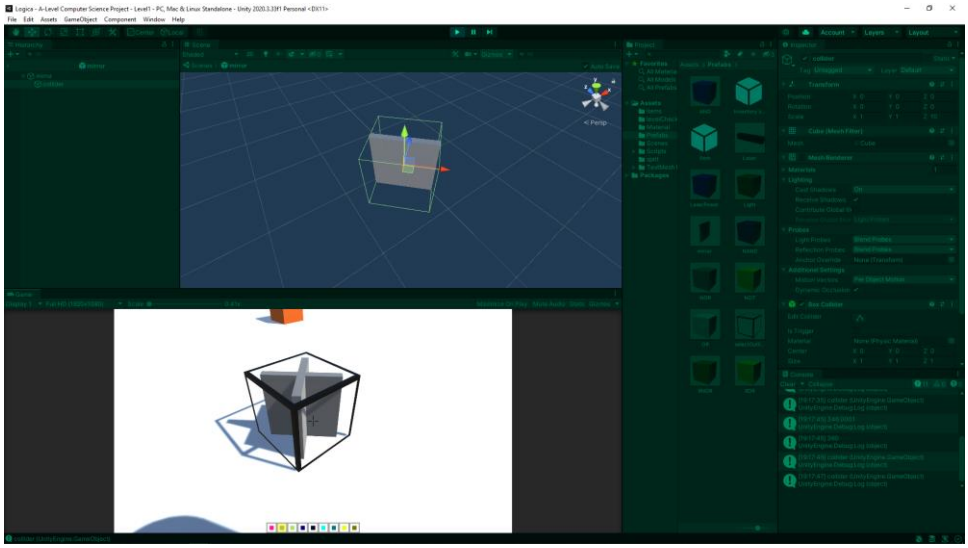
```
selected = inventoryC.items[inventoryC.inv[inventoryC.select]];
selectObj = selected.obj;

RaycastHit hit;

if (Physics.Raycast(cameraT.position, cameraT.forward, out hit, 5, mask)) // shoots a line and if is true if its collides with an object (that has a box collider)
{
    posDistance = hit.collider.transform.position - hit.point;
    pos = hit.point - cameraT.forward * 0.01f;
    placePos = new Vector3(Mathf.RoundToInt(pos.x), Mathf.RoundToInt(pos.y), Mathf.RoundToInt(pos.z));
    if (placePos.y == 1)
    {
        if (Input.GetMouseButtonDown(1))
        {
            temp = (Mathf.RoundToInt(playerT.eulerAngles.y / 90))*90;
            placeDir = new Vector3(0, temp, 0);
            if (selected.name == "mirrorBlock")
            {
                temp = (Mathf.RoundToInt(playerT.eulerAngles.y / 45)) * 45;
                if (temp % 90==0)
                {
                    if(playerT.eulerAngles.y < temp)
                    {
                        temp = 315;
                    }
                    else if(playerT.eulerAngles.y > temp)
                    {
                        temp = 45;
                    }
                }
                placeDir = new Vector3(0, temp, 0);

            }
            GameObject placed = Instantiate(selectObj, placePos, Quaternion.Euler(placeDir));
            placed.name = selectObj.name;
            if (selected.name == "powerBlock")
            {
                placed.GetComponentInChildren<Uiprompt>().cameraT = cameraT;
                winCheck.powers.Add(placed.GetComponent<power>());
            }
```

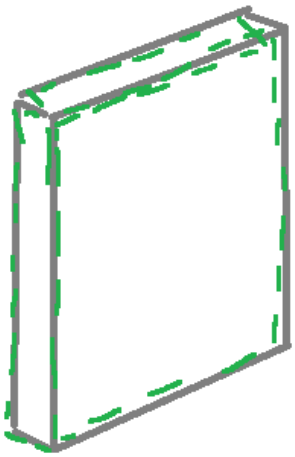
I did this by dividing the player transform by 45 and the rounding it. And then multiplying this value by 45. This effectively rounds the temporary value to the nearest 45. I then check if 90 can go into the number using the MOD function. If my result is 0 that means it will be placed like a straight line

and not at an angle like you see in the image below with the mirrors on the side. This makes sure that all numbers that were rounded to something that can be divided wholly by 90 will be readjusted so that depending on the angle it will round it to either be 45 or 315. Hence making it so you can only place these mirrors at a 45° angle, and they will not be straight on.



Next, I had to solve the issue with placing mirrors inside each other due to their colliders being the size of the mirrors and not the 1x1x1 size like other blocks the implementation of the sandbox script did not work with the mirrors.

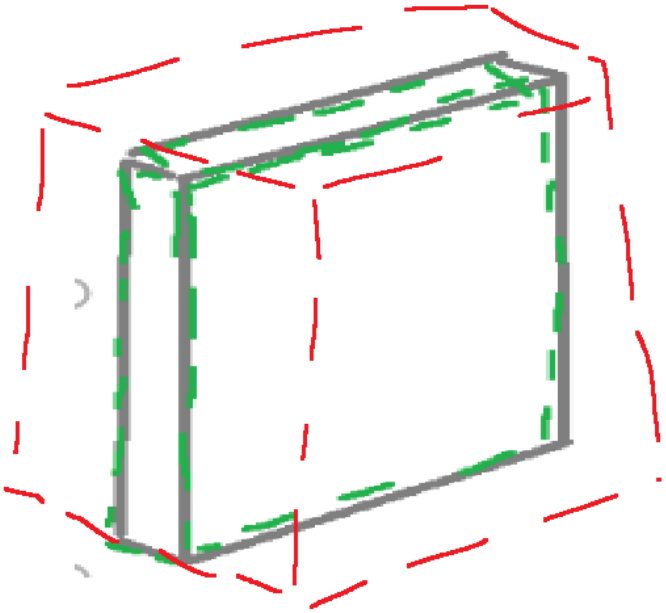




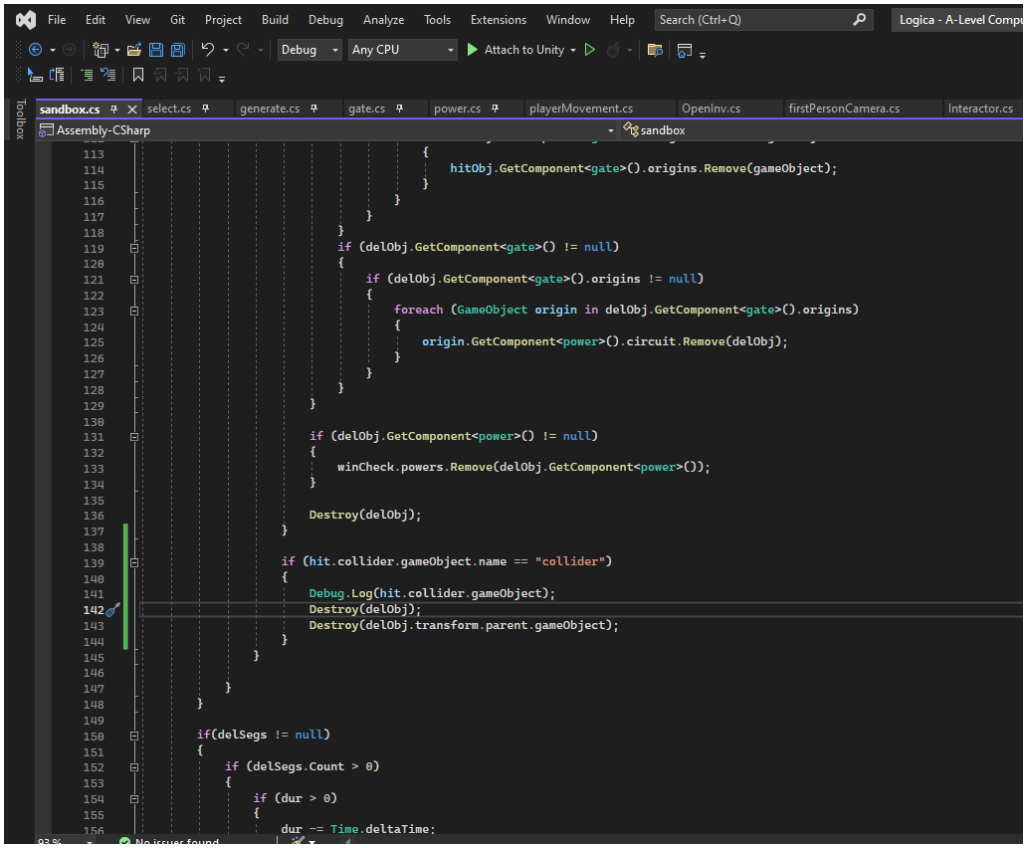
When the collider is attached to the mirror object it morphs to suit the size of the mirror.

This is not how the collider (green dashed line) should be like, I need to treat the mirror object as if it is a 1x1 object.

Below is a rough sketch in the dashed lines of red is what the supposed collider should look like for the object

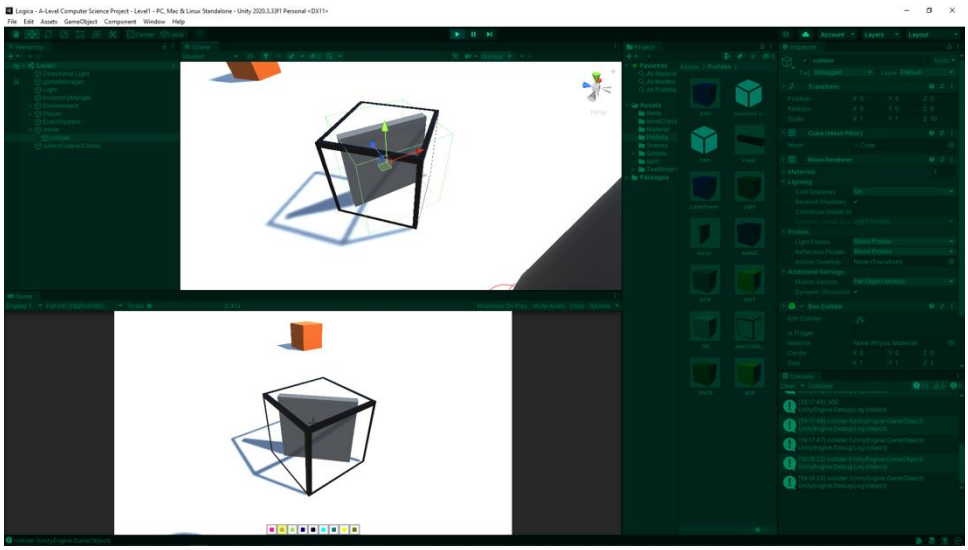


My first approach was to separate the collider from the rest of the block and create a child object for the collider. This collider will not mesh to the mirror object but instead be like a cube, like the rest of the blocks. However, what I did not realise is that when breaking the block, I will break the collider and not the actual block itself since the collider is separated into a separated game object



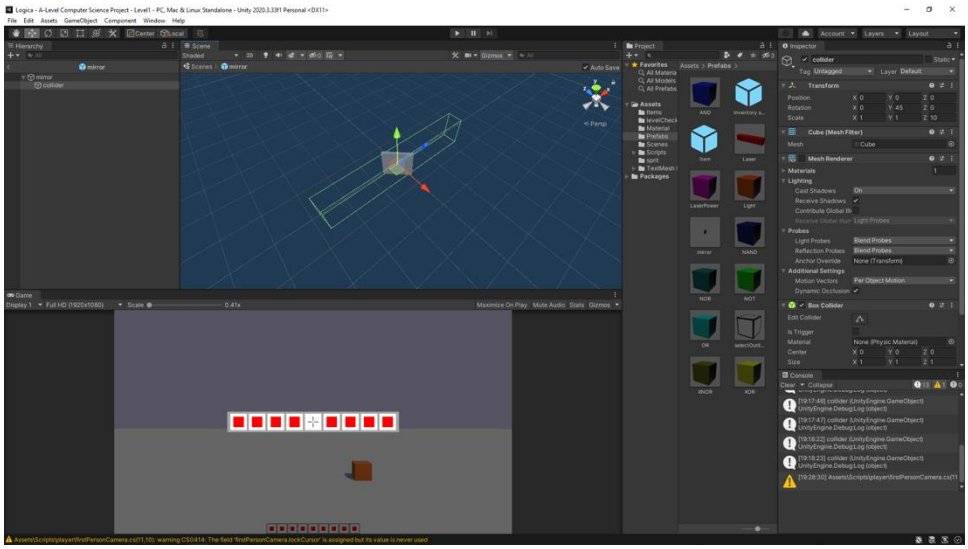
```
                                {
                                    hitObj.GetComponent<gate>().origins.Remove(gameObject);
                                }
                            }
                        }
                    }
                    if (delObj.GetComponent<gate>() != null)
                    {
                        if (delObj.GetComponent<gate>().origins != null)
                        {
                            foreach (GameObject origin in delObj.GetComponent<gate>().origins)
                            {
                                origin.GetComponent<power>().circuit.Remove(delObj);
                            }
                        }
                    }
                }

                if (delObj.GetComponent<power>() != null)
                {
                    winCheck.powers.Remove(delObj.GetComponent<power>());
                }

                Destroy(delObj);
            }

            if (hit.collider.gameObject.name == "collider")
            {
                Debug.Log(hit.collider.gameObject);
                Destroy(delObj);
                Destroy(delObj.transform.parent.gameObject);
            }
        }

    }
}

if(delSegs != null)
{
    if (delSegs.Count > 0)
    {
        if (dur > 0)
        {
            dur -= Time.deltaTime;
```

To fix this I made an if statement to check if the name of the collider is collider and if so destroy itself and its parent, hence destroying the collider and the actual mirror object

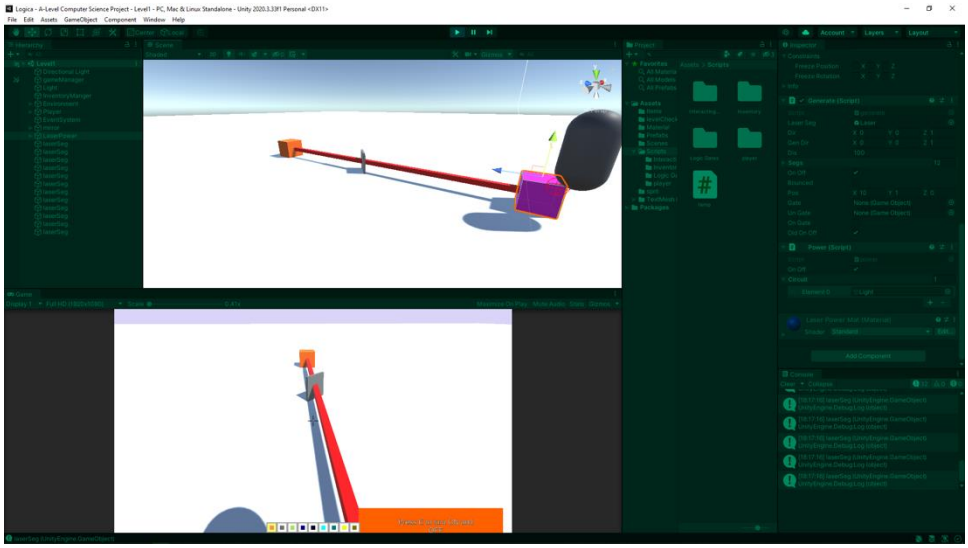


After I fixed this, I still ran into an issue of being able to place mirrors within themselves due to the collider rotating along with the block despite the collider needing to not rotate so the mirror can act like a block and not a smaller block.

My first approach was to rotate the collider itself so that when placed it will counteract the mirrors rotation however the mirror cannot have negative rotation and can only go between 0 and 360 so this does not work. I tried to make the rotation 45° but this messed up the collider even more and was nowhere close to what I wanted.



However, when trying to fix this problem I ran into another problem that meant lasers cannot bounce off mirrors from the back of it. The laser would only rotate if the laser was going in the direction as you were looking at it when placing the mirror down.



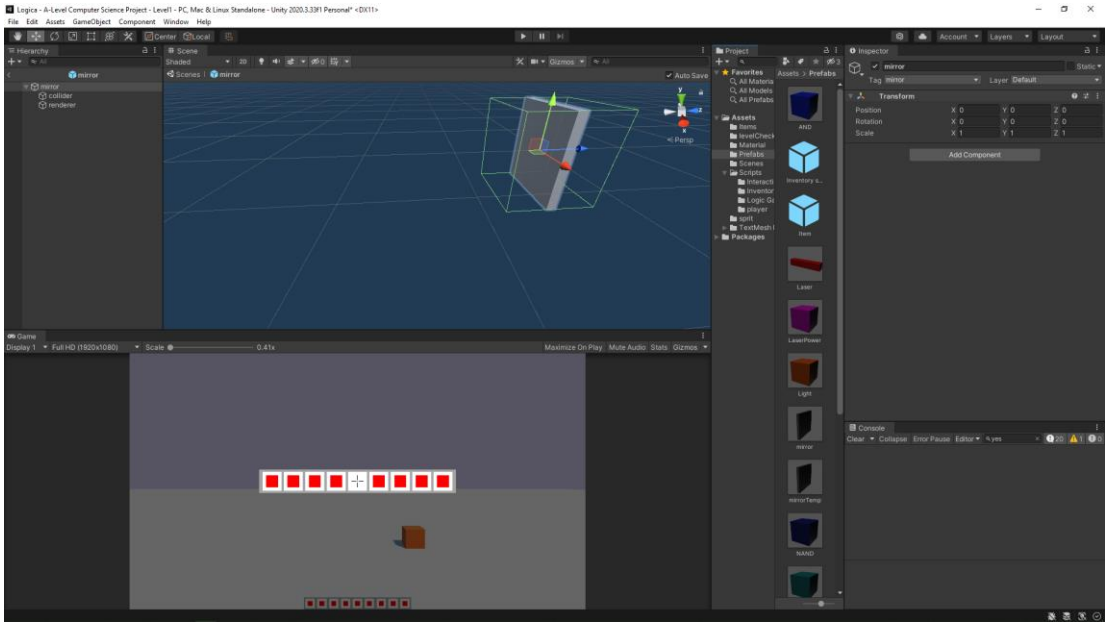
To fix this I added an or statement to check the rotation of the mirror for two other rotations that I have not accounted for when placing them facing in other directions.

After I added the extra conditions, the problem was fixed

```
if (tag == "mirror")
{
    // checks if the mirror is rotated 315 degrees to base the output in the correct direction
    if (Mathf.RoundToInt(hitObj.transform.parent.transform.eulerAngles.y) == 315 || Mathf.RoundToInt(hitObj.transform.parent.transform.eulerAngles.y) == 135)
    {
        if (dir.x != 0)
        {
            dir = new Vector3Int(0, 0, dir.x);
        }
        else if (dir.z != 0)
        {
            dir = new Vector3Int(dir.z, 0, 0);
        }
    }
    // checks if the mirror is rotated 45 degrees to base the output in the correct direction
    if (Mathf.RoundToInt(hitObj.transform.parent.transform.eulerAngles.y) == 45 || Mathf.RoundToInt(hitObj.transform.parent.transform.eulerAngles.y) == 225)
    {
        if (dir.x != 0)
        {
            // sets the new direction of the laser 90 degrees to the left
            dir = new Vector3Int(0, 0, -dir.x);
        }
        else if (dir.z != 0)
        {
            // sets the new direction of the laser 90 degrees to the right
            dir = new Vector3Int(-dir.z, 0, 0);
        }
    }
    genDis = 0;
    pos = hitObj.transform.position;
    bounced = true;
    gen = false;
}
```

Back to the collider problem.

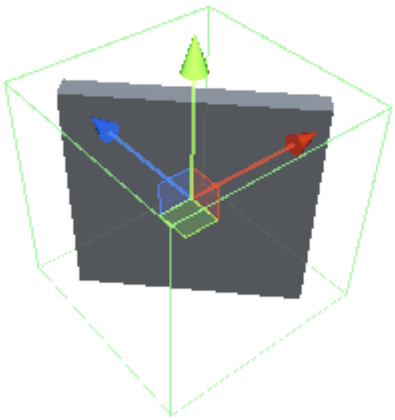
My next approach was to separate the mesh renderer and collider so that it can be a 1x1x1 box like other blocks and rotate everything else when placing it



Next, I added an if statement to check if the current placed object has any children. If they do, then I change the rotation of the first child which in the mirrors case will be the collider and counteract the rotation of the placement.

```
    }
    GameObject placed = Instantiate(selectObj, placePos, Quaternion.Euler(placeDir));
    if(placed.transform.childCount != 0)
    {
        mirrorChild = placed.transform.GetChild(0).gameObject;
        mirrorChild.transform.localRotation = Quaternion.Euler(new Vector3(0, 45, 0));
    }
```

This solved my problem, and I cannot place mirrors within one another due to the colliders acting like actual blocks.



This is the collider fixed in the image above.

After further consultation, my client is content with the functionality of Mirrors within the program.

WinCheck

My first approach with winCheck was to use circuits however this did not work as different circuits had different objects within them and it felt unreliable.

I changed my approach to add each of the circuits and to do this I first had to add new conditions to check for within generate. Firstly, I added to a check to see if lightObj is null then I added a check for onGate which is true when the generate script is on a gate and then iterate through each origin and removing the lightObject through each circuit.

I then created another foreach to remove each lightObject from the end circuit which is part of the endGen script which is a new script I created to manage the endCircuit list.

It then checks if there is onGate is false if so, it will still remove lightObject but this time does not access need extra conditions to check before removing lightObject.

It then sets lightObject to null

```
if(lightObj != null)
{

    if (onGate)
    {
        foreach (GameObject origin in GetComponent<gate>().origins)
        {
            if (origin != null)
            {
                origin.GetComponent<power>().circuit.Remove(lightObj);
            }
        }
        if (endLight.GetComponent<endGen>().endCircuit.Count != 0)
        {
            foreach (GameObject obj in endLight.GetComponent<endGen>().endCircuit.ToList())
            {
                endLight.GetComponent<endGen>().endCircuit.Remove(obj);
            }
        }
    }
    else
    {

        GetComponent<power>().circuit.Remove(lightObj);
        endLight.GetComponent<endGen>().endCircuit.Remove(gameObject);

    }
    lightObj = null;
```

I then also added a line within Sandbox to assign the lightObj of the scene to the placed down logic gate so that it knows what lightObj is.

```
            winCheck.powers.Add(placed.GetComponent<power>());
        }
        if(placed.GetComponent<generate>() != null)
        {
            generates.Add(placed.GetComponent<generate>());
        }
        if (placed.tag == "gate")
        {
            placed.GetComponent<generate>().endLight = lightObj;
        }

        foreach (power pow in winCheck.powers)
        {
            pow.gameObject.GetComponent<generate>().genUpdate();
        }
    }
}
```

I then started within endGen to iterate through each power script within the powers list that was stored in winCheck and made a condition to check if there were items within the list.

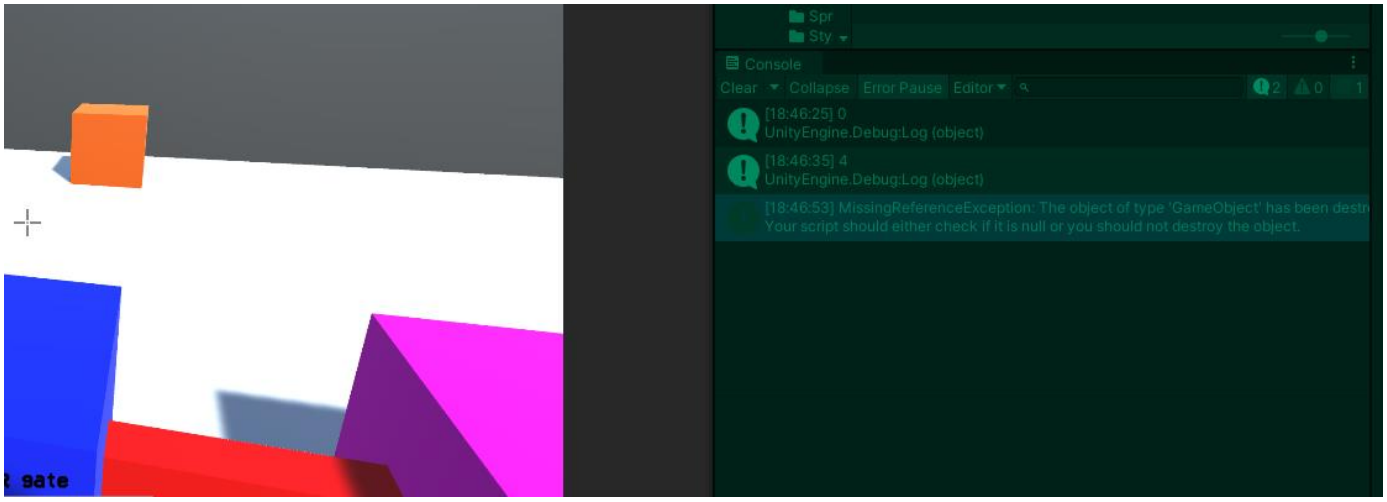
I next added another check within the previous if statement I added to check if the last position of the circuit is light. If so, then use the union function to join the current endCircuit with the circuit within the current power script that the foreach is on and remove any duplicates which is what the union function does. The code below shows the code that I added to the script.

```
Unity Message | 0 references
void Update()
{
        foreach (power pow in winCheck.powers)
        {
            if (pow.circuit.Count != 0)
            {
            if (pow.circuit.Last().name == "Light")
            {
                endCircuit = endCircuit.Union(pow.circuit).ToList();
                Debug.Log(gameObject.name);
                //while (endCircuit.Contains(gameObject.name == "light")) endCircuit.Remove(gameObject);
            }
            }
        }
}
```

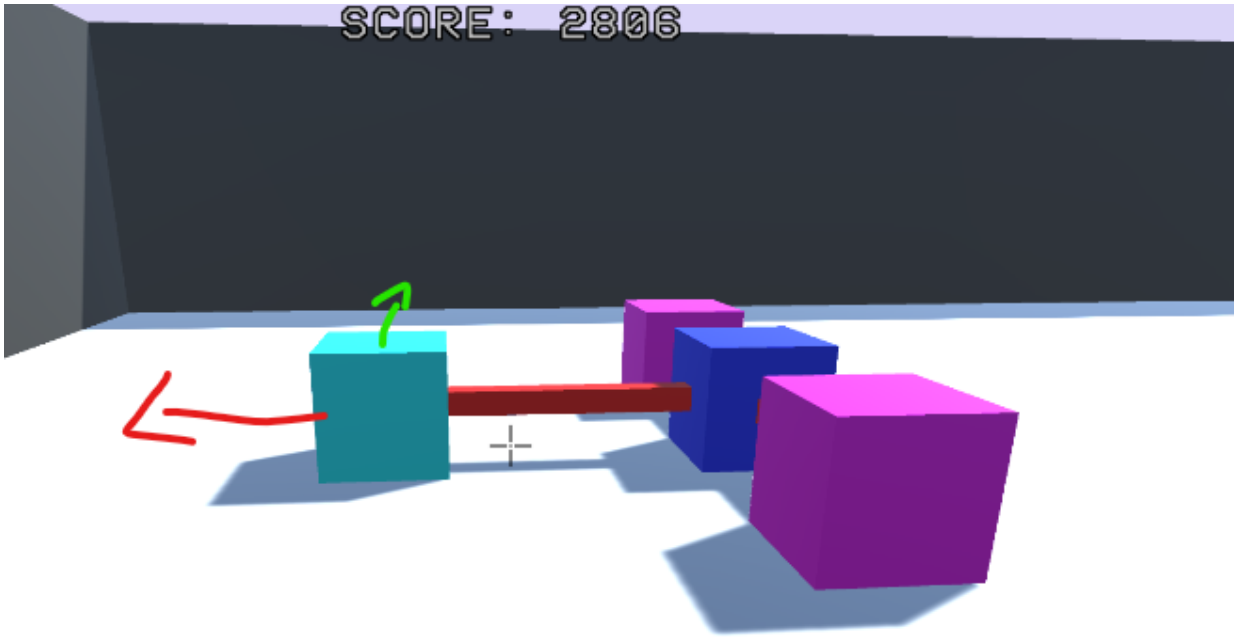
I then added checks to make sure that the gameObject that this script is attached to is not null which is should never be since it is the light object. I then checked if the current gameObject is called Light which it should be and then removed that from the endCircuit.

```
Unity Message | 0 references
void Update()
{
    foreach (power pow in winCheck.powers)
    {
        if (pow.circuit.Count != 0)
        {
            if (gameObject != null)
            {
                if (pow.circuit.Last().name == "Light")
                {
                    endCircuit = endCircuit.Union(pow.circuit).ToList();
                    if (gameObject.name == "Light")

                        endCircuit.Remove(gameObject);
                }
            }
        }
    }
}
```

However, after testing I ran into an error that I am still trying to access a destroyed gameObject.



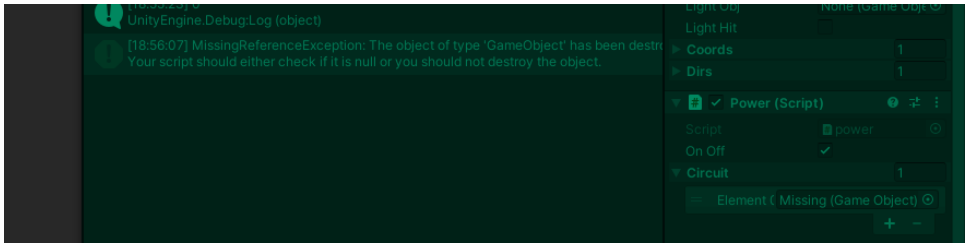
This error was occurring due to whenever I placed an object facing the wrong way (red arrow) it will not generate since it was not facing in the correct direction (green arrow). Therefore, Whenever I broke the object, it would return that error since that is the last item in the list and it is now null.



To fix this I added an extra condition to check if the name of the last object in the circuit is different to null so that it can then access the last object since the code now knows that it is different to null meaning that it exists and has a name. Therefore, allowing the code to check if the last objects name is equal to the Light

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    foreach (power pow in winCheck.powers)
    {
        if (pow.circuit.Count != 0)
        {
            if (gameObject != null)
            {
                if (pow.circuit.Last().name != null)
                {
                    if (pow.circuit.Last().name == "Light")
                    {
                        endCircuit = endCircuit.Union(pow.circuit).ToList();
                        if (gameObject.name == "Light")

                            endCircuit.Remove(gameObject);
                    }
                }
            }
        }
    }
}
```

However, this still did not work because I was checking if the name was different to null and not if the actual last object was null.



To fix this I removed '.name.

```
if (gameObject != null)
{
    if (pow.circuit.Last() != null)
    {
```

However, I now got a new error.

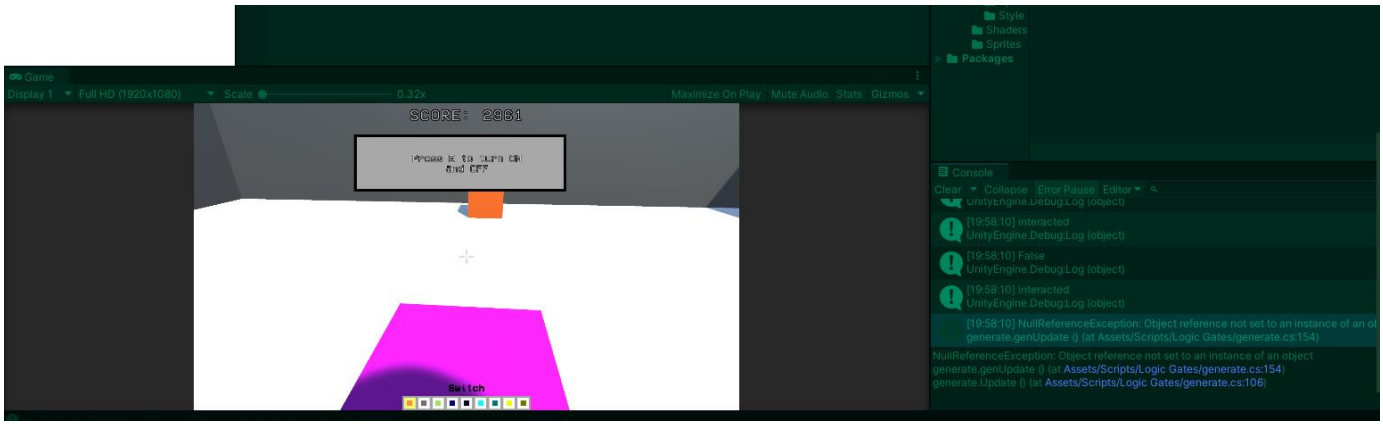
This error was caused by the when I added a new object to circuit it did not check the rotation and position of the input. To fix this I moved the code, so it was under the if(onGate) which was also part of another if statement that checked the position and rotation of the hitObj

```
if (onGate)
{
    foreach (GameObject origin in GetComponent<gate>().origins)
    {
        if (origin != null)
        {
            if (!hitObj.GetComponent<gate>().origins.Contains(origin))
            {
                hitObj.GetComponent<gate>().origins.Add(origin);
            }

        }
    }
    foreach (GameObject origin in GetComponent<gate>().origins)
    {
        if (origin != null)
        {
            if (!inList(hitObj, origin.GetComponent<power>().circuit))
            {
                origin.GetComponent<power>().circuit.Add(hitObj);

            }
        }
    }
}
else
```

I then simplified the top screenshot into one foreach to save lines of code and processing power.

```
if (onGate)
{
    foreach (GameObject origin in GetComponent<gate>().origins)
    {
        if (origin != null)
        {
            if (!hitObj.GetComponent<gate>().origins.Contains(origin))
            {
                hitObj.GetComponent<gate>().origins.Add(origin);
            }
            if (!inList(hitObj, origin.GetComponent<power>().circuit))
            {
                origin.GetComponent<power>().circuit.Add(hitObj);

            }
        }
    }
}
else
```

I then got another error when I turned the switch on and hit the light and then turned it back off.



To fix this, I added the line of code that removes the light from circuit and endCircuit from an else statement, I added a new condition using inList which is a procedure, I use this procedure a lot of times due to the nature of the program and using modularity massively helps code and maintain the program, I then check if the gameObject you are passing through is in the list that is the next parameter. I checked if the light as gameObject was in the endCircuit and then I removed it if it was true.

```
        }
    }
    else
    {
        if (inList(gameObject, lightObj.GetComponent<endGen>().endCircuit))
        {
            lightObj.GetComponent<endGen>().endCircuit.Remove(gameObject);
        }

    }
    lightObj = null;
}
```

This fixed the issue.

I then add another issue where now it would never add not gates into the circuit list because I had removed it outside of the else statement where it was checking if it was a not gate.

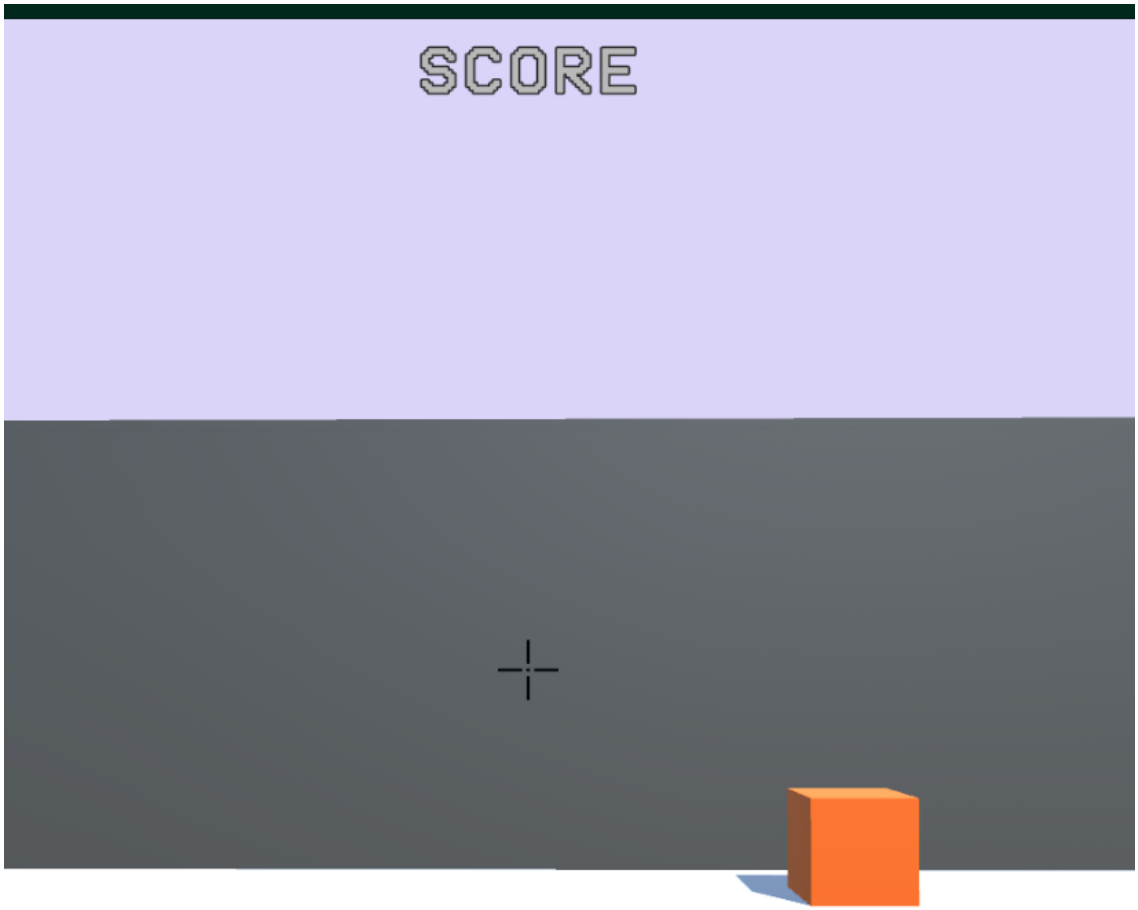
I fixed this by checking if the hitObj which is now a not gate is not in the list, and I then add it to the circuit list if it not in the list. Depending on the state of onGate depends on how I add the not Gate to the circuit.

```
if (hitObj.GetComponent<gate>().type == 0)
{
    if (Mathf.RoundToInt(hitObj.transform.eulerAngles.y) == 0 && dir.z == 1 || hitObj.transform.eulerAngles
    {
        if (!inList(gameObject, hitObj.GetComponent<gate>().inputs))
        {
            hitObj.GetComponent<gate>().inputs.Add(gameObject);
        }
        gate = hitObj;
        genDis = 25;
        gen = false;

        if (onGate)
        {
            foreach (GameObject origin in GetComponent<gate>().origins)
            {
                if (origin != null)
                {
                    if (!hitObj.GetComponent<gate>().origins.Contains(origin))
                    {
                        hitObj.GetComponent<gate>().origins.Add(origin);
                    }
                    if (!inList(hitObj, origin.GetComponent<power>().circuit))
                    {
                        origin.GetComponent<power>().circuit.Add(hitObj);

                    }
                }
            }
        }
        else
        {
            if (!hitObj.GetComponent<gate>().origins.Contains(gameObject))
            {
                hitObj.GetComponent<gate>().origins.Add(gameObject);
            }
            if (!inList(hitObj, GetComponent<power>().circuit))
            {
                GetComponent<power>().circuit.Add(hitObj);
            }
        }
        hitObj.GetComponent<generate>().genUpdate();
```

After further consultation, my client is content with the functionality of the checking if the win condition has been reached.

Score for 3D

I must add score to the levels so that the students' progress can be tracked



I first started off by initialising the variables

I made a scoreDisplay variable where I will drag in the display for the score to be updated on the screen

I then made a player score variable along with a timer. Both will be integers because take up less space than a float.

```
Unity Script | 0 references
public class score : MonoBehaviour
{
    public TextMeshProUGUI scoreDisp;
    int pScore;
    private int timer;
```

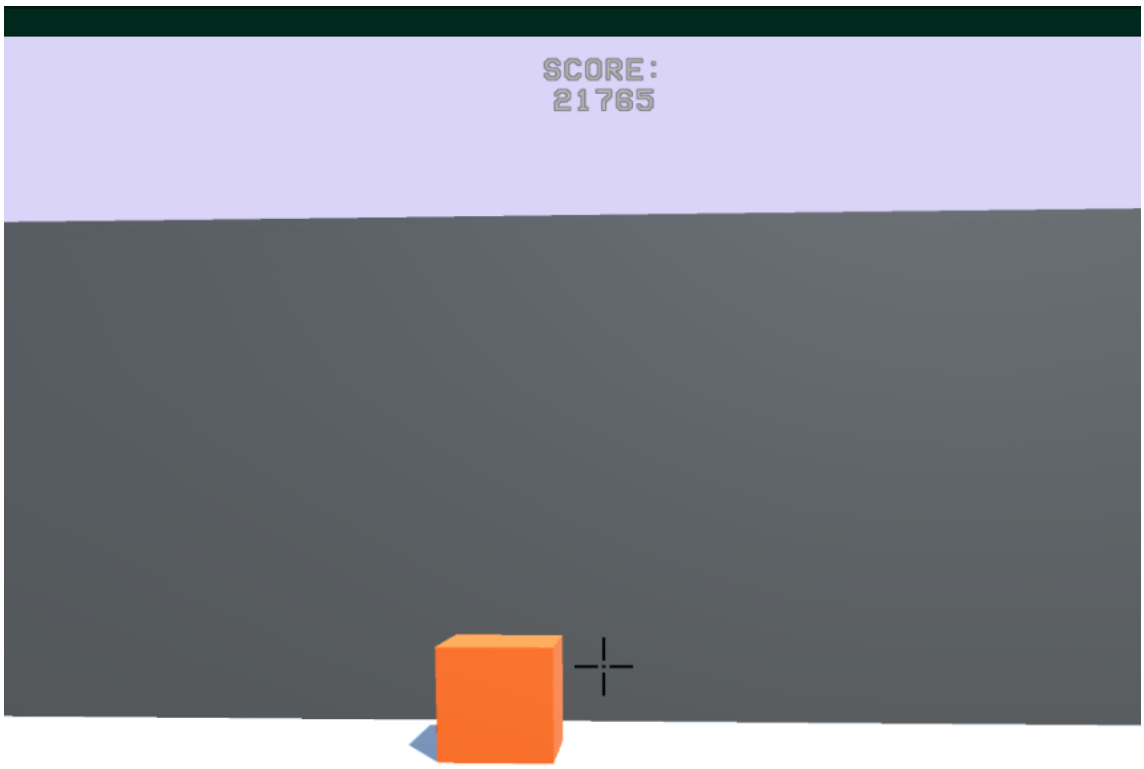
I then initialised player Score to be 0 in the start function.

In the update function I used time.deltaTime which is an inbuilt Unity function that takes the time between the last frame and the current frame

The code then checks to see if the timer as reached more than 5 if so then player score will increase. I then update the score display text by changing the score to a string.

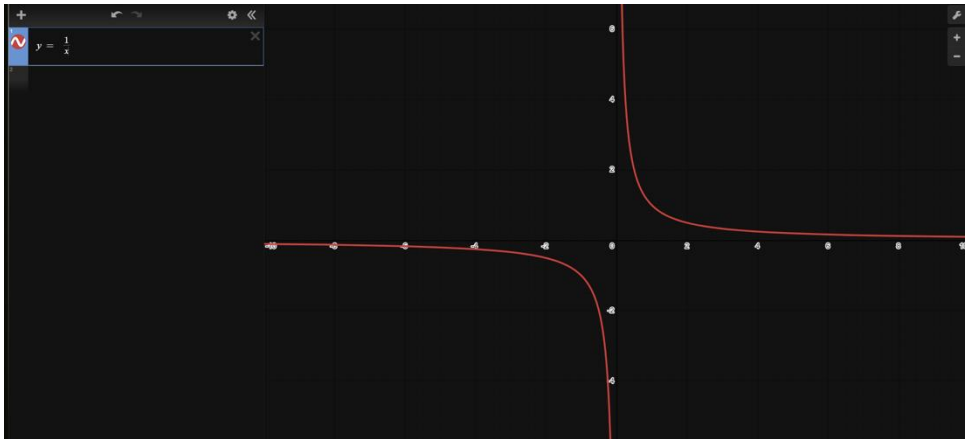
```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    pScore = 0;
}

Unity Message | 0 references
void Update()
{

    timer = (int)Time.deltaTime;

    if (timer > 5f)
    {

        pScore += 5;

        //We only need to update the text if the score changed.
        scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

        //Reset the timer to 0.
        timer = 0;
    }
}
}
```

This works fine as shown below.

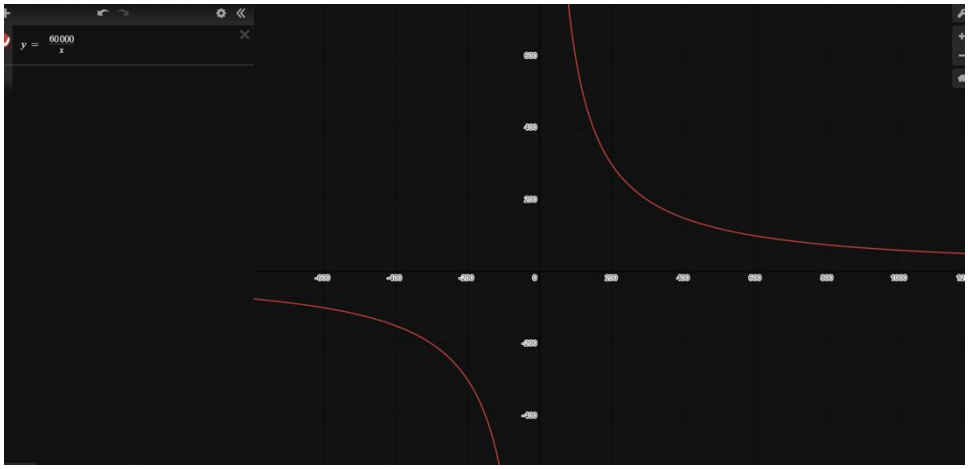


However, I want the score to be indirectly proportional to the time, meaning that the less time you take to complete a level the higher your score.

My first approach was using a maths function such as 1/x.



This graph is what I was looking for as it is indirectly proportional to the time however it was too steep, and the score decreased too fast to fix this I changed the values of the function to match my needs better.



This graph suits my idea for how I want to the score to change as time goes on.

I now started to implement this into my code

```
public TextMeshProUGUI scoreDisp;
int pScore;
private int timer;

// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    pScore = 1;
}

Unity Message | 0 references
void Update()
{

    timer = Mathf.FloorToInt(Time.timeSinceLevelLoad);

    if (timer != 0)
    {
        pScore = 60000 / timer;
    }
    //We only need to update the text if the score changed.
    scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

}
```

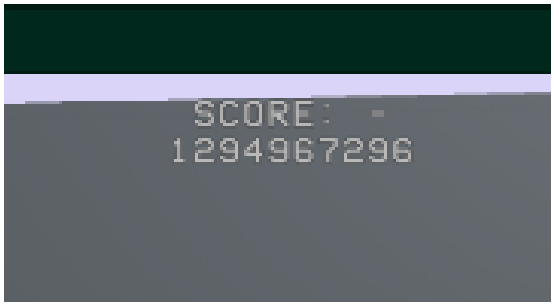
I made an if statement that while timer is not equal to 0 the player score will be 6000 divided by the timer.

However, this decreased way too quickly

To fix this, I increased the starting score to a long data type and had to change it to int after I divided it by the timer.

```
timer = Mathf.FloorToInt(Time.timeSinceLevelLoad);

if (timer != 0)
{
    pScore = (int)( 6000000000 / timer);
}
//We only need to update the text if the score changed.
scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();
```

However, this caused some issues, and the score did not feel natural.

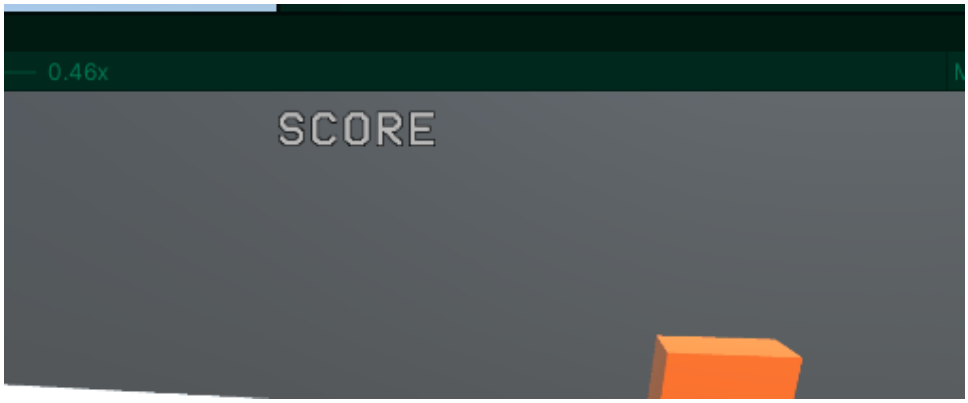


I consulted my client about the score and Mr Flain said it would be much better if the score starts at a certain value and decreases by a constant since the current score did not feel natural and did not fully work either.

To start implementing my clients plan of how to measure the score, I initialised the players score to start at 10,000. I then checked to see if the timer has passed more than 5 frames. I then took away player score by 5 and reset the timer to 0.

```
Unity Script | 0 references
public class score : MonoBehaviour
{
    public TextMeshProUGUI scoreDisp;
    int pScore;
    private int timer;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        pScore = 10000;
    }

    Unity Message | 0 references
    void Update()
    {

        timer = Mathf.FloorToInt(Time.timeSinceLevelLoad);

        if (timer > 5f)
        {

            pScore -= 5;

            //We only need to update the text if the score changed.
            scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

            //Reset the timer to 0.
            timer = 0;
        }
    }
}
```

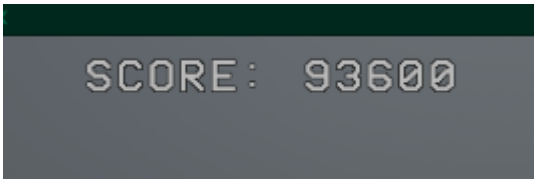
However, this made score take too long to show up and felt too slow

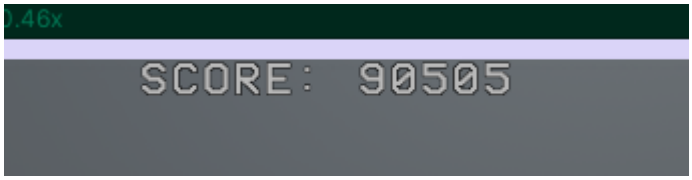


To fix this I made the timer check at 1 frame instead of 5

```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    pScore = 100000;
}

Unity Message | 0 references
void Update()
{

    timer = Mathf.FloorToInt(Time.timeSinceLevelLoad);
    Debug.Log(timer);
    if (timer > 1f)
    {

        pScore -= 5;

        //We only need to update the text if the score changed.
        scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

        //Reset the timer to 0.
        timer = 0;
    }
}
```

This fixed the issue and made the score decrease





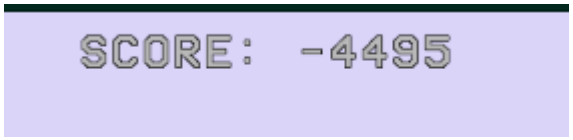
I then tried to access the winCheck script which checks for if the order of the placed gates is the same as the as the in-built answer of what order the gates should be in

```
if (step == levels[level].gateOrder.Count)
{
    win = true;
}
```

I made the score only decreases if the win variable from the winCheck script is false. If not, then finalScore is set to the player Score. I currently have it to debug a message whenever you win, and the score will stop

```
public winCheck winCheck;

// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    pScore = 100000;
}

Unity Message | 0 references
void Update()
{
    if (!winCheck.win)
    {
        timer = Mathf.FloorToInt(Time.timeSinceLevelLoad);
        pScore -= 5;

        //We only need to update the text if the score changed.
        scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

        //Reset the timer to 0.
        timer = 0;
    }
    else
    {
         finalScore = pScore;
        Debug.Log("YIPEE");
    }
}
```

However, I still have a problem where my score can go below 0 which should not be possible as it should end the game.



To fix this I added another if statement to make sure that player score is larger than 0 and added another debug after that to show that you ran out of time.

```
void Start()
{
    pScore = 216000;
}

Unity Message | 0 references
void Update()
{
    if (pScore > 0)
    {
        if (!winCheck.win)
        {
            timer = Mathf.FloorToInt(Time.timeSinceLevelLoad);
            pScore -= 5;

            //We only need to update the text if the score changed.
            scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

            //Reset the timer to 0.
            timer = 0;
        }
```

However, this is not a suitable solution due to the current timer not considering PC specifications as it may take longer to process certain things, and this will unfairly treat some students especially since this will be used on school computers.

```
Unity Message | 0 references
void Update()
{
    if (pScore > 0)
    {
        if (!winCheck.win)
        {
            pScore = (float)pScore;
            pScore -= 5 * Time.deltaTime;
            pScore = (int)pScore;
            //We only need to update the text if the score changed.
            scoreDisp.text = scoreDisp.text = "Score: " + pScore.ToString();

            //Reset the timer to 0.
            timer = 0;
        }
        else
        {
            finalScore = (int)pScore;
            Debug.Log("YIPEE");
        }
    }
    else
    {
        Debug.Log("You ran out of time");
    }
}
```

To fix this issue I made player Score be affected by Time.deltaTime instead of and using 10 as the speed constant and which the score will decrease.

```
if (pScore > 0)
{
    if (!winCheck.win)
    {
        pScore -= 10 * Time.deltaTime;
        //We only need to update the text if the score changed.
        scoreDisp.text = scoreDisp.text = "Score: " + Mathf.FloorToInt(pScore).ToString();

        //Reset the timer to 0.
        timer = 0;
    }
    else
    {
        finalScore = (int)pScore;
        Debug.Log("YIPEE");
    }
```

I changed pScore to be a float and floored the score so that it gets rounded on the display text

After testing this and talking to my client we both agreed that this is the best solution due to it feeling natural and gives the students a good understanding of the time they have left.

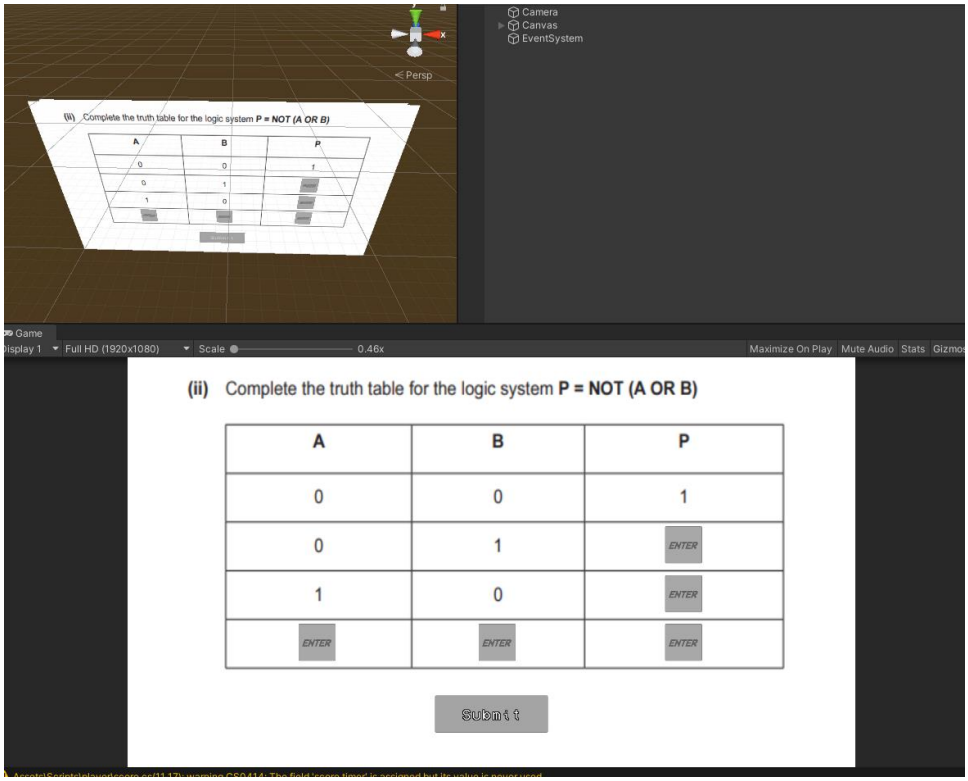
After further consultation, my client is content with the implementation of score within 3D levels.

2D Levels

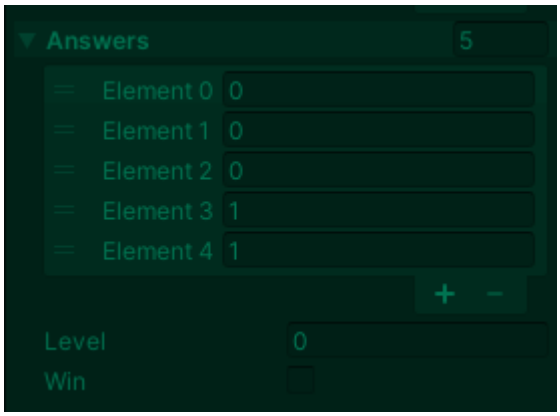
Next, I had to code and make the 2D Levels to make more

```
Unity Script | 1 reference
public class levelSO : ScriptableObject
{
    public List<int> gateOrder;
    public List<int> truthOrder;
}
```

For the first type of question which is where students will have to input 1s and 0s into a truth table I first made a list called truthOrder. This variable is the order of the correct answer that will later be compared to when checking for a win.



I used the image from my level design and added input boxes along with a submit button.



I added the answers into the level scriptable object.

(ii) Complete the truth table for the logic system **P = NOT (A OR B)**

A	B	P
0	0	1
0	1	0
1	0	0
1	1	0

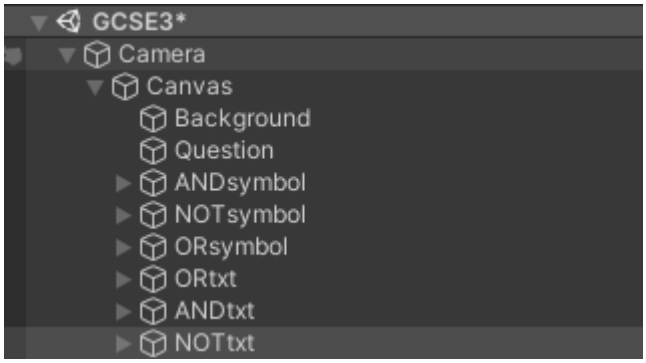
Submit

This should be the correct answer shown above.

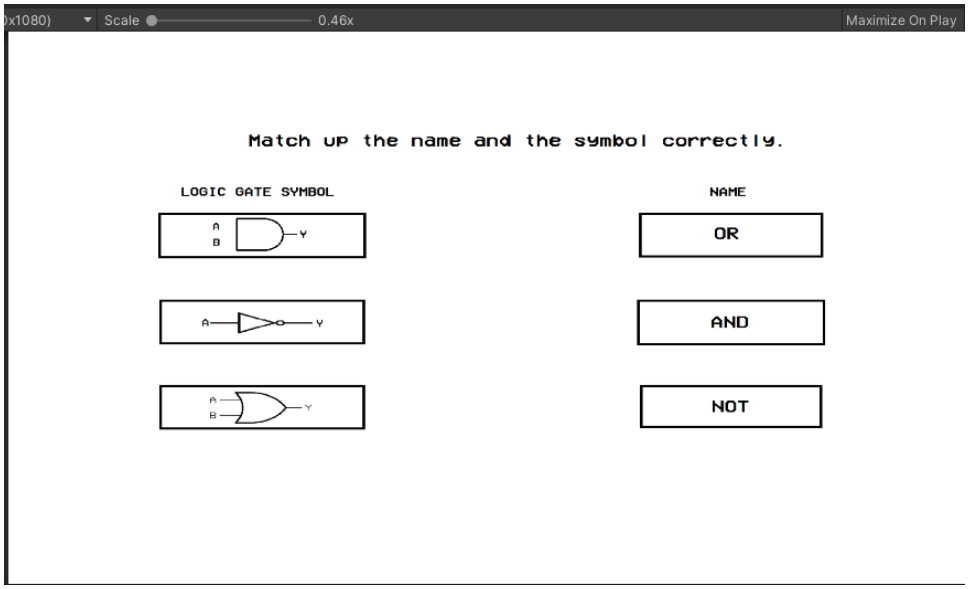
```
0 references
public void truthInput()
{

    List<string> temp = new List<string> { TL.text, ML.text, BL.text, BM.text, BR.text };
    answers = temp;

    step = 0;
    foreach (string i in answers) {
        if (step < levels[level].truthOrder.Count)
        {
            Debug.Log(i);
            Debug.Log(levels[level].truthOrder[step]);
            if (i == levels[level].truthOrder[step])
            {
                step += 1;

            }
        }
    }

    if (step == levels[level].truthOrder.Count)
    {
        win = true;
    }
}
```

Next, I reused code from my previous win Check for 3D levels to develop this. The code worked and win became true after I inputted the correct answer and press submit.

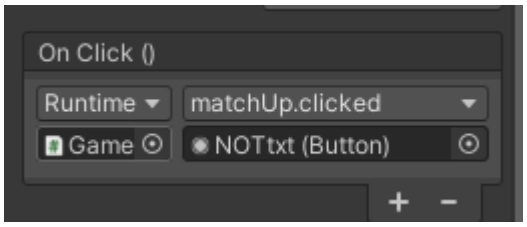
Next, I had to work on the next type of level which is where you must match up each name to the logic gate symbol



Here is the screen I made for this level taking inspiration from an exam question that I found in my level design.



I first created a separate script and attached this to every button within the level



I next coded what the action of clicking the button will do. I first passed in the button as a button object, so I was able to access the buttons name and tag.

My first approach was to check if the number of buttons that I have clicked is not equal to 2 then I will check to make sure oldTag is null because I need to assign oldTag only when tag already has been assigned so now in the else statement I assigned. I then checked in an else statement which means whenever there is 3,1 or 0 clicked it will check if oldTag and tag equal each other and current name and oldName do not match each other so that the code knows the player has selected two separate buttons with the same tag meaning they are matched.

Else it sets the event system on the selectGameObject to null. My intent was to make the buttons disappear and then I set buttonsclicked to be 0. However, this did not work since it did not go to true

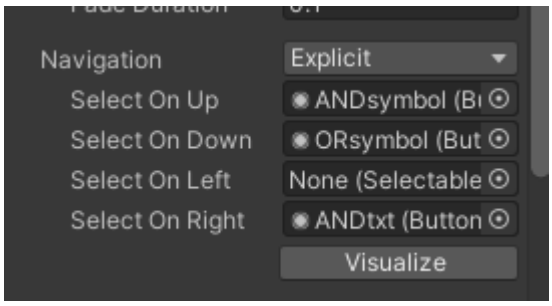
when I clicked two of the same.

```
0 references
public void clicked(Button button)
{
    if (buttonsClicked != 2) {
        if (oldTag == null)
        {
            tag = button.tag;
            currentName = button.name;
        }
        else
        {
            oldTag = tag;
            oldName = currentName;
            tag = button.tag;
            currentName = button.name;
        }
        buttonsClicked++;
    }
    else
        {
            if(oldTag == tag && currentName != oldName)
            {
                win = true;
            }
            else
            {
                GameObject myEventSystem = GameObject.Find("EventSystem");
                myEventSystem.GetComponent<UnityEngine.EventSystems.EventSystem>().SetSelectedGameObject(null);
            }
            buttonsClicked = 0;
        }
}
```

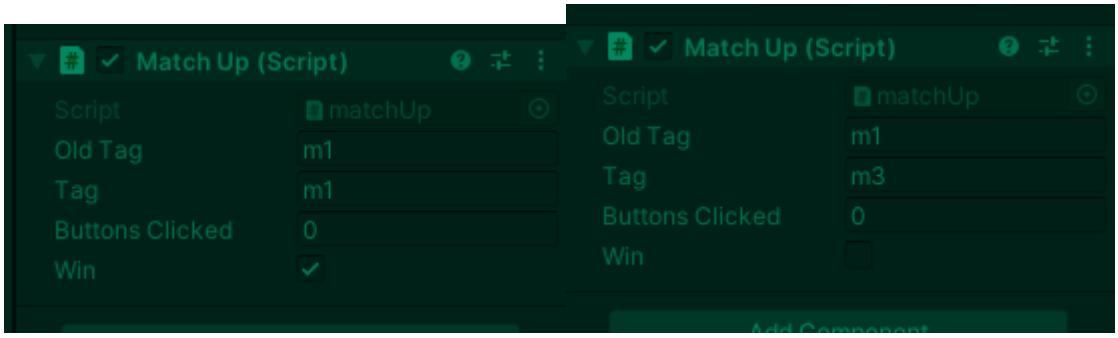
My next iteration of this approach was to remove the else so that it always checks if you have matched something and not when you have clicked not clicked 2 buttons since I did not consider the player matching two straight away.

```
public void clicked(Button button)
{
    if (buttonsClicked != 2) {
        if (oldTag == null)
        {
            tag = button.tag;
            currentName = button.name;
        }
    else
    {
        oldTag = tag;
        oldName = currentName;
        tag = button.tag;
        currentName = button.name;
    }
    buttonsClicked++;
    if (oldTag == tag && currentName != oldName)
    {
        win = true;
        buttonsClicked = 0;
    }
    else
    {
        GameObject myEventSystem = GameObject.Find("EventSystem");
        myEventSystem.GetComponent<UnityEngine.EventSystems.EventSystem>().SetSelectedGameObject(null);
        buttonsClicked = 0;
    }
    }
}
```

Next, I added navigation to this level using the arrow keys so it's easier to traverse between the buttons



After this I tested the code, and it worked since whenever I matched two objects win would become trye



However, when I added this new if statement that checked for if there were two buttons clicked and win was false, I got an error which I did not understand

```
if(buttonsClicked == 2 && !win)
{
    GameObject myEventSystem = GameObject.Find("EventSystem");
    myEventSystem.GetComponent<UnityEngine.EventSystems.EventSystem>().SetSelectedGameObject(null);
    buttonsClicked = 0;
}
```

```
[17:27:49] NullReferenceException: Object reference not set to an instance of an object
matchUp.clicked (UnityEngine.UI.Button button) (at Assets/Scripts/Levels/matchUp.cs:41)
```

To counter this, I changed my approach a bit and made it so that whenever I match two buttons, they will stay red.

```
public void clicked(Button button)
{
    if (buttonsClicked != 2) {
        if (oldButton.tag == null)
        {
            tag = button.tag;
            currentName = button.name;
        }
        else
        {
            oldButton.tag = tag;
            oldButton.name = currentName;
            tag = button.tag;
            currentName = button.name;
        }
        buttonsClicked++;
        if (oldButton.tag == tag && currentName != oldButton.name)
        {
            oldButton.GetComponent<Image>().color = new Color32(255 , 0, 0,134);
            button.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
            win = true;
            buttonsClicked = 0;
        }
        if(buttonsClicked == 2 && !win)
        {
            button.GetComponent<Image>().color = new Color32(0, 0, 0, 0);
            buttonsClicked = 0;
        }
    }
}
```

I first made the entire function be inside the buttonsClicked is not equal to two and I also added me getting the image components to change the colour of a button to an opacified red so that it is visible when you have matched two of the correct items. However, if they do not match it resets to its original colour

```
[17:29:40] NullReferenceException: Object reference not set to an instance of an object
matchUp.clicked (UnityEngine.UI.Button button) (at Assets/Scripts/Levels/matchUp.cs:40)
```

However, I got an error where the object reference is not set to an instance.

Initially to fix this I removed oldTag and used oldButton.tag to access the oldTag thinking this will be more efficient

```
0 references
public void clicked(Button button)
{
    if (buttonsClicked != 2) {
        if (oldButton == null)
        {
            tag = button.tag;
            currentName = button.name;
        }
        else
        {
            oldButton = button;
            tag = button.tag;
            currentName = button.name;
        }
        buttonsClicked++;
        if (oldButton != null)
        {
            if (oldButton.tag == tag && currentName != oldButton.name)
            {
                oldButton.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
                button.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
                win = true;
                buttonsClicked = 0;
            }
            if (buttonsClicked == 2 && !win)
            {
                button.GetComponent<Image>().color = new Color32(0, 0, 0, 0);
                buttonsClicked = 0;
            }
        }
    }
}
```

However, this did not fix the issue, so I undid my change with trying to access old objects via .name and. tag and made the old button also get set to its original state as well as the current button.

```
0 references
public void clicked(Button button)
{
    if (buttonsClicked != 2) {
        if (oldTag == null)
        {
            tag = button.tag;
            currentName = button.name;
        }
        else
        {
            oldTag = tag;
            oldName = currentName;
            oldButton = button;
            tag = button.tag;
            currentName = button.name;
        }
        buttonsClicked++;
        if (oldTag == tag && currentName != oldName)
        {
            win = true;
            oldButton.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
            button.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
            buttonsClicked = 0;
        }
        if(buttonsClicked == 2 && !win)
        {
            oldButton.GetComponent<Image>().color = new Color32(0, 0, 0, 0);
            button.GetComponent<Image>().color = new Color32(0,0,0,0);
            buttonsClicked = 0;
        }
    }
}
```

However, I also did not take into consideration that I had to reset all the values when I clicked two buttons since they did not match

```
public void clicked(Button button)
{
    if (buttonsClicked <= 2) {
        if (buttonsClicked==0)
        {
            tag = button.tag;
            currentName = button.name;
            oldButton = button;
        }
        else
        {
            oldTag = tag;
            oldName = currentName;
            tag = button.tag;
            currentName = button.name;
        }
        buttonsClicked++;
        if (oldButton != null)
        {
            if (oldTag == tag && currentName != oldName)
            {
                win = true;
                oldButton.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
                button.GetComponent<Image>().color = new Color32(255, 0, 0, 134);
            }
            if (buttonsClicked == 2 && !win)
            {
                oldButton.GetComponent<Image>().color = new Color32(0, 0, 0, 0);
                button.GetComponent<Image>().color = new Color32(0, 0, 0, 0);
            }

        }
        if (buttonsClicked == 2)
        {
            buttonsClicked = 0;
            oldButton = null;
            oldTag = null;
            oldName = null;
        }
    }
}
```

After I made this change the matching process worked however now I must check how many are matched.

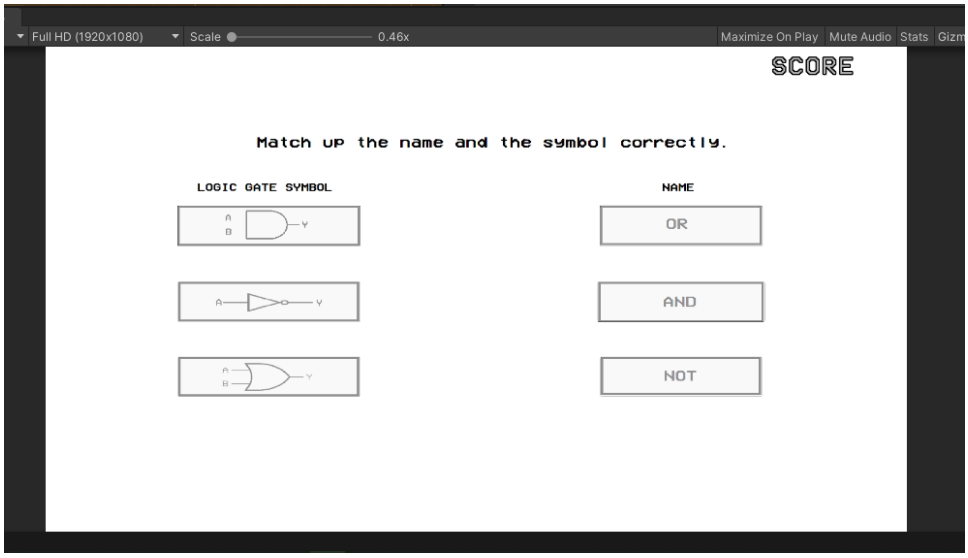
```
if (oldTag == tag && currentName != oldName)
{
    matched ++;
    match = true;
    buttonsClicked = 0;
```

To do this I created a new variable called matched which will increment every time a match happens.

I next added the score into this level by reusing the same code as before.

```
public TextMeshProUGUI scoreDisp;
float pScore;
int finalScore;

// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    pScore = 600f;
}

Unity Message | 0 references
void Update()
{
    if (pScore > 0)
    {
        if (!win)
        {
            pScore -= 10 * Time.deltaTime;
            //We only need to update the text if the score changed.
            scoreDisp.text = scoreDisp.text = "Score: " + Mathf.FloorToInt(pScore).ToString();
        }
        else
        {
            finalScore = (int)pScore;
            Debug.Log("YIPEE");
        }
    }
    else
    {
        Debug.Log("You ran out of time");
    }
}
```

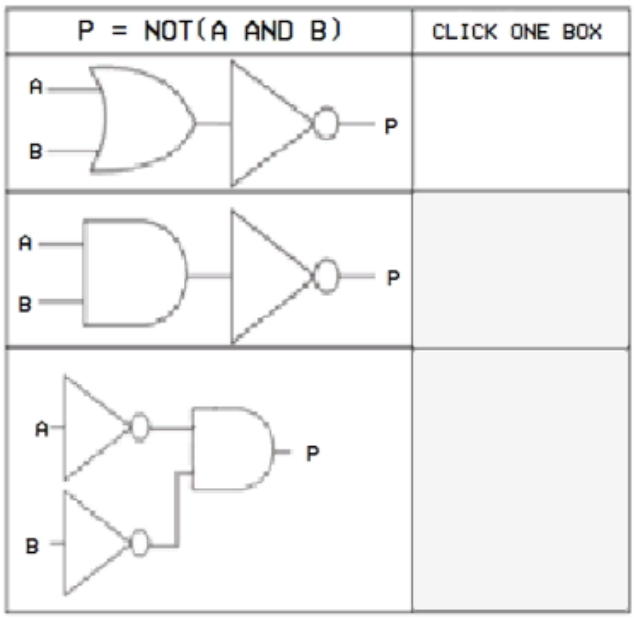
This is how my level looks including the score.



Next was making the multiple-choice style of question

This level was again inspired by an exam question I found during the design stage.

I first created this layout and including buttons in each of the boxes for the player to click



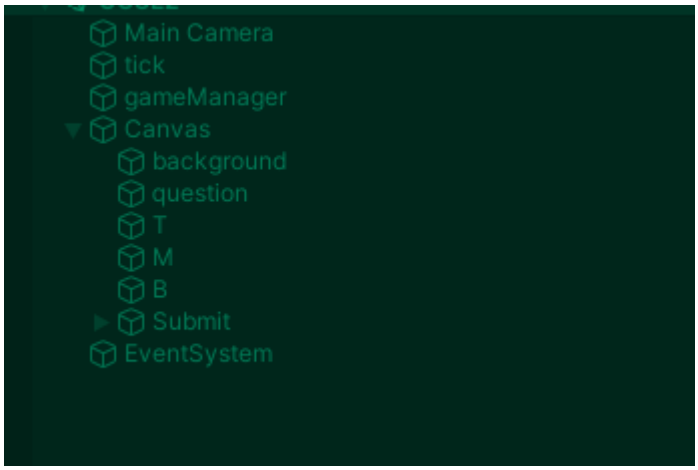
I first made two different procedures to execute

```
{
    public Vector3 pos;
    public GameObject tick;
    public bool win;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
    0 references
    public void ticked(Transform transform)
    {
        pos = transform.position;
        GameObject ticked = Instantiate(tick, pos, Quaternion.identity);
        ticked.name = "tick";
    }
    0 references
    public void ticked(Button button)
    {
       if(button.name == "T")
        {
            win = true;
        }
    }
}
```

The first procedure is used to instantiate the tick in the same position as the button that you clicked.

I then set the instantiated objects name to be tick

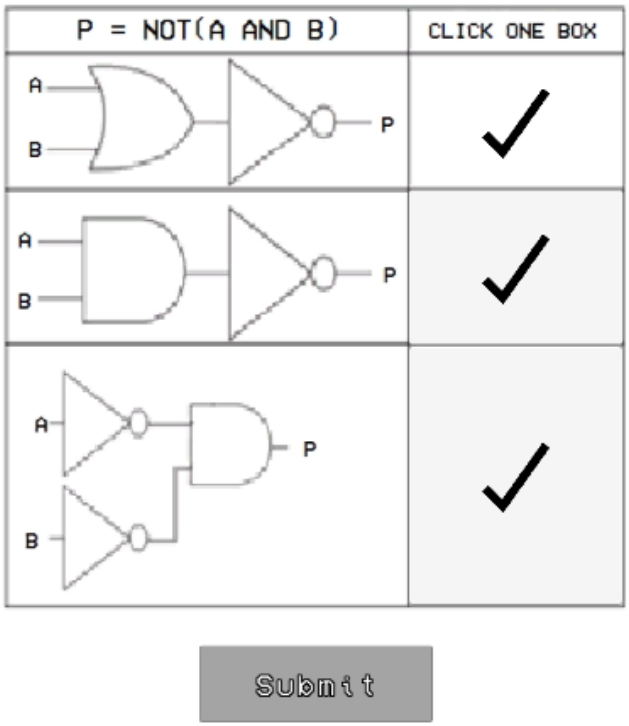
The next procedure is used to check if you have ticked the correct button, currently it has the same name as the other procedure, but I will change this shortly



However, when I instantiated, the tick did not appear on screen since it was instantiated into the scene and was not set to be a child of the canvas

```
public void ticked(Transform transform)
{
    pos = transform.position;
    GameObject ticked = Instantiate(tick, pos, Quaternion.identity);
    ticked.name = "tick";
    ticked.transform.SetParent(canvas);
}
0 references
```

In the code above I got the transform and set its parent to be the canvas which is a new variable I added that is initialised as to what object is the canvas in the inspector.



However, I got a problem that I was able to tick multiple/ all options.

My solution was to create a new variable that temporarily stores the position of the tick.

```
0 references
public void ticked(Transform transform)
{
    if(oldPos == null)
    {
        pos = transform.position;
        GameObject temp = Instantiate(tick, pos, Quaternion.identity);
        temp.name = "tick";
        temp.transform.SetParent(canvas);
        tickNew = temp;
    }
    else
    {
        Destroy(tickNew);
        pos = transform.position;
        GameObject temp = Instantiate(tick, pos, Quaternion.identity);
        temp.name = "tick";
        temp.transform.SetParent(canvas);
        oldPos = pos;
        tickNew = temp;
    }
```

In the code above I first check if old position is null if so then I get the position of the buttons transform and set that to pos. I then instantiate the tick prefab at the pos variable that we just declared to be the same as the buttons position and then I set the rotation to be quaternion. Identity which is the default. Next, I set the name to be tick and its parent to be the canvas and the I set tickNew which is local to the entire script and not just in the if statement (which temp is).

In the else statement which only happens if oldPos is not null. I delete newTick which will never be null because of the if and else statement and then I redo the same code as the if statement but this time I set oldPos to be pos so that the code now always branches to the else statement.

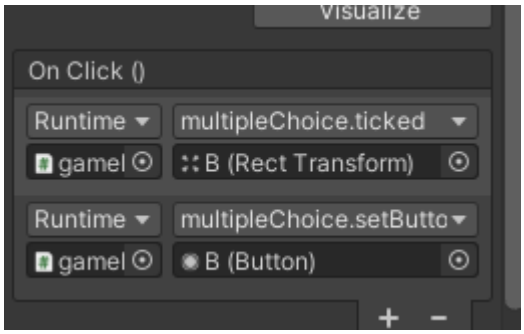
```
0 references
public void ifwin(Button button)
{
    if(button.name == "M")
    {
        win = true;
        Debug.Log("win");
    }
}
```

Next, I found a problem with the ifwin procedure. When I press submit this will not check if the correct thing has been ticked since it will pass in the submit button and not the one that the player has pressed. This is shown below

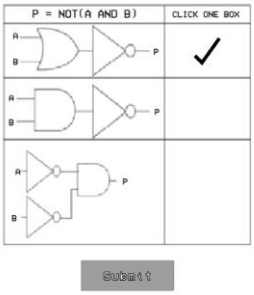
To fix this made I made all the choices execute the set Button procedure set name to the buttons name. Now when I press submit It will compare name to M which is the answer

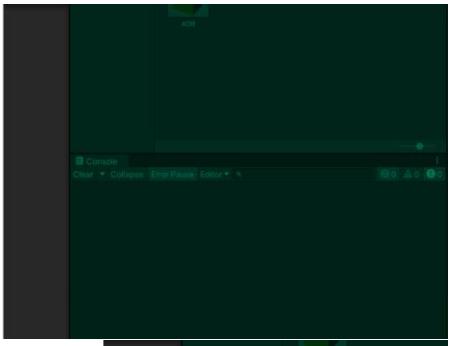
```
0 references
public void setButton(Button button)
{
    name = button.name;
}
0 references
public void ifwin(Button button)
{
    if(name == "M")
    {
        win = true;
        Debug.Log("win");
    }
```

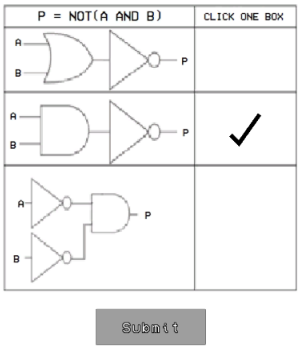
As you can see below this is how I made each button execute each void / procedure.

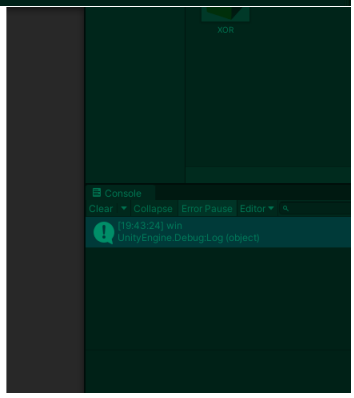


Below are screenshots showcasing it working whenever I press submit.

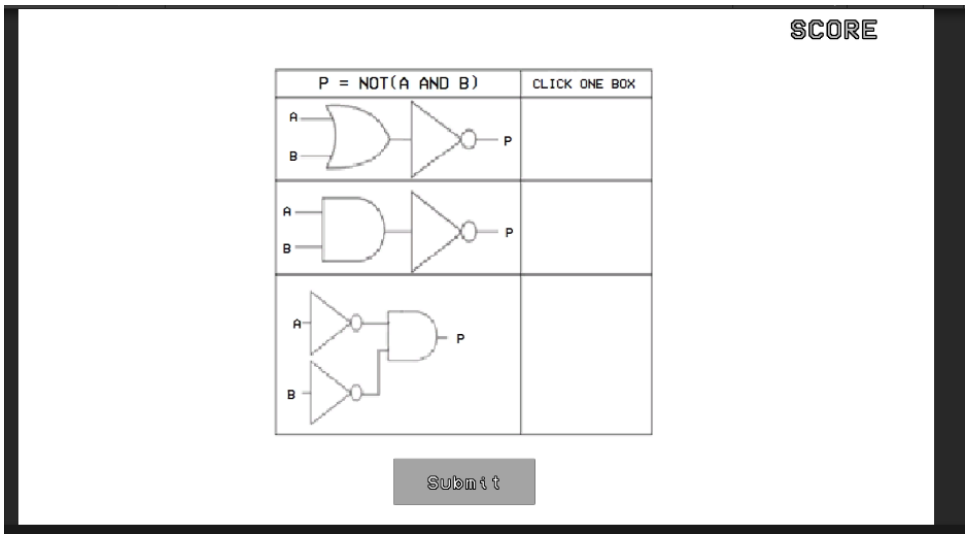








I next added the score. Reusing the same code as the previous three times.



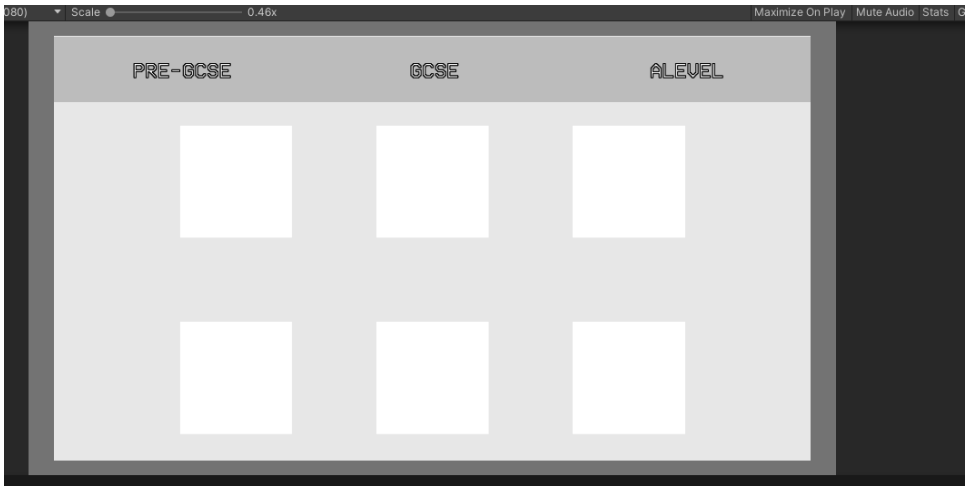
```
void Start()
{
    pScore = 600f;
}

void Update()
{
    if (pScore > 0)
    {
        if (!win)
        {
            pScore -= 10 * Time.deltaTime;
            //We only need to update the text if the score changed.
            scoreDisp.text = scoreDisp.text = "Score: " + Mathf.FloorToInt(pScore).ToString();
        }
        else
        {
            finalScore = (int)pScore;
            Debug.Log("YIPEE");
        }
    }
    else
    {
        Debug.Log("You ran out of time");
    }
}
```

Here is the reused code.

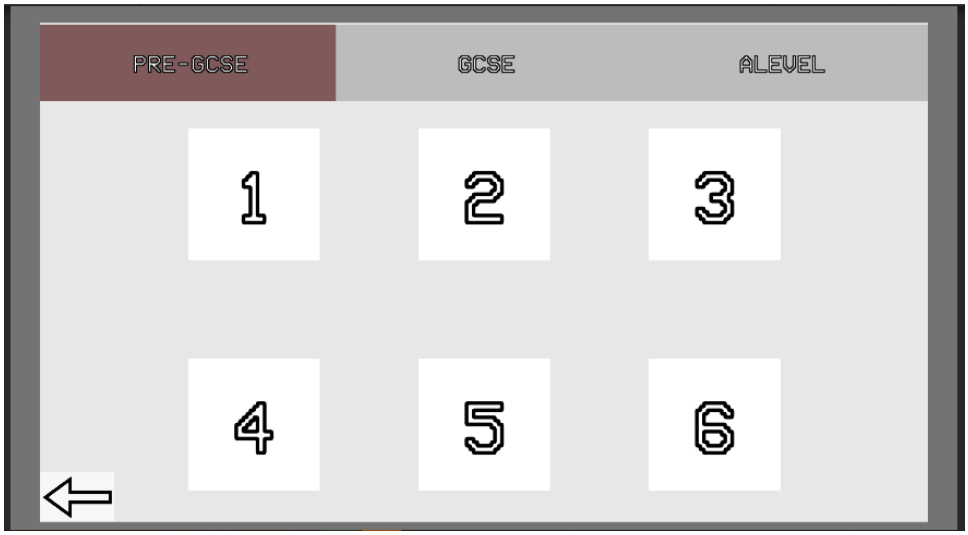
After further consultation, my client is content with the functionality of the 2D levels, and he agrees that they fulfil their purpose.

Menu

I next started to work on the levelSelection screens.



I created 3 scenes each of them highlighting the tabs and being able to access each scene by pressing the corresponding tabs



I next created a screen Director script that directs each of the buttons to execute loading a new scene. I had multiple choices on how to approach this, but I decided to create a MainMenu procedure to load the Main Menu whenever this procedure is executed.

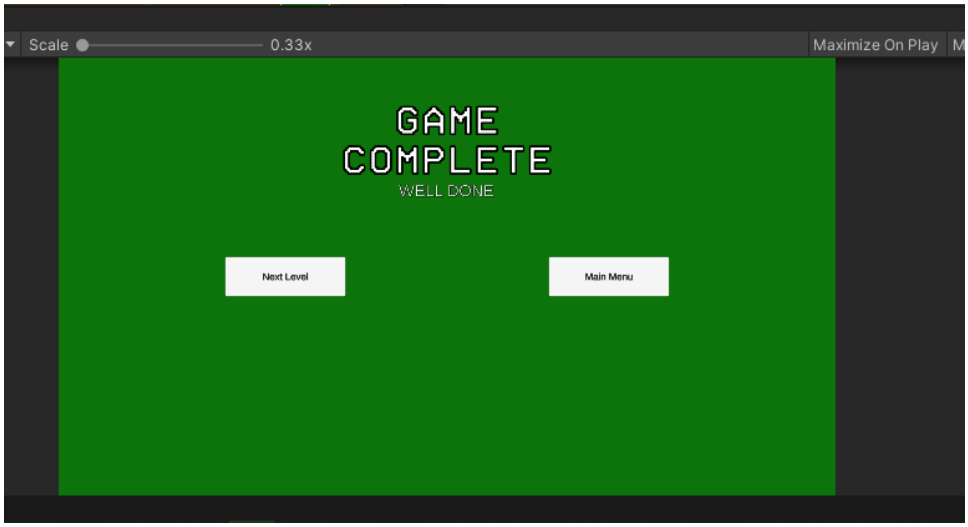
I then created a Load Button which will load the button using the button name which will be the index position. I decided to do this to save memory as I will have to copy and paste the same procedures over 40 times since this project will have around 40 scenes especially for scenes that I will only be loaded by a singular button unlike Main Menu which will have multiple buttons leading to that scene. I intended to use stacks to Push the button name so that I can pop it from the stack to go back.

```
public void MainMenu()
{
    SceneManager.LoadScene(13);
}
0 references
public void Load(Button button)
{
    SceneManager.LoadScene(Int32.Parse(button.name));
    levelindex = Int32.Parse(button.name);
    Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
    sName = temp.name;
    //levelStack.Push(Int32.Parse(button.name));
}
```

I created the nextLevel procedure which I will use at the gameComplete scene to access the next level this try to access sName variable which was assigned in the load void. This will check if the scene contains a 3. Currently, I have only made 3 levels for each year group section this will be soon changed to 6. I then use IndexOf ,which is a function that finds the index position of a character and will return –1 if not found. If it is not found, then it loads the next scene. In the build settings I put all the scenes next to each other so this will work. I then set sName to the current scene's name accessed by the levelIndex (sName and levelIndex are both static variables meaning that they will not be reset during a scene change).

```
0 references
public void NextLevel()
{
    Debug.Log(sName);
    if (sName.IndexOf("3") == -1)
    {
        SceneManager.LoadScene(levelindex + 1);
        levelindex += 1;
        Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
        sName = temp.name;
    }

}
```

I next checked to see if this works using the game complete screen and it did.

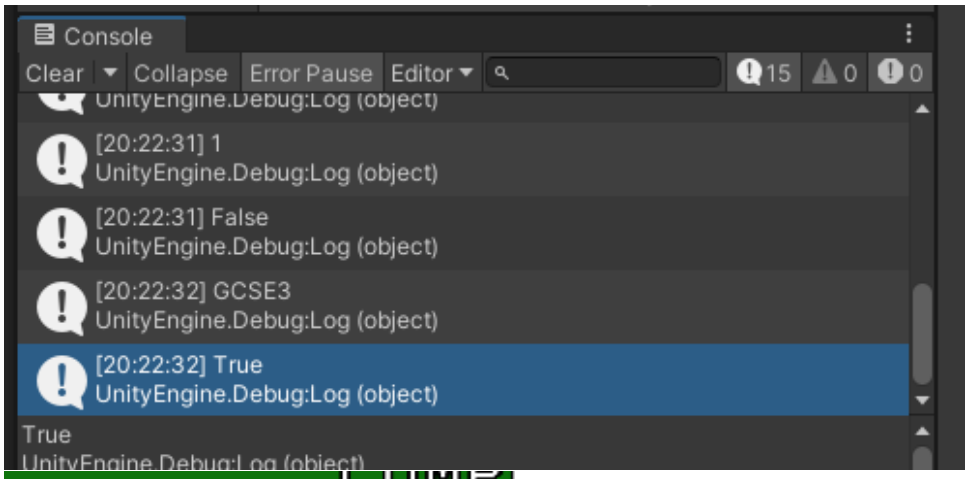


I next wanted to change the text of the screen so that it will restart at level 1 instead of saying next level when you have reached the end.

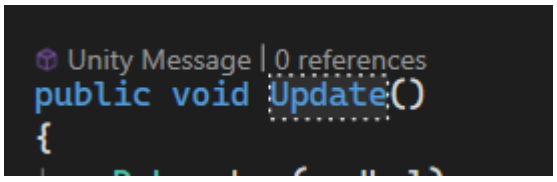
```
Unity Message | 0 references
public void Start()
{
    Debug.Log(endLvl);
    if(SceneManager.GetActiveScene().buildIndex == 14 && endLvl == true)
    {
        display.text = "It works";
    }
}
```

I added a start function which only executes at the first frame of the scene, to the screen director script so that it checks if endLvl is true which is set in the else statement in the nextLevel procedure and is set to false in the if statement.

However, whenever I checked to see if it worked endLvl was false and only turns true when I press the next level button, and only then does it change the button text



My first approach was to change the start void to an Update void which executes every frame.



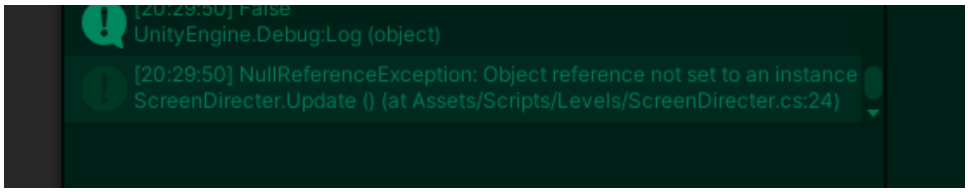
This however did not work so my next approach was to check if the current scene is the game complete scene, then if sName does contain a 3 by making sure it was not equal to –1 meaning that it has an index position. I then set endLvl to true and then the display text to It works.

```
Unity Message | 0 references
public void Update()
{
    Debug.Log(endLvl);
    if(SceneManager.GetActiveScene().buildIndex == 14)
    {
        Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
        sName = temp.name;
        if (sName.IndexOf("3") != -1)
        {
            endLvl = true;
            display.text = "It works";
        }
    }
}
```

After I removed the else statement from the NextLevel procedure

```
0 references
public void NextLevel()
{
    Debug.Log(sName);
    if (sName.IndexOf("3") == -1)
    {
        SceneManager.LoadScene(levelindex + 1);
        levelindex += 1;
        endLvl = false;
    }

}
}
```

However, whenever I pressed nextLevel it caused an error



This fixed the syntax error, but I still got a logic error as when pressing the button for game comeplete right before the last level the button will quickly flash with the display text

```
0 references
public void NextLevel()
{
    if (sName.IndexOf("3") == -1)
    {
        SceneManager.LoadScene(levelindex + 1);
        levelindex += 1;
        endLvl = false;
    }
    Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
    sName = temp.name;

}
```

To fix this I then changed the update procedure to be a Start procedure so that it only happens at the start of loading a level which fixed this issue

```
0 references
public void Start()
{
    Debug.Log(sName);
    if (SceneManager.GetActiveScene().buildIndex == 14)
    {
        if (sName.IndexOf("3") != -1)
        {
            endLvl = true;
            display.text = "It works";
        }
    }
}
```

I then changed the text below

```
public void Start()
{
    if (SceneManager.GetActiveScene().buildIndex == 14)
    {
        if (sName.IndexOf("3") != -1)
        {
            endLvl = true;
            display.text = "Restart from level 1";
        }
    }
}
```

I next added an else statement which would only occur at the end of the level and minus the current levelIndex by –2 so that it would restart at level 1

```
0 references
public void NextLevel()
{
    if (sName.IndexOf("3") == -1)
    {
        SceneManager.LoadScene(levelindex + 1);
        levelindex += 1;
        endLvl = false;
    }
    else
    {
        SceneManager.LoadScene(levelindex-2);
        levelindex -= 2;
    }
    Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
    sName = temp.name;

}
```

This now works and works fluidly just like my client wanted.

I next made the playerInfo script

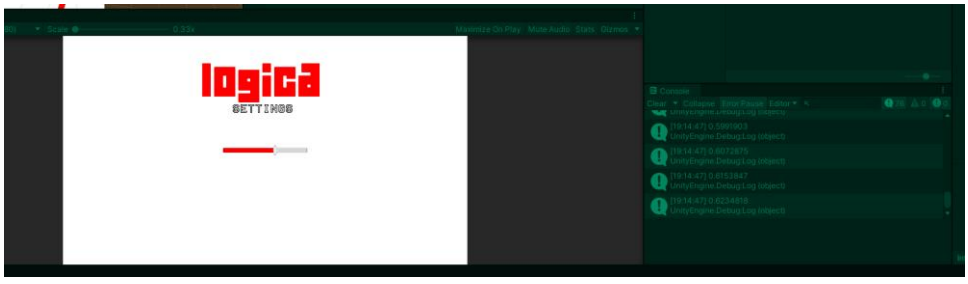
I originally intended for player settings to contain a player sensitivity slider along with a brightness slider.

This would be more accessible to someone with sensitive eyes for example however Unity does not have a feature where you can change the brightness unless you access the light in each scene which in 2D scenes I do not have this or the alternative way is to have an object which can be overlayed ontop of your screen however this will majorly increase memory usage so I ended up not doing that.

```
Unity Script | 0 references
public class PlayerInfo : MonoBehaviour
{
    public static float playerSens;
    public static float brightness;

    0 references
    public void sensitvity(Slider sensSlide)
    {
        playerSens = sensSlide.value;
        Debug.Log(playerSens);
    }
}
```

I first started off by checking to see how Unity's slider works and what values it outputs.

As I slide the bar across it gave me a value between 0 and 1 in decimals / float.



I did the same for the brightness slider as at this point, I still had intentions to make a brightness setting.

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

Unity Script (2 asset references) | 1 reference
public class PlayerInfo : MonoBehaviour
{
    public float playerSens;
    public static float playerBright;

    0 references
    public void sensitvity(Slider sensSlide)
    {
        playerSens = sensSlide.value;
        Debug.Log(playerSens);
    }

    0 references
    public void brightness(Slider brightSlide)
    {
        playerBright = brightSlide.value;
        Debug.Log(playerBright);
    }
}
```

I then tried to access playerSens within the FirstPersonCamera Script where sensitivity is used however, I cannot access static variables even if they are public

```
Cursor.lockState = CursorLockMode.Locked; // lock the cursor in the centre of
Debug.Log(gameObject.GetComponentInParent<PlayerInfo>().playerSens);
```

```
0
```

This is shown above in this image.

To fix this I made a getter method to return the current sensitivity selected by the slider

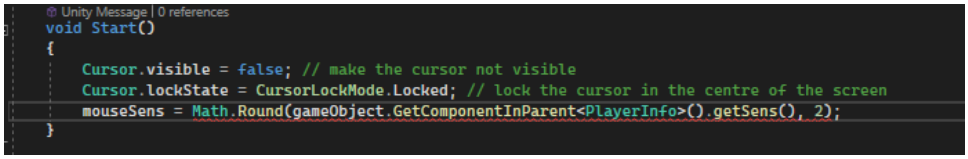
```
    1 reference
    public float getSens()
    {
        return tempSens;
    }
}
```

```
mouseSens = gameObject.GetComponentInParent<PlayerInfo>().getSens();
```

```
0.7791086
```

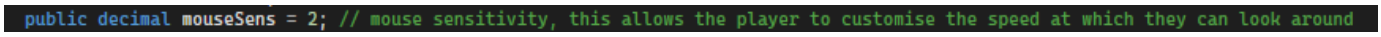
This retrieved the correct value.

However, this value does not look nice and is not suitable as even the tiniest change on the slider can affect the sensitivity on a miniscule level and for the future when I want to store the sensitivity, I want to make sure that the value does not take up too much storage.



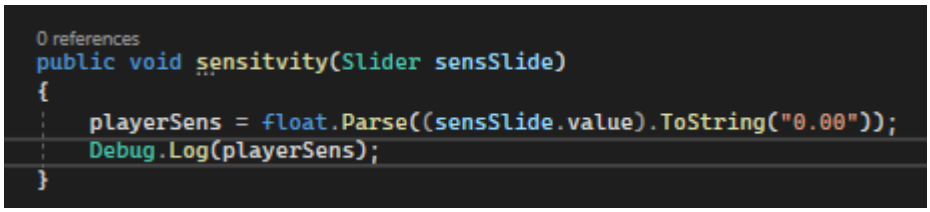
To fix this I first tried an attempt of trying to Round the retrieved value to 2 decimal places.

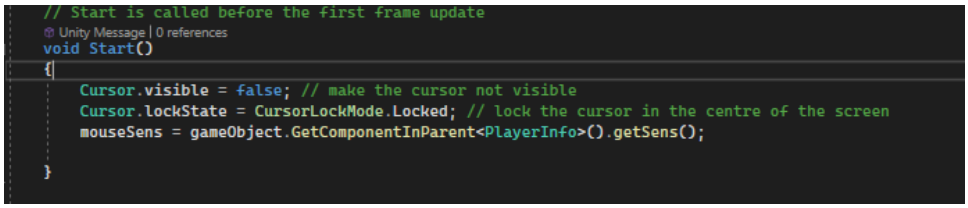
This did not work since you cannot round float values like this only decimal values.



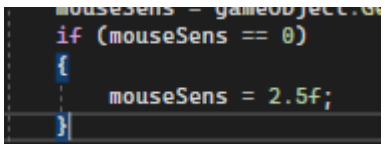
I then changed the mouseSens to be a decimal however this did not let me multiply the Y or X axis input from the mouse by the sensitivity since it was a decimal, so I had to approach this in a different way

My initial idea was to change the value to a 2dp format using a string and then change it back to a float





I then changed the FIrstPersonCamera script back to using .getSens() and not changing the value there



I then added an if statement that if mouseSens for some reason equals 0 to set it back to the default value.

However, I found a more efficient method by using RoundToInt function instead of changing it to a string

The reasoning is shown below,

However, in general, the `Mathf.RoundToInt(value / 100) * 100` approach is likely to be more performant as it involves basic arithmetic operations without the overhead of string formatting and parsing. Additionally, the `Mathf.RoundToInt` method is optimized for performance.

Now I implemented this new approach into my code. However, this round the value to a whole number

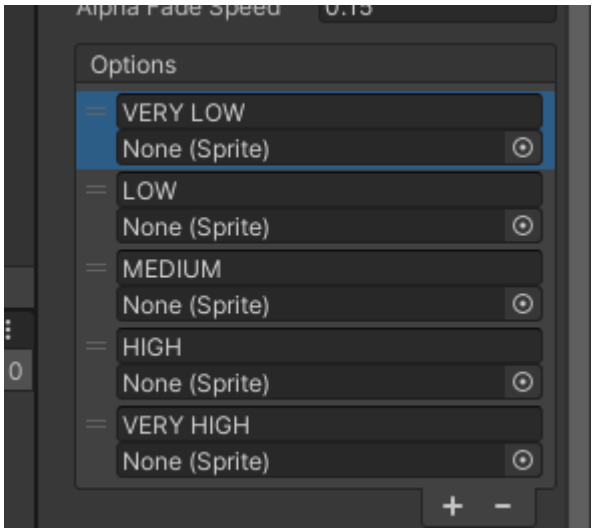
```
0 references
public void sensitvity(Slider sensSlide)
{
    playerSens = Mathf.RoundToInt(sensSlide.value * 100) / 100;
    Debug.Log(playerSens);
}
```

To fix this I used Mathf.Round instead of Math.Round. This method worked and is more optimised than using a string.

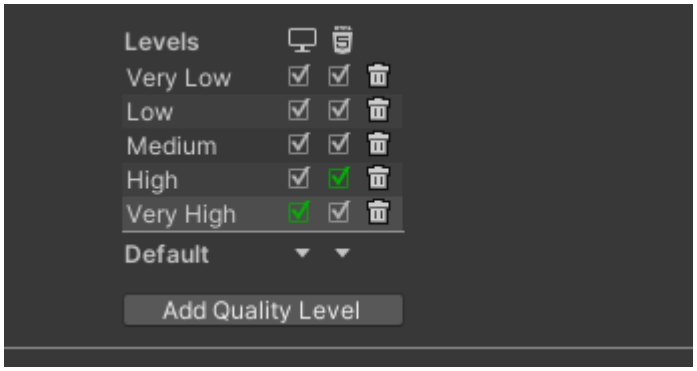
```
0 references
public void sensitvity(Slider sensSlide)
{
    playerSens = Mathf.Round(sensSlide.value * 100) / 100;
    Debug.Log(playerSens);
}
```

Since, brightness was not a settings option. I made a drop-down Menu for graphics quality.

I first set up the options



I also made sure to check the options matched the projects Unity quality levels for my specific project.



I then added a procedure within Screen Director to set the quality level to the index of the position of the selected drop-down Item.

```
0 references
public void setQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
}
```

I then in the start function for the player Info script made it so that the value of the slider changes to the player Sens.

```
Unity Message | 0 references
public void Start()
{
    if (gameObject.name == "Sensitivity")
    {
        gameObject.GetComponent<Slider>().value = playerSens;
    }
}
```

The code above did not work as the script was attached to the manager game object and not the sensitivity slider or drop-down menu, I changed my approach to add tags to the settings objects called settings.

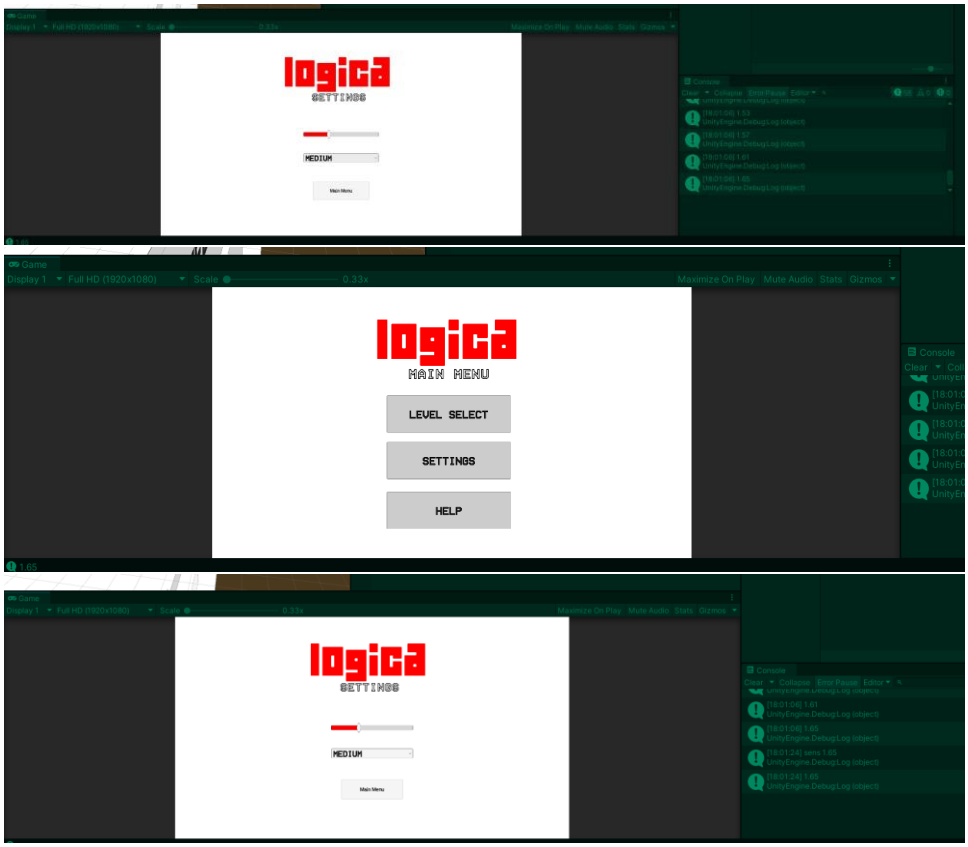
This wat I was able to find both game objects using the tag and get their values to be initialised to the static variables of the players current setting so it will be stored, and the player does not have to keep changing it back to their preference.

```
private void Start()
{
    if (gameObject.name == "Manager")
    {
        if (playerSens != 0)
        {
            Debug.Log("sens " + playerSens);
            GameObject[] gameObjects;
            gameObjects = GameObject.FindGameObjectsWithTag("settings");
            if (gameObjects.Length != 0)
            {
                gameObjects[0].GetComponent<Slider>().value = playerSens;
            }
        }
    }
}
```

I then Initialised the quality settings too.

```
gameObjects[1].GetComponent<TMP_Dropdown>().value = QualitySettings.GetQualityLevel();
```

Below are screenshots of me highlighting that the game stores the settings and initialises them back when you go to the settings screen again.



I now started to work on the pause menu.

I first made a procedure in Screen Director as this is what the Resume Button will use

This loads the scene of the levelIndex which is not updated when losing pause menu meaning it will load the previous scene.

```
0 references
public void gameContinue()
{
    SceneManager.LoadScene(levelindex);
}
```

The image below shows the pause screen following the design I made during the design phase of my project.

The code below is me implementing a way to check for if the escape key is pressed so that it can load the pause menu and get the build index of the current active scene and assign thay to level index so resume will always work.

```
Unity Message | 0 references
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        levelindex = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(16);
    }
}
```

However, when changing scenes from 3d level the mouse cursor is not there making it hard to see where your mouse is and where to click to fix this, I added cursor.visible equals true and set lock state equal to None. The problem was caused due to the 3D level turning off mouse cursor and setting the lock state to be at the centre of the screen so that the gameplay feels smooth.

```
Unity Message | 0 references
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        levelindex = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(16);
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }
}
}
```

I then set this before win is set to true so that my mouse is able to move in the game complete scene too

```
if (step == levels[level].gateOrder.Count)
{
    win = true;
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
}
}
```

I then created a new variable called backIndex which I will use for resuming certain scenes

```
Unity Script (18 asset references) | 0 references
public class ScreenDirecter : MonoBehaviour
{
    static Stack levelStack;
    public static int levelindex;
    public static int backindex;
    static string sName;
    public static bool endLvl;
    public TextMeshProUGUI display;
```

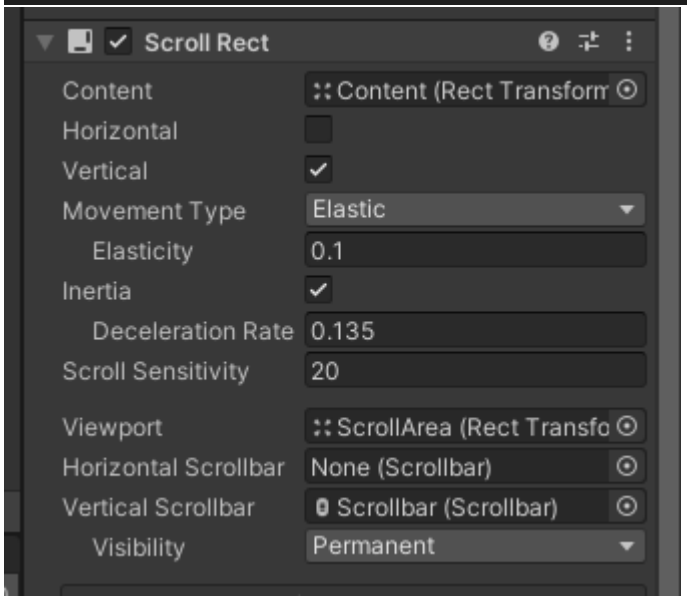
I first started to add two of statements for when this will be used. The first occasion will be when the student is playing a level and presses escape to open the pause menu and then presses the sweeting's menu to access their settings. I need the player to be able to go back to the pause menu and not the main menu since the other occasion is using the back button to access main menu instead of pause menu.

To do this in the first if statement I set backindex to equal the index of the current active scene after comparing the index to make sure it's the correct one.

After I made a procurer called gameBack which uses backindex to go back to the previous scene while still storing the levelindex which is used for resuming in the pause menu.

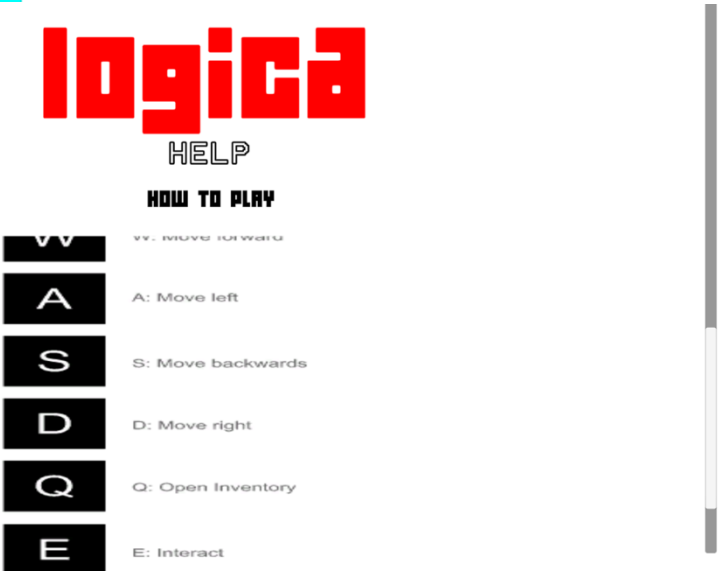
```
0 references
public void Load(Button button)
{
    SceneManager.LoadScene(Int32.Parse(button.name));
    levelindex = Int32.Parse(button.name);
    Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
    sName = temp.name;
    if (SceneManager.GetActiveScene().buildIndex == 16)
    {
        backindex = 16;
    }
    else if (SceneManager.GetActiveScene().buildIndex == 13)
    {
        backindex = 13;
    }
    //levelStack.Push(Int32.Parse(button.name));
}

0 references
public void gameBack()
{
    SceneManager.LoadScene(backindex);
}
```

```
0 references
public void Load(Button button)
{
    SceneManager.LoadScene(Int32.Parse(button.name));
    levelindex = Int32.Parse(button.name);
    Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
    sName = temp.name;
    if (SceneManager.GetActiveScene().buildIndex == 16)
    {
        levelindex = 16;
    }
    else if (SceneManager.GetActiveScene().buildIndex == 13)
    {
        levelindex = 13;
    }
    //levelStack.Push(Int32.Parse(button.name));
}
```



I next started to work on the help menu. To do this I first started by attaching a Scroll Rect to make the page scrollable and then made a scroll bar and set that to the vertical scroll bar in the scroll rect.

This linked the scroll bar to the Scroll Rect allowing me to scroll using the page and the scroll bar.



The two images above show my help screen

However, I realised that my code sometimes still messed up as levelIndex was still set to the button name, so I included it in the selection process of the code. It first checks if it needs to use the backIndex conditions else, levelIndex is set.

```
0 references
public void Load(Button button)
{
    SceneManager.LoadScene(Int32.Parse(button.name));
    if (SceneManager.GetActiveScene().buildIndex == 16)
    {
        backindex = 16;
    }
    else if (SceneManager.GetActiveScene().buildIndex == 13)
    {
        backindex = 13;
    }
    else
    {
        levelindex = Int32.Parse(button.name);
        Scene temp = SceneManager.GetSceneByBuildIndex(levelindex);
        sName = temp.name;
    }
}
```

After further consultation, my client is content with the menus.

Databases

I first had to use my database design to know what fields and fields my tables need when creating them

Sign up and Log In

I made a script called teacher where everything that is accessed by the teacher will be stored. This will include adding, removing and the leader board section along with signing up and logging in.

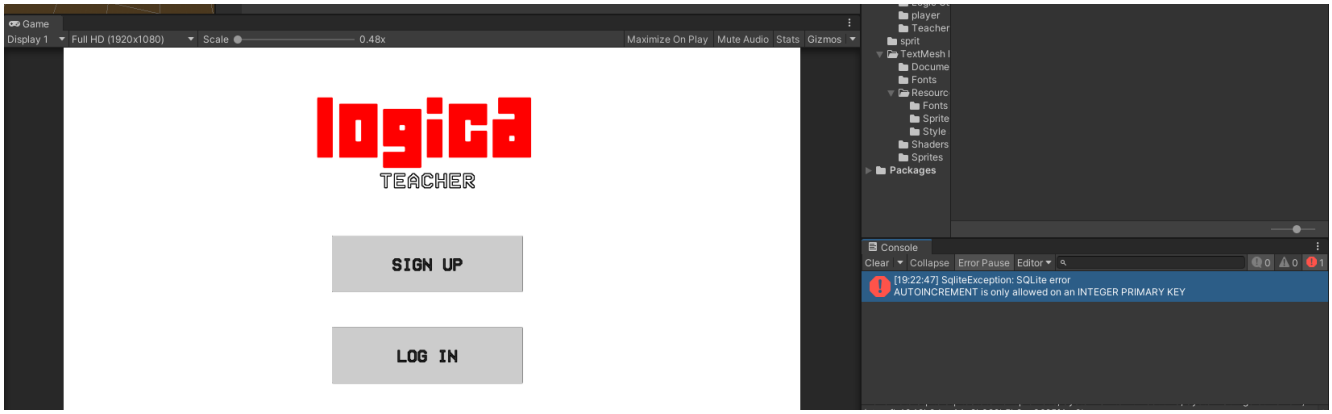
I first connect to dbName which is a database file that will store all my tables in one place. This is more efficient as I can access all the databases using less storage and less code. It will also be easier to test and check if databases are working correctly due to being able to access all the tables by opening a single file.

I then connected to sqlite3 using dbName and then executed the sql statement where I create the teacher database using the fields that I designed in the database design section.

I made the TeacherID autoincrement so that I will not have to keep track of the variable and make sure it is unique. This saves time coding and increases efficiency

```
0 references
public void Createdb()
{
    //Creates a new Sqlite database
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Creates the table with the fields Username and Password which both have a max input length of 20 characters
            command.CommandText = "CREATE TABLE IF NOT EXISTS Teacher(TeacherID INT, Username VARCHAR(20), Password VARCHAR(20),PRIMARY KEY (TeacherID AUTOINCREMENT));";
            command.ExecuteNonQuery();
        }
        connection.Close();
    }
```

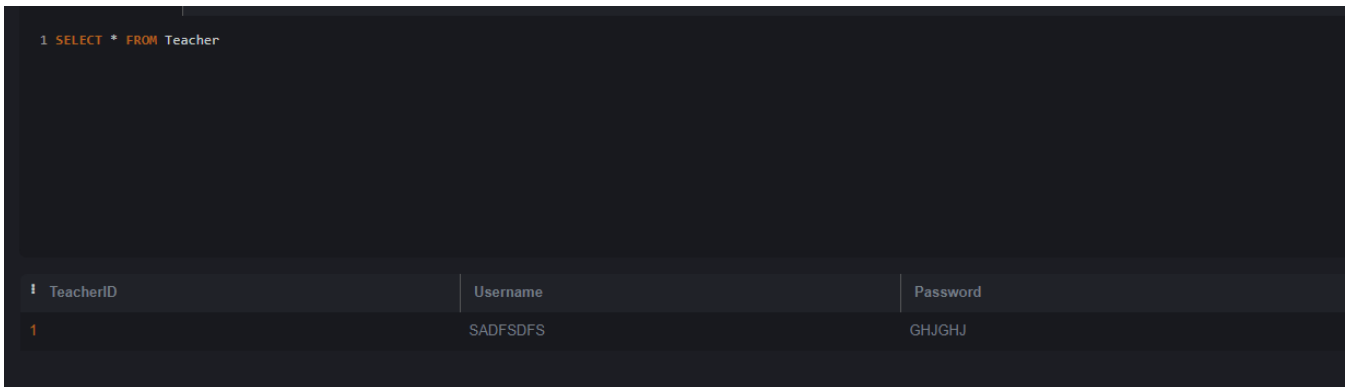
However, this game me an error as I did not set TeacherID to the correct term when defining as integer since I used ‘INT’ and not ‘INTEGER’.



I quickly fixed this and then tested my code. I attached the CreateDB procedure whenever I pressed the teacher button in the start menu to access the scene above.

```
public void Createdb()
{
    //Creates a new Sqlite database
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Creates the table with the fields Username and Password which both have a max input length of 20 characters
            command.CommandText = "CREATE TABLE IF NOT EXISTS Teacher(TeacherID INTEGER, Username VARCHAR(20), Password VARCHAR(20),PRIMARY KEY (TeacherID AUTOINCREMENT));";
            command.ExecuteNonQuery();
        }
        connection.Close();
    }
}
```

As you can see blow, the code worked when I viewed my database using SQLite online and importing my database file.



I next created a new scene for the teacher to sign up. Then I created a procedure to access all the inputs and made validation to make sure the password is the one that they wanted to type in.

```
//Retrieves the entered textfrom the sign up page
0 references
public void RetrieveFromSignUp()
{
    username = SusernameINP.text;
    password = SpasswordINP.text;
    confirmpassword = SconfirmpasswordINP.text;

    if (password == confirmpassword)
    {
        proceed = true;
    }
    else
    {
        submitStext.text = "Passwords did not match";
    }
}
```

However, my approach was wrong at first and instead decided to make it check for if the user has done the wrong thing and not the correct thing. This will make checking things easier and hence making the code more efficient. I added validation to test if the inputs were not put in and also if the password does not match.

```
//Retrieves the entered textfrom the sign up page
0 references
public void RetrieveFromSignUp()
{
    username = SusernameINP.text;
    password = SpasswordINP.text;
    confirmpassword = SconfirmpasswordINP.text;

    if (username == "" || password == "" || confirmpassword == "")
    {
        submitStext.text = "Fill in all inputs";
    }
    else if(password != confirmpassword)
    {
        submitStext.text = "Passwords did not match";
    }
    else
    {
        proceed = true;
    }
}
```

Below is a screenshot of me testing if the code works and it does.

I next added more validation such as making the password include a special character and making sure it has more than 8 characters.

```
//Retrieves the entered textfrom the sign up page
0 references
public void RetrieveFromSignUp()
{
    username = SusernameINP.text;
    password = SpasswordINP.text;
    confirmpassword = SconfirmpasswordINP.text;


    if (username == "" || password == "" || confirmpassword == "")
    {
        submitStext.text = "Fill in all inputs";
    }
    else if(password.Length < 8)
    {
        submitStext.text = "Passwords is too short";
    }
    else if (HasSpecialChars(password) == false)
    {
        submitStext.text = "Password does not have a special character";
    }
    else if(password != confirmpassword)
    {
        submitStext.text = "Passwords did not match";
    }
    else
    {
        proceed = true;
    }
}
```

Below is the Boolean function I added to check if the passed in parameter contains a special character. This is done by checking whether ‘ch’ is not a letter or digit, hence meaning it is a special character which is the condition to be checked for the password.

```
1 reference
bool HasSpecialChars(string password)
{
    return password.Any(ch => !char.IsLetterOrDigit(ch));
}
```

I then tested the new validation checks in the code below and it works as intended.

I now started the login scene where I created a procedure to check if their login details are correct. Furthermore, since this is my first-time using SQLite in Unity, I had to test to see how the reader.Read function worked

```
connection.Open();
using (var command = connection.CreateCommand())
{
    //Inserts the new values into the database
    command.CommandText = command.CommandText = "SELECT * FROM Teacher WHERE Username = '" + username + "'AND Password ='" + password + "';";
    command.ExecuteNonQuery();
    using (IDataReader reader = command.ExecuteReader())
    {
        //Every line the SQL code outputs is read
        while (reader.Read())
        {
            //Stores password stored in the databases
            object v = reader["Password"];
            readPassword = v.ToString();
        }
        connection.Close();
    }
}

ebug.Log(readPassword);
```

I checked to see if see if it debugged the correct password.

Next, I coded the retrieving from login in which I assigned new variables for the input boxes so that it will be easier to differentiate what variable is for what object.

```
0 references
public void RetrieveFromLogIn()
{
    username = LusernameINP.text;
    password = LpasswordINP.text;

    if (username == "" || password == "")
    {
        submitStext.text = "Fill in all inputs";
    }
    else
    {
        proceed = true;
    }
}
```

After I checked if the login section worked for the teacher, I moved onto creating the student script. I will reuse code from the teacher script however I will have to adjust this to suit my database design for the student table and my design on what I will have to use the database for when the student logs in or signs up.

I first started off by creating a student database using my database design in the process.

StudentID will autoincrement after every new student added.

Username and password are so that the user can login. Sensitivitity is a float value that is rounded to 2dp. Graphics will be the index position for the graphics drop down and the graphics quality in build settings

```
27      0 references
28      public void Createdb()
29      {
30          //Creates a new Sqlite database
31          using (var connection = new SqliteConnection(dbName))
32          {
33              connection.Open();
34              using (var command = connection.CreateCommand())
35              {
36                  //Creates the table with the fields Username and Password which both have a max input length of 20 characters
37                  command.CommandText = "CREATE TABLE IF NOT EXISTS Student(StudentID INTEGER, Username VARCHAR(20), Password VARCHAR(20),Sensitivity FLOAT, Graphics INTEGER, PRIMARY KEY (StudentID AUTOINCREMENT));";
38                  command.ExecuteNonQuery();
39              }
40              connection.Close();
41          }
```

I then reused the retrieve signup code, changing the variable names to suit the script for maintenance.

```
//Retrieves the entered textfrom the sign up page
0 references
public void RetrieveFromSignUp()
{
    username = SusernameINP.text;
    password = SpasswordINP.text;
    confirmpassword = SconfirmpasswordINP.text;


    if (username == "" || password == "" || confirmpassword == "")
    {
        submitStext.text = "Fill in all inputs";
    }
    else if (password.Length < 8)
    {
        submitStext.text = "Passwords is too short";
    }
    else if (HasSpecialChars(password) == false)
    {
        submitStext.text = "Password does not have a special character";
    }
    else if (password != confirmpassword)
    {
        submitStext.text = "Passwords did not match";
    }
    else
    {
        proceed = true;
```

No issues found

I then reused previous code from the teacher script below and adjusted it to fit the student database design and student screen design etc. I also changed variable names to suit the student script more so that it is more maintainable and has better readability.

```
0 references
public void RetrieveFromLogIn()
{
    username = LusernameINP.text;
    password = LpasswordINP.text;

    if (username == "" || password == "")
    {
        submitStext.text = "Fill in all inputs";
    }
    else
    {
        proceed = true;
    }
}

//Adds a new player into the PlayerInfo table
0 references
public void AddStudent()
{
    if (proceed)
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Student(Username, Password, Sensitivity, Graphics) " +
                    "VALUES ('" + username + "','" + password + "','" + 2.5 + "','" + 0 + "');";
                command.ExecuteNonQuery();
            }
            connection.Close();
        }
    }
}

0 references
public void CheckStudent()
{
    if (proceed)
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
```

I also reused the checkTeacher procedure from the teacher script however I change the SQL query to suit what I will be retrieving

```
0 references
public void CheckStudent()
{
    if (proceed)
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = command.CommandText = "SELECT * FROM Student WHERE Username = '" + username + "'AND Password ='" + password + "';";
                command.ExecuteNonQuery();
                using (IDataReader reader = command.ExecuteReader())
                {
                    //Every line the SQL code outputs is read
                    while (reader.Read())
```

I retrieved the sensitivity and graphics of the player that is logged in and assigned the retrieved data to the static variables.

```
//Stores the number of coins stored in the
object v = reader["Sensitivity"];
readSens = (float)v;
object c = reader["Graphics"];
readGFX = Convert.ToInt32(c);
```

This will be used to initialise the students' settings when they log in

After further consultation, my client is content with the functionality of the sign up and log in process.

Using the Database for Students

Retrieving player settings

I first create a new procedure for initialising sensitivity in the playerInfo so that the static variables are equal to the retrieved data from the database

```
0 references
public void intialiseSettings()
{
    gameObject.GetComponent<PlayerInfo>().setGraphics(readGFX);
    gameObject.GetComponent<PlayerInfo>().setSens(readSens);
}
```

I created the setter methods to set player sensitivity in the playerInfo script and set the player graphics quality in the same script aswell.

```
1 reference
public void setSens(float sens)
{
    playerSens = sens;
}

0 references
public void setGraphics(int i)
{
    QualitySettings.SetQualityLevel(i);
}
```

However, when I was reading from the database it did not work since readSens is local to only the procedure that is reading the database is not local to the script since I have another readSens variable defined outside the procedure

```
command.ExecuteNonQuery();
using (IDataReader reader = command.ExecuteReader())
{
    //Every line the SQL code outputs is read
    while (reader.Read())
    {
        //Stores the settings stored in the databases
        object v = reader["Sensitivity"];
        float readSens = (float)v;
        object c = reader["Graphics"];
        readGFX = Convert.ToInt32(c);
    }
    connection.Close();
}
```

My first approach was to parse the v object however it did inot work since parse only works for string variables.

```
{
    //Stores the settings stored in the databases
    object v = reader["Sensitivity"];
    readSens = float.Parse(v);
    object c = reader[
    readGFX = Convert.
}
connection.Close();
```

(local variable) object v

CS1503: Argument 1: cannot convert from 'object' to 'string'

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

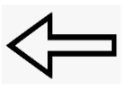
To fix this I changed the v object to a string and then parsed it.

```
    while (reader.Read())
    {
        //Stores the settings stored in the databases
        object v = reader["Sensitivity"];
        readSens = float.Parse(v.ToString());
        object c = reader["Graphics"];
        readGFX = Convert.ToInt32(c);
    }
    connection.Close();
}
```

Next to test my code I changed the insert variables to be different than the default values so I will be able to view the changes made when going into settings

```
"VALUES ('" + username + "','" + password + "','" + 3 + "','" + 1 + "');";
```

However, when I checked the sensitivity was updated from the player database but not the graphics



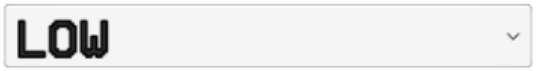
To fix this I made the drop-down value to change to the static playerGraphics variable that I created which is an integer for the quality settings.

```
if (gameObjects.Length != 0)
{
    gameObjects[0].GetComponent<Slider>().value = playerSens;
    gameObjects[1].GetComponent<TMP_Dropdown>().value = playerGraphics;
}
```

I then changed the setGraphics to also change the playerGraphics variable to I which is the paramater passed in by the student script

```
1 reference
public void setGraphics(int i)
{
    QualitySettings.SetQualityLevel(i);
    playerGraphics = i;
}
```

After testing this it works as intended



I then changed the insert variables back to the default

Updating player settings

I started on updating the database values whenever the quality or settings are changed.

For the settings inside the procedure that sets the static playerSens variable to the slider value and rounds it to two. I next access the student script and the update playerSens by passing in the playerSens

```
0 references
public void sensitvity(Slider sensSlide)
{
    playerSens = Mathf.Round(sensSlide.value * 100) / 100;
    gameObject.GetComponent<student>().UpdateSens(playerSens);
}

1 reference
```

I did the same thing but changed the called procedure to UpdateGraphics and passing in the quality index.

```
0 references
public void setQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
    gameObject.GetComponent<student>().UpdateGraphics(qualityIndex);
}
```

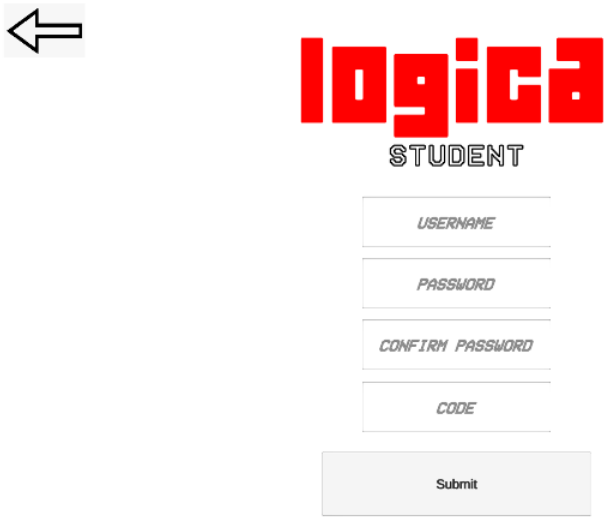
Within the UpdateSens procedure I connected to my database file containing all my tables and updated the sensitivity field within my database where it is the current logged in user with the passed in parameter.

```
1 reference
public void UpdateSens(float sens)
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Updates the new values
            command.CommandText = "UPDATE Student SET Sensitivity = '" + sens + "'  WHERE Username = '" + username + "'AND Password ='" + password + "';";
            command.ExecuteNonQuery();
        }
        connection.Close();
    }
}
```

Within the UpdateGraphics procedure I connected to my database file containing all my tables and updated the graphics field within my database where it is the current logged in user with the passed in parameter.

```
1 reference
public void UpdateGraphics(int gfx)
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Updates the new values
            command.CommandText = "UPDATE Student SET Graphics = '" + gfx + "'  WHERE Username = '" + username + "'AND Password ='" + password + "';";
            command.ExecuteNonQuery();
        }
        connection.Close();
    }
}
```

This is the student sign up page where I will test this.



I also added the code variable for the student to join the correct class. I also added this into the insert into database code for the AddStudent procedure

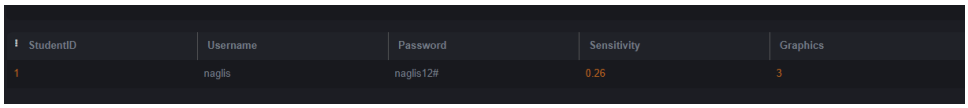
```
//Inserts the new values into the database
command.CommandText = "INSERT INTO Student(ClassID,Username, Password, Sensitivity, Graphics) " +
    "VALUES ('" + code + "','" + username + "','" + password + "','" + 2.5 + "','" + 0 + "');";
command.ExecuteNonQuery();
```

I next added validation for the code making sure it is a maximum and minimum of 4 digits, I also tested to make sure that the entered code has numbers entered.

```
//Retrieves the entered textfrom the sign up page
0 references
public void RetrieveFromSignUp()
{
    username = SusernameINP.text;
    password = SpasswordINP.text;
    confirmpassword = SconfirmpasswordINP.text;
    codeSTR = ScodeINP.text;
    if (username == "" || password == "" || confirmpassword == "" || codeSTR == "")
    {
        submitStext.text = "Fill in all inputs";
    }
    else if (password.Length < 8)
    {
        submitStext.text = "Passwords is too short";
    }
    else if (HasSpecialChars(password) == false)
    {
        submitStext.text = "Password does not have a special character";
    }
    else if (password != confirmpassword)
    {
        submitStext.text = "Passwords did not match";
    }
    else if(!(int.TryParse(codeSTR,out code)))
    {
        submitStext.text = "Code should be a number";
    }
    else if (codeSTR.Length > 4 || codeSTR.Length < 4)
    {
        submitStext.text = "Code is not 4 digits";
    }
    else
    {
        proceed = true;
    }
}
```

I tested to make sure that when you logged in the settings save. Furthermore, that the settings update within the database when you change your settings.

Below is a screenshot of changing the variables for my settings and they get updated within the database.

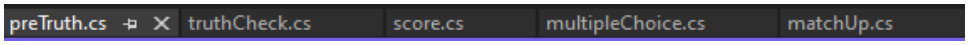


Note that the this was before I deleted the database file and resetting the student table to include a class code.

Level Progress

```
public int LevelNum;
```

First, I added a public LevelNum integer to each of the scripts below



This is so that I can use the LevelNum integers to distinguish levels in the LevelProgress table.

First, I created the LevelProgress table referencing my database design in the design section and made a composite key consisting of StudentID and LevelNum. This will be created if it does not exist when the student presses level selection button.

Score will be the score that the player achieved from the game

```
0 references
public void Createdb2()
{
    //Creates a new Sqlite database
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Creates the table with the fields Username and Password which both have a max input length of 20 characters
            command.CommandText = "CREATE TABLE IF NOT EXISTS LevelProgress(StudentID INTEGER, LevelNum INTEGER, Score VARCHAR(20), PRIMARY KEY (StudentID, LevelNum));";
            command.ExecuteNonQuery();
        }
        connection.Close();
    }
}
```

Next, I went into the student script and changed the insert code of AddStudent

I made sure to add the insertion of classID and the code that the student will type in.

```
0 references
public void AddStudent()
{
    Debug.Log(proceed);
    if (proceed)
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Student(ClassID,Username, Password, Sensitivity, Graphics) " +
                    "VALUES ('" + code + "','" + username + "','" + password + "','" + 2.5 + "','" + 0 + "');";
                command.ExecuteNonQuery();
```

Next I added some validation for password, as it still makes the user go to the main menu or teacher menu despite not actually having a correct password for a user.

```
0 references
public void MainMenu()
{
    if (proceed)
    {
        SceneManager.LoadScene(22);
    }
}
```

To fix this I checked if readID is 0 which is what the database retrieves from the SQL statement when it is executed, and no password ID fits the password or username entered.

I added this to both the teacher and student script

```
                readSens = float.Parse(v.ToString());
                object c = reader["Graphics"];
                readGFX = Convert.ToInt32(c);
                object x = reader["StudentID"];
                readID = Convert.ToInt32(x);
                Debug.Log(readID);
            }
            connection.Close();
        }
    }
    if (readID == 0)
    {
        submitStext.text = "Incorrect Password";
        proceed = false;
    }
}
```

I also added a separate procedure in this student and teacher script to go to their corresponding menus since they must check if proceed is true or not false.

Next, I made the studentScore procedure which will either insert if it is the students first time playing the level or update if it is the students are playing it again to try get a better score.

I first started off by creating the void and passing in the score and level. The score will be the finalscore from that level and the level will be the LevelNum which is the new variable that I added in all my score/ level checking scripts.

I next connected to the database file called dbName

Then, I selected all the data from LevelProgress where the readID which is the studentID since this is the student script and where LevelNum is equal to the parameter that was passed in called level.

I then retrieve the score from the database for this student and for this level and assign this to readScore.

I then check if the score is empty. If so, then I insert the students ID, level number and their score into the table in the corresponding fields.

However, I then convert score and readScore into integers and compare them. If the paramater score is more than this replaces the current score which is readScore and updates the database.

I update the database by setting score to the score parameter where studentID is the readID and LevelNum is equal to level

```
0 references
public void studentScore(string score, int level)
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //executes sql query
            command.CommandText = command.CommandText = "SELECT * FROM LevelProgress WHERE Username = '" + readID + "'AND LevelNum ='" + level + "';";
            command.ExecuteNonQuery();
            using (IDataReader reader = command.ExecuteReader())
            {
                //Every line the SQL code outputs is read
                while (reader.Read())
                {
                    // Reads the current score stored
                    object v = reader["Score"];
                    readScore = v.ToString();
                }
                connection.Close();
            }
            // if it is empty it inserts a new entry to the database
            if (score == "")
            {
                command.CommandText = "INSERT INTO LevelProgress(StudentID,LevelNum, Score) " +
                   "VALUES ('" + readID + "','" + level + "','" + score + "');";
                command.ExecuteNonQuery();
            }
            // if new score is larger than the stored database it updates it
            else if (Convert.ToInt32(score) > Convert.ToInt32(readScore))
            {
                //Updates the new values
                command.CommandText = "UPDATE LevelProgress SET Score = '" + score + "'  WHERE StudentID = '" + readID + "'AND LevelNum ='" + level + "';";
                command.ExecuteNonQuery();
            }
        }
        connection.Close();
    }
}
```

Next in each of the same scripts where I added the LevelNum variable, I then added this line of code before it loads the game complete scene.

This called the studentScore procedure passing in finalScore as a String and LevelNum.

```
gameObject.GetComponent<student>().studentScore(finalScore.ToString(), LevelNum);
SceneManager.LoadScene(23);
```

However, I realised since I copied my code, I forgot to change the part where it says ‘WHERE Username =’

I am not trying to check if the username is the same as readID so I changed it to StudentID as seen below

```
//executes sql query
command.CommandText = command.CommandText = "SELECT * FROM LevelProgress WHERE StudentID = '" + readID + "'AND LevelNum ='" + level + "';";
command.ExecuteNonQuery();
using (IDataReader reader = command.ExecuteReader())
```

There was still a problem when I ran the code. This was due to when comparing the score to readScore the Convert.ToInt32 converts the score within the else if statement and never changes it back to string.

```
[13:04:26] FormatException: Input string was not in a correct format.
System.Number.StringToNumber (System.String str, System.Globalization.NumberStyles options, System.Number+NumberBuffer
```

To fix this I made score be passed as an integer

```
5 references
public void studentScore(int score, int level)
{
```

I only converted score to a string when it was being put inside the database.

```
}
// if it is empty it inserts a new entry to the database
if (readScore.ToString() == "")
{
    command.CommandText = "INSERT INTO LevelProgress(StudentID,LevelNum, Score) " +
       "VALUES ('" + readID + "','" + level + "','" + score.ToString() + "');";
    command.ExecuteNonQuery();
}
// if new score is larger than the stored database it updates it
else if (score > Convert.ToInt32(readScore))
{
    //Updates the new values
    command.CommandText = "UPDATE LevelProgress SET Score = '" + score.ToString() + "'  WHERE StudentID = '" + readID + "'AND LevelNum ='" + level + "';";
    command.ExecuteNonQuery();
}
```

string string.operator +(string left, string right)

However the code gave me an error that the database is not open

```
[13:08:46] InvalidOperationException: Database is not open
Mono.Data.Sqlite.SqliteCommand.InitializeForReader () (at <afb4049b8dec44e9b329b5b9ee3695f4>:0)
```

This is because I closed it after I read the data from the database, to fix this I removed 'connection.Close()'

```
using (IDataReader reader = command.ExecuteReade
{
    //Every line the SQL code outputs is read
    while (reader.Read())
    {
        // Reads the current score stored
        object v = reader["Score"];
        readScore = v.ToString();
    }
}
// if it is empty it inserts a new entry to the
if (readScore.ToString() == "")
{
    command.CommandText = "INSERT INTO LevelProg
       "VALUES ('" + readID + "','" + level + "'
```

Next, I changed the reader to StudentID since I had accidentally typed in Graphics.

```
readGFX = Convert.ToInt32(c);
object x = reader["StudentID"];
readID = Convert.ToInt32(x);
```

After this change, I decided to test the database

The database kept giving me 0 as studentID despite autoincrement always starting at 1. My intial thought as to why this was because I forgot to set readID to static and integers are always initialised at 0

StudentID	LevelNum	Score
0	0	568
0	1	582

I then changed readID to static

```
public static int readID;
public string readScore;
```

When I tested the database, the previous results were still there but this time it retrieved the correct ID (studentID 2)

StudentID	LevelNum	Score
0	0	568
0	1	582
2	2	555

After further consultation, my client is content with the implementation and making use of databases within the program for the student data.

Using the Database for Teachers

Add a Class

I first started off by creating the database when the teacher presses the add Class button in the teacher menu

This will create a class with classID as the primary key, teacherID as a foreign key and class name as a field that teachers will be able to distinguish their classes by.

```
0 references
public void Createdb2()
{
    //Creates a new Sqlite database
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Creates the table with the fields Username and Password which both have a max input length of 20 characters
            command.CommandText = "CREATE TABLE IF NOT EXISTS Class(ClassID INTEGER, TeacherID INTEGER, className VARCHAR(20),PRIMARY KEY (ClassID AUTOINCREMENT));";
            command.ExecuteNonQuery();
        }
        connection.Close();
    }
}
```

I next added code to read the database and obtain the current teachers ID from the teachers table whenever they login or sign up to use when creating a class.

```
//Every time the SQL code outputs is read
while (reader.Read())
{
    //Stores password stored in the database
    object v = reader["TeacherID"];
    readID = Convert.ToInt32(v);
}
connection.Close();
```

Next, I started to make the code for teachers to add classes. I started off by connecting to the database so that I will be able to execute the insert SQL query. Next, I executed the SQL query to execute teacherID and the class name obtained by the static readID variable and the input boxes text.

```
0 references
public void AddClass()
{

        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Class(TeacherID, className) " +
                    "VALUES ('" + readID + "','" + classNameINP.text + "');";
                command.ExecuteNonQuery();
            }
            connection.Close();
        }

}
```

I next worked on creating the adding a class screen where I came up with this and took inspiration from my screen designs in my design stage.

I first started to work on generating the classID code

To do this I used the inbuilt random function from system and generated a number between 0 to 10,000. I then formatted this using the ToString function so that it will be 4 digits and if it's a number such as 38 it will lead with 0s and make 0038.

I then debugged digits and tested to see if this worked

```
0 references
public void Generate()
{
    System.Random rng = new System.Random();
    int number = rng.Next(0,10000);
    string digits = number.ToString("0000");
    Debug.Log(digits);
}
```

As show below by the debug it generates a random number

I next changed the text for the code on screen show that it shows the generated number

```
string digits = number.ToString("0000");
codeText.text = digits;
```

I then added the classID when creating the database

```
(ClassID VARCHAR(4),
```

In the AddClass() procedure I changed the code so that it now also inserts the code into ClassID

The variable ‘code’ is a static variable used and is assigned to be the same as digits.

```
//Inserts the new values into the database
command.CommandText = "INSERT INTO Class(ClassID,TeacherID, className) " +
    "VALUES ('" + code + "','" + readID + "','" + classNameINP.text + "');";
command.ExecuteNonQuery();
```

When testing and viewing the database, the code is working as expected

ClassID	TeacherID	className
8025	1	13MW

TeacherID	Username	Password
1	flain	flain12##

Remove a Class

Next, I worked on the removing a class feature. For example, if you a class leaves there is no need to store that data and clunk up the drop-down menus in the leaderboard.

I first made a drop down set up procedure.

The procedure firsts clear the options of the drop-down menu called ‘deleteClass’.

I then access the database stored in dbName and select all the className’s that belong to the currently logged in teacher. This is accessed via a where statement by finding all the className’s where TeacherID field is equal to readID, a static variable that was assigned when the teacher signed in / logged in.

Next, I coded it so that it reads the database after the executed query and adds the classes into a classOptions list.

This is list is then added to the deleteClass drop down menu.

```
1 reference
public void DropDownSetUp()
{
    deleteClass.ClearOptions();
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Inserts the new values into the database
            command.CommandText = command.CommandText = "SELECT className FROM Class WHERE TeacherID = '" + readID + "';";
            command.ExecuteNonQuery();
            using (IDataReader reader = command.ExecuteReader())
            {
                //Every line the SQL code outputs is read
                while (reader.Read())
                {
                    //Stores password stored in the databases
                    object v = reader["className"];
                    classOptions.Add(v.ToString());
                }
                connection.Close();
            }
        }
    }
    deleteClass.AddOptions(classOptions);
}
```

I called the DropDownSetUp in the start function so that it will initialise the menu at the first frame

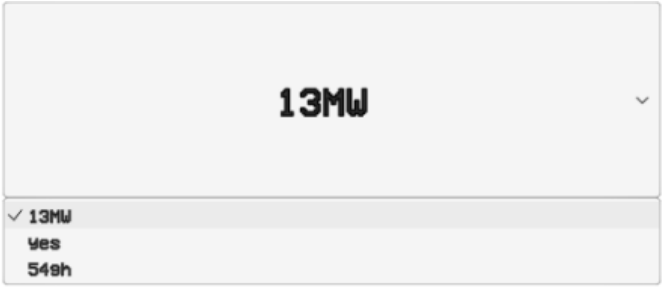
```
Unity Message | 0 references
public void Start()
{
    DropDownSetUp();
}
```

However, there was a problem where it will try to initialise the drop-down menu in a scene where there is no drop-down Menu since this script is used in multiple scenes.

To fix this I checked if the buildIndex is the same as the remove a Class scene.

```
Unity Message | 0 references
public void Start()
{
    if (SceneManager.GetActiveScene().buildIndex == 36)
    {
        DropDownSetUp();
    }
}
```

Below is showing that the code is working.



Next, I must make the program delete the class from the database.

To do this I created a new procedure that gets the value of the current selected option from the drop-down menu and then gets the text of this value and assigns this to the selected string variable.

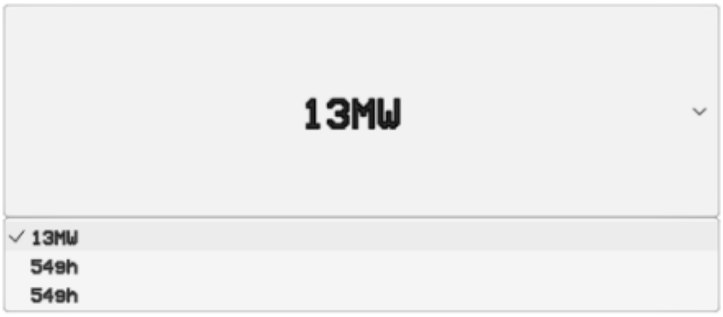
I then connect to the file that stores my database tables, dbName. Then I execute the deletion of the class where the className is equal to selected.

```
0 references
public void RemoveClass()
{
    string selected = deleteClass.options[deleteClass.value].text;
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Inserts the new values into the database
            command.CommandText = command.CommandText = "DELETE FROM Class WHERE className = '" + selected + "';";
            command.ExecuteNonQuery();
        }
    }
    DropDownSetUp();
}
```

I check to see if this has worked. I selected the class called yes. It deleted the class as shown in the database

ClassID	TeacherID	className
8025	1	13MW
7592	1	54gh

However once deleted the classes do not update within the scene.



To fix this I cleared classOptions before I recalled DropDownSetUp() in the removeClass procedure

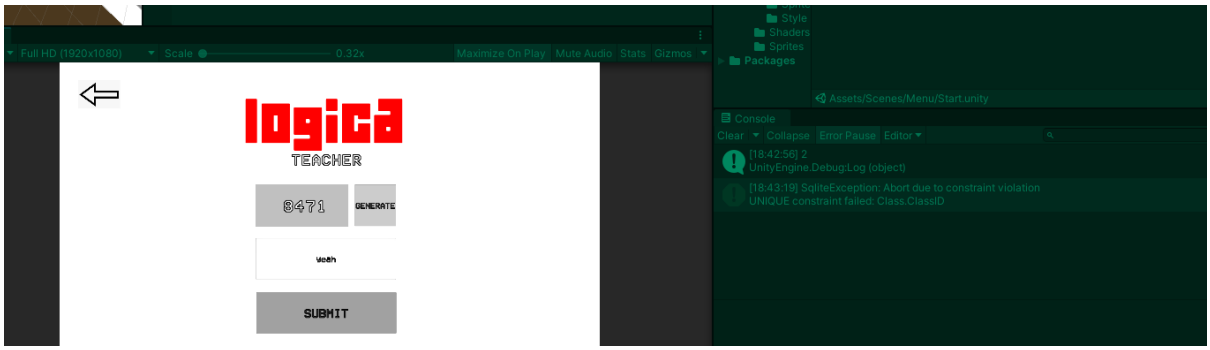
```
    }
}
classOptions.Clear();
DropDownSetUp();
```

As seen below this now works.

However, there is a problem when deleting since teachers can have the same name for classes so to fix this, I added an extra condition to the SQL query where teacherID is the same as readID so that it deletes the class currently selected from the teacher, that is currently logged in, set of classes.

```
command.CommandText = "DELETE FROM Class WHERE className = '" + selected +"'AND TeacherID ='" + readID + "';";
```

Next, I added more validation to the SQL for generating a code.



When randomly generated a class code there a chance that the generated code is the same as another. When adding to the database this will give an error as there will no longer be a unique primary key.

To fix this made sure that the code is unique, if so then it will run the generate code again, hence generating a new number.

To check the generated code is unique by selecting the ClassID from class. I then read each returned field and check to see if this is equal to digits. If it is, then it will call generate again

```
bool same = false;
using (var connection = new SqliteConnection(dbName))
{
    connection.Open();
    using (var command = connection.CreateCommand())
    {
        //Inserts the new values into the database
        command.CommandText = command.CommandText = "SELECT ClassID FROM Class;";
        command.ExecuteNonQuery();
        using (IDataReader reader = command.ExecuteReader())
        {
            //Every line the SQL code outputs is read
            while (reader.Read())
            {
                //Stores password stored in the databases
                object v = reader["ClassID"];
                if (v.ToString() == digits)
                {
                    same = true;
                }
            }
            connection.Close();
        }
    }
}
if (same)
{
    Generate();
    Debug.Log("ye");
}
codeText.text = digits;
}
```

However, it is more readable and maintainable when I separated the code across two procedures each for a different purpose, one for generating the number and checking if the generated number is unique. The code is generally the same however I compare classID to code and not digits since digits is local to Generate and code is local throughout the script.

```
1 reference
public void Generate()
{
    System.Random rng = new System.Random();
    int number = rng.Next(0,10000);
    string digits = number.ToString("0000");
    code = digits;
    codeText.text = digits;
}
0 references
public void checkGen()
{
    bool same = false;
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Inserts the new values into the database
            command.CommandText = command.CommandText = "SELECT ClassID FROM Class;";
            command.ExecuteNonQuery();
            using (IDataReader reader = command.ExecuteReader())
            {
                //Every line the SQL code outputs is read
                while (reader.Read())
                {
                    //Stores password stored in the databases
                    object v = reader["ClassID"];
                    if (v.ToString() == code)
                    {
                        same = true;
                    }
                }
                connection.Close();
            }
        }
    }
    if (same)
    {
        Generate();
    }
}
```

Leaderboard

My first approach was to copy and paste the DropDownSetUp procedure, however I then realised I can reuse the procedure and make use of modular programming.

I first started off by checking the current scene and making sure the current active scene is either the remove from a Class or the leaderboard scene

```
Unity Message | 0 references
public void Start()
{
    if (SceneManager.GetActiveScene().buildIndex == 36)
    {
        DropDownSetUp();
    }
    if (SceneManager.GetActiveScene().buildIndex == 37)
    {
        DropDownSetUp();
    }
}
```

I next changed the deleteClass variable name to be selectClass so that it better suits the purpose of the variable since the purpose has now changed and, I also changed the name in the removeClass procedure.

```
3 references
public void DropDownSetUp()
{
    selectClass.ClearOptions();
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Inserts the new values into the database
            command.CommandText = command.CommandText = "SELECT className FROM Class WHERE TeacherID = '" + readID + "';";
            command.ExecuteNonQuery();
            using (IDataReader reader = command.ExecuteReader())
            {
                //Every line the SQL code outputs is read
                while (reader.Read())
                {
                    //Stores password stored in the databases
                    object v = reader["className"];
                    classOptions.Add(v.ToString());
                }
                connection.Close();
            }
        }
    }
    selectClass.AddOptions(classOptions);
}
```

I next created the leaderboardSetUp procedure

The purpose of this procedure will be to access the score and username under a chosen class and level for the currently logged in teacher.

The first line of code in the procedure that I wrote was to connect to the database that stores all my tables. I then executed an SQL query to access the username from the Student table and score from the LevelProgress table and I then join both Class and LevelProgress onto the Student table.

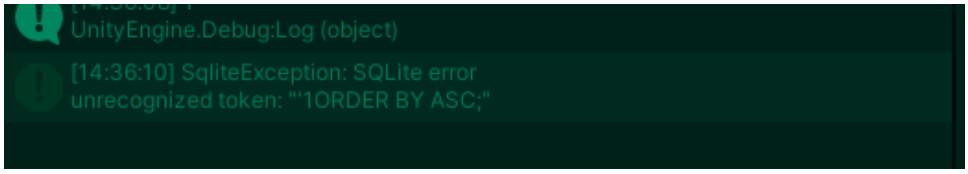
To do this I joined the Class table by stating the relationship between the two tables, that being the foreign key of ClassID within the Student table and the primary key of the Class table which is ClassID are linked. To code this in SQL, I used the JOIN ON syntax to join the Class table onto ClassID which is in both tables and since they are linked and ClassID is a foreign key the syntax requires me to make the two fields equal to each other.

I then used the same syntax to join the LevelProgress table onto the student table. As shown below

I then used a where clause to select the username and score from the selected level from the drop-down menu and the selected class from another drop-down menu and the TeacherID to be equal to readID. I then ordered the list

```
0 references
public void leaderboardSetUp()
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            //Inserts the new values into the database
            command.CommandText = @"SELECT Student.Username , LevelProgress.Score
                                    FROM Student
                                    JOIN Class
                                    ON Student.ClassID = Class.ClassID
                                    JOIN LevelProgress
                                    ON Student.StudentID = LevelProgress.StudentID
                                    WHERE
                                    LevelProgress.LevelNum = '" + selectLevel.value +
                                    "'AND Class.className = '" + selectClass.options[selectClass.value].text +
                                    "'AND  Class.TeacherID = '" + readID + "ORDER BY ASC;";
            command.ExecuteNonQuery();
            using (IDataReader reader = command.ExecuteReader())
            {
                //Every line the SQL code outputs is read
                while (reader.Read())
                {
                    //Stores password stored in the databases
                    object v = reader["Student.Username"];
                    usernames.Add(v.ToString());
                }
                connection.Close();
            }
            Debug.Log(usernames[0]);
        }
    }
}
```

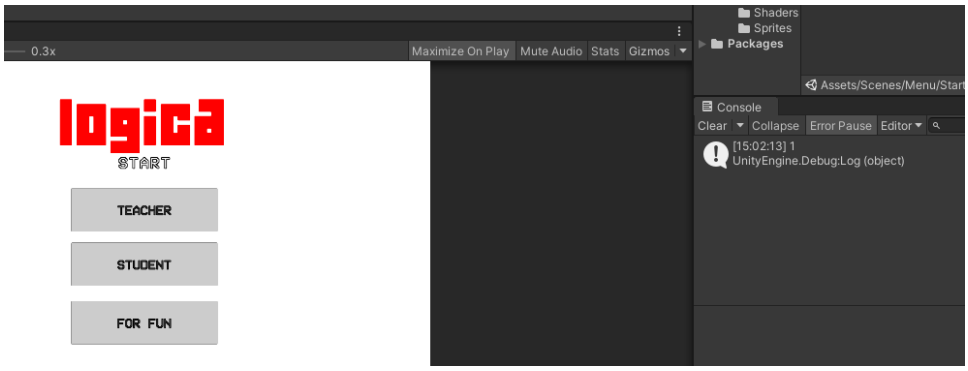
However, I got an error in my SQL code, saying order by is an unrecognised token



My first approach was to debug and see if the problem was within the reader

```
                "'AND  Class.TeacherID = '" + readID + "
command.ExecuteNonQuery();
using (IDataReader reader = command.ExecuteReader())
{
    //Every line the SQL code outputs is read
    while (reader.Read())
    {
        //Stores password stored in the databases
        object v = reader["Student.Username"];
        Debug.Log(v.ToString());
        Debug.Log("hey");
        usernames.Add(v.ToString());
    }
    connection.Close();
}
```

When running the code, the reader did not print anything, so I knew it was within the SQL code



However, during testing, I found out that the teacher does not need to type in the correct password to log in which might've given the error

```
0 references
public void backToTM()
{
    if (proceed)
    {
        SceneManager.LoadScene(34);
    }
}
```

To fix this I added the backToTM (back to teacher Menu) procedure into the teacher script so I can make use of the validation I added for the password and the proceed boolean.

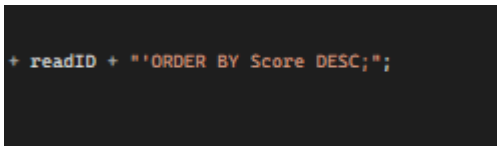
I next tested to see if any values where being added to usernames, and they were.

I soon realised that the code was not working since it did not know which variable to order by.

Below I fixed the code and added ‘ORDER BY Score ASC’

When testing and viewing the database online, I found that the scores are not being ordered in the correct order since I wanted to have the highest scores at the top

To fix this I changed ASC to DESC, which will order the score in descending order.



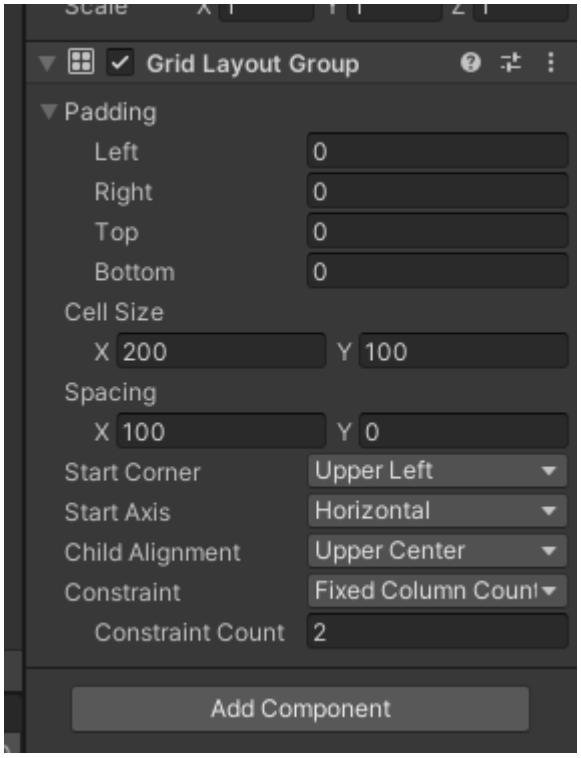
I then started to work on the set up of the leaderboard.

My idea was to instantiate an object prefab that contained two text objects that I can then change to the username and score.

The objects will be set to a child of a parent object that has the grid layout group component attached to it, this will automatically layout according to the settings any child objects to the object with that component attached.

The component below is attached to the parent object which is a game object containing the two texts which are both 200 by 100 as set by the cell size the spacing between the two objects are 100

The constraint count is set to 2 meaning that only two objects per row, hence allowing me to pair up the username and score.



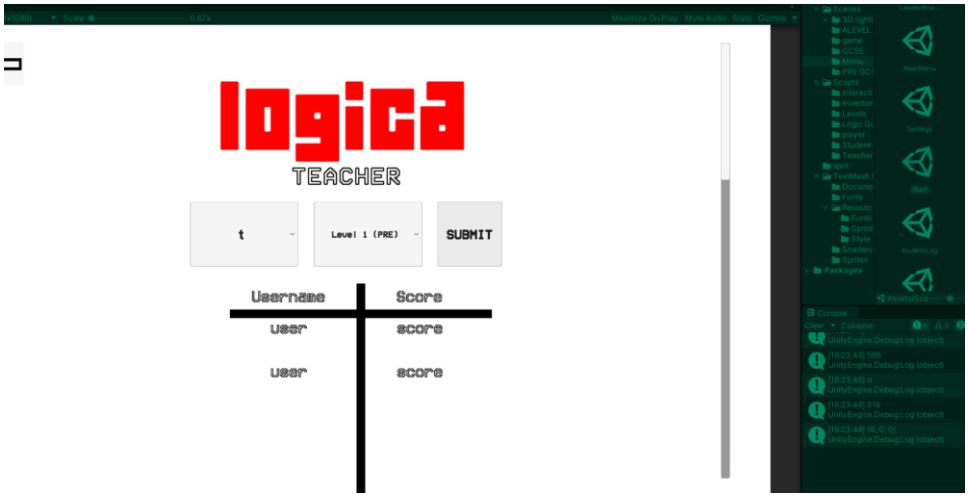
I made a list called scores and made the reader read retrieve the values from the database and add this to the list same with what I did with usernames.

I then made a foreach that iterates through each item in the list, within this loop I will instantiate the studentDetails prefab and set studentDetails' parent to be the empty object that has the grid layout group attached.

I then set the instantiated object as the last sibling so that new objects will be moved down in the grid.

```
1 reference
public void leadSetUp()
{
    Vector3Int pos = new Vector3Int(0, 0, 0);
    foreach (string score in scores)
    {
        s = Instantiate(studentDetails, pos, Quaternion.identity);
        s.transform.SetParent(parentObj);
        s.transform.SetAsLastSibling();
    }
}

1 reference
bool HasSpecialChars(string password)
```

I then tested to see if the code worked, and it did.



I then accessed the children within the prefab and got access to their text components so that I can change the text shown on screen.

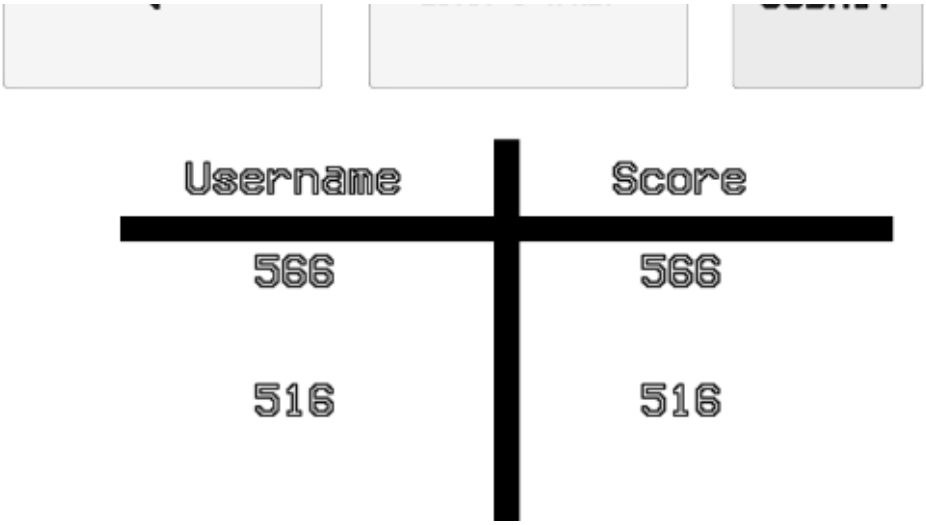
To do this I created an array of components called texts which gets all the components from the children that are TextMeshProUGUI which is a component I use for text.

I then use a foreach to access each component individually and change its text

I first change the text to equal score so that I can test to see if the code works or if I must rework my approach.

```
1 reference
public void leadSetUp()
{
    Vector3Int pos = new Vector3Int(0, 0, 0);
    Debug.Log(pos);
    foreach (string score in scores)
    {
        s = Instantiate(studentDetails, pos, Quaternion.identity);
        s.name = "studentDetails";
        s.transform.SetParent(parentObj);
        s.transform.SetAsLastSibling();
        texts = s.GetComponentsInChildren<TextMeshProUGUI>();
        foreach (TextMeshProUGUI t in texts)
        {
            if(t.name == "")
            {
                t.text = score;
            }
            else
            {
                t.text = score;
            }
        }
    }
}
```

Below is a screenshot of the code working and changing the text to show the score.



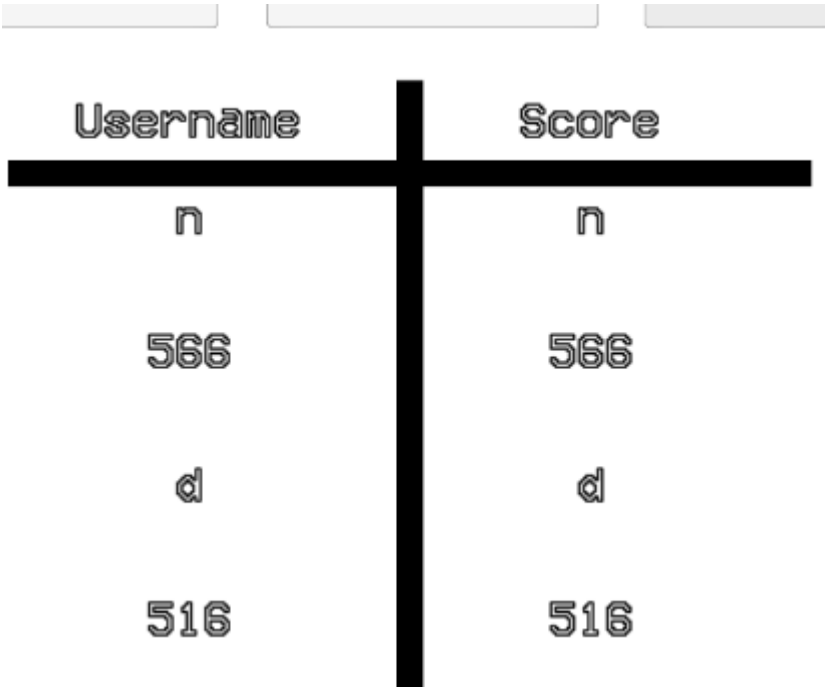
I then attempted a new approach where I add all the details into a singular list

```
    //Stores password stored in the databases
    object v = reader["Username"];
    Debug.Log(v.ToString());
    details.Add(v.ToString());
    object x = reader["Score"];
    Debug.Log(x.ToString());
    details.Add(x.ToString());
}
connection.Close();
```

I then changed the foreach to iterate through details and every time the t.name is equal to score then I set this to D however this did not work as d would be set to both texts and not just one

```
1 reference
public void leadSetUp()
{
    Vector3Int pos = new Vector3Int(0, 0, 0);
    Debug.Log(pos);
    foreach (string d in details)
    {
        s = Instantiate(studentDetails, pos, Quaternion.identity);
        s.name = "studentDetails";
        s.transform.SetParent(parentObj);
        s.transform.SetAsLastSibling();
        texts = s.GetComponentsInChildren<TextMeshProUGUI>();
        foreach (TextMeshProUGUI t in texts)
        {
            if(t.name == "Score")
            {
                t.text = d;
            }
            else
            {
                t.text = d;
            }
        }
    }
}
```

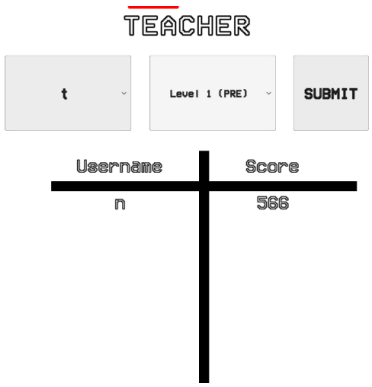
This error is shown below

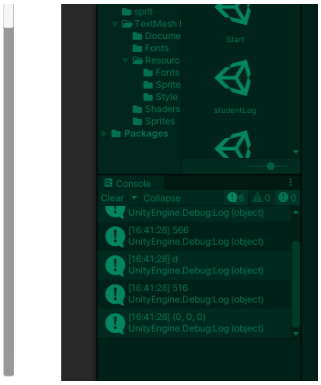


I then tried a new approach using a for loop where it would iterate through half of the list with a step of 2. This would then set the text whenever t is equal to score, it would access the value at index position of 'i' + 1 and set that to the text and if t is equal to user, then it would just access the index position of 'i'. This is because user is added first and then score.

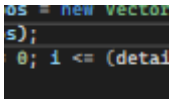
```
1 reference
public void leadSetUp()
{
    Vector3Int pos = new Vector3Int(0, 0, 0);
    Debug.Log(pos);
    for (int i = 0; i < (details.Count)/2; i = i +2)
    {
        s = Instantiate(studentDetails, pos, Quaternion.identity);
        s.name = "studentDetails";
        s.transform.SetParent(parentObj);
        s.transform.SetAsLastSibling();
        texts = s.GetComponentsInChildren<TextMeshProUGUI>();
        foreach (TextMeshProUGUI t in texts)
        {
            if(t.name == "score")
            {
                t.text = details[i+1];
            }
            if(t.name == "user")
            {
                t.text = details[i];
            }
        }
    }
}
```

However, this did give me the correct formatting but did not give me all the students in the list

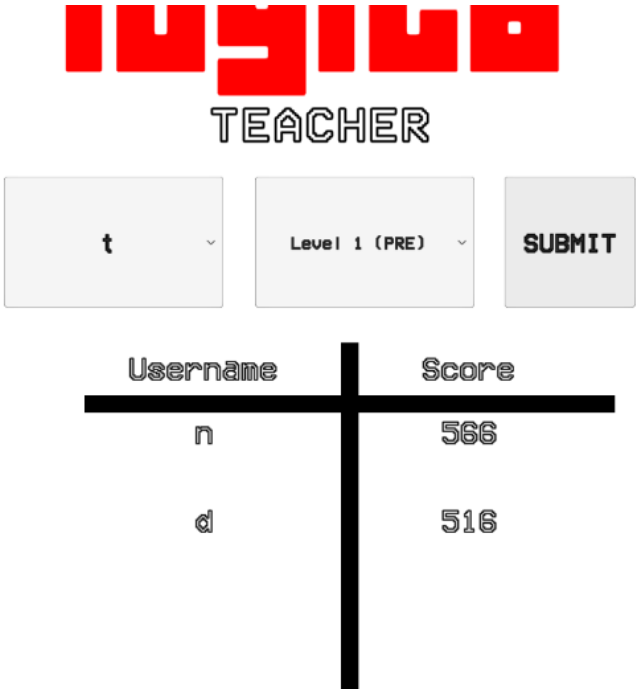




To fix this, my approach was to change the condition under which the for loop ran which is for ‘i’ to be less than or equal to half the size of the list



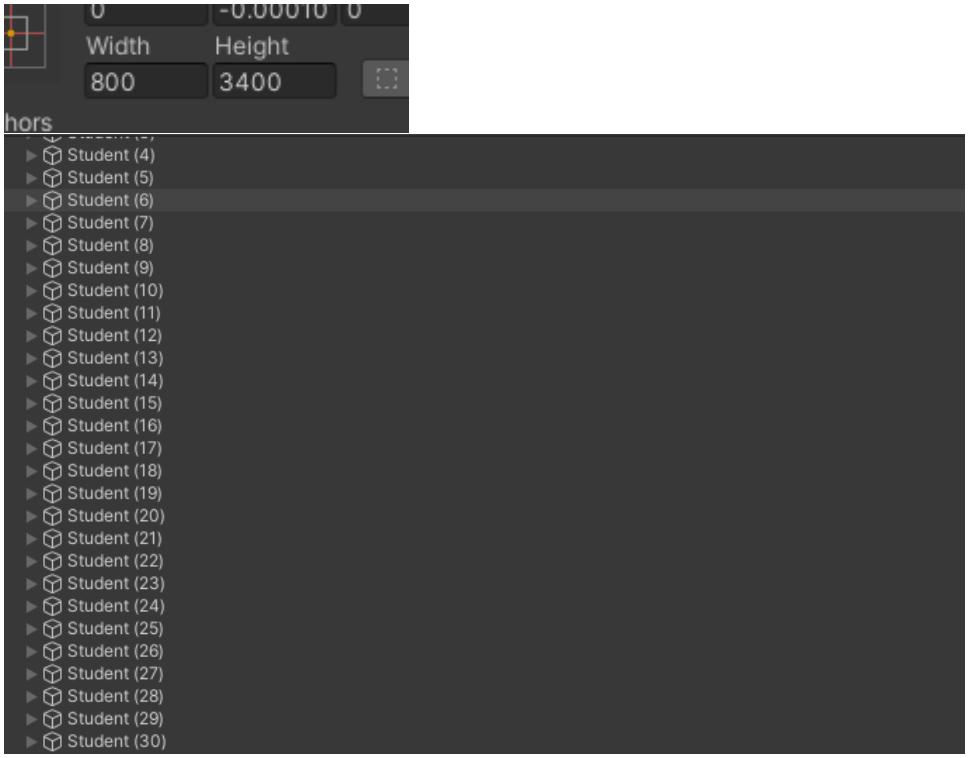
This appeared to work at first. However, with more testing it will show that this method does not work



I attempted to make the height of the scrollable area to depend on how many objects are currently instantiated

I first spawned in 32 student objects which I assume will be the most any class will have.

The perfect height for the scrollable page is 3400, this fits in all student objects.



I assumed that there will be a constant that will help determine the correct size for the number of student objects

H = KS, Height = Constant * Student Objects

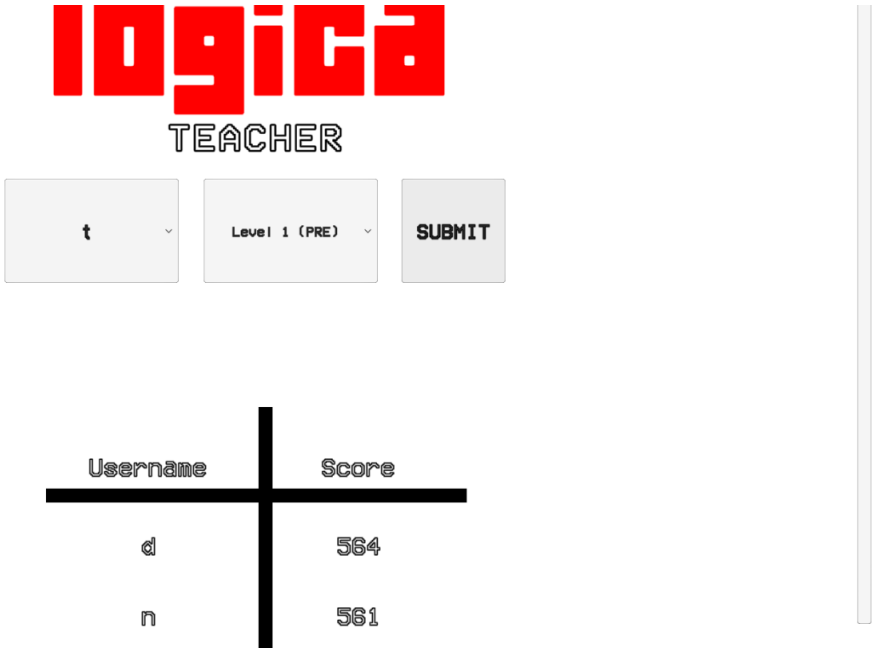
3400 = K (32)

K =106.25

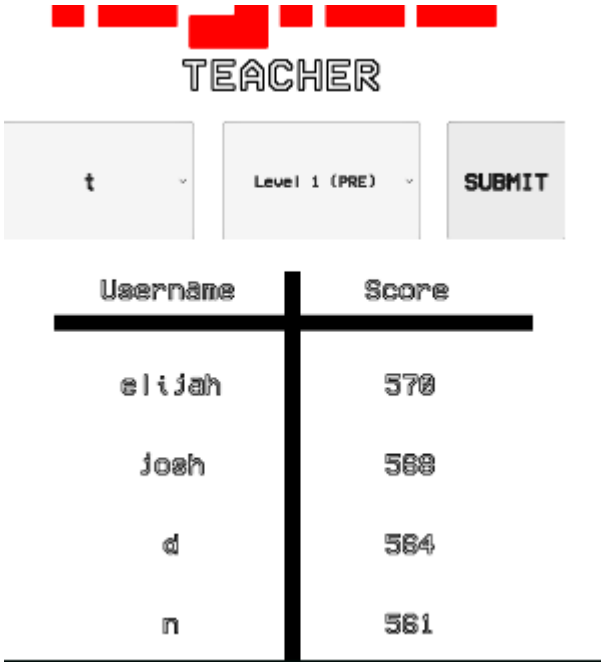
I tried to use this to change the height of the content area that can be scrolled through.

```
1 reference
public void leadSetUp()
{
    Vector3Int pos = new Vector3Int(0, 0, 0);
    float height = ((details.Count) / 2) * 106.25f;
    content.sizeDelta = new Vector2(800, height);
    scroll.value = 1;
    for (int i = 0; i <= (details.Count)/2; i = i +2)
    {
        s = Instantiate(studentDetails, pos, Quaternion.identity);
        s.name = "studentDetails";
        s.transform.SetParent(parentObj);
        s.transform.SetAsLastSibling();
        texts = s.GetComponentsInChildren<TextMeshProUGUI>();
        foreach (TextMeshProUGUI t in texts)
        {
            if(t.name == "user")
            {
                t.text = details[i];
            }
            if(t.name == "score")
            {
                t.text = details[i + 1];
            }
        }
    }
}
```

However, when testing this dramatically changed the position of the leader board, so I decided to scrap this idea and leave the height at 3400



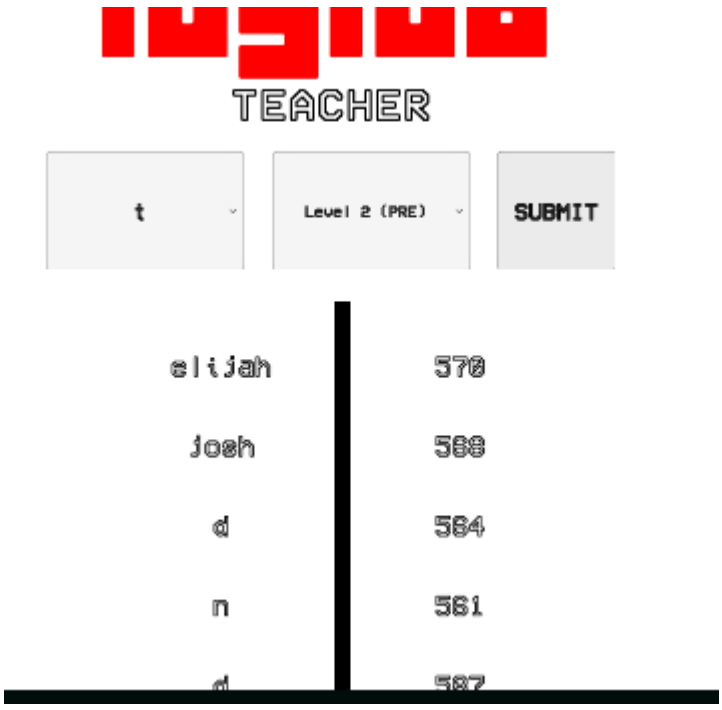
I decided to test more by adding another 2 students to see how it looks on the leaderboard



I then changed the scroll value to 1 so it always starts at the top of the page

```
1 reference
public void leadSetUp()
{
    Vector3Int pos = new Vector3Int(0, 0, 0);
    /*float height = ((details.Count) / 2) * 106.25f;
    content.sizeDelta = new Vector2(800, height);*/
    scroll.value = 1;
    for (int i = 0; i <= (details.Count)/2 + 2; i = i +2)
    {
        s = Instantiate(studentDetails, pos, Quaternion.identity);
        s.name = "studentDetails";
        s.transform.SetParent(parentObj);
        s.transform.SetAsLastSibling();
        texts = s.GetComponentsInChildren<TextMeshProUGUI>();
        foreach (TextMeshProUGUI t in texts)
        {
            if(t.name == "user")
            {
                t.text = details[i];
            }
            if(t.name == "score")
            {
                t.text = details[i + 1];
            }
        }
        Debug.Log((details.Count) / 2);
        Debug.Log(i);
    }
}
```

However, when trying to change to view a different level or class the details on the leaderboard are never cleared.



To fix this I created a procedure that clears the details list and changes first to true meaning that the first leaderboard request has happened

```
}
0 references
public void clicked()
{
    first = true; //to make sure its the first time it is being clicked
    details.Clear(); // clear details list
}
```

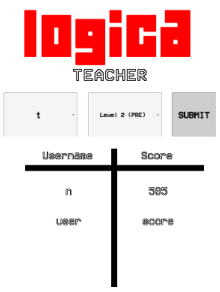
Then my first approach was to add an if statement check for first and if it's true then I remove and destroy the objects in the studentObjs list. I then added a line of code to add the instantiated object into the studentObjs list. I also renamed the variables to be more readable and better suited for the code

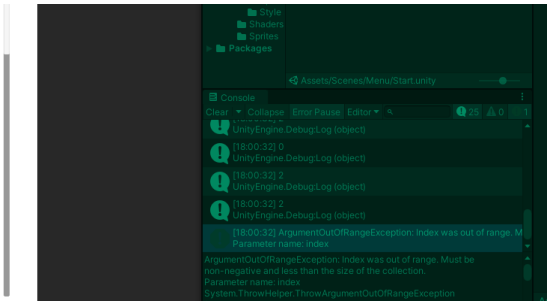
```
content.sizeDelta = new Vector2(800, height);*/
scroll.value = 1;
for (int i = 0; i <= (details.Count)/2 + 2; i = i +2)
{
    if (first)
    {
        foreach (GameObject s in StudentObjs)
        {
            StudentObjs.Remove(s);
            Destroy(s);
        }
    }
    StudentObj = Instantiate(studentDetails, pos, Quaternion.identity);
    StudentObjs.Add(StudentObj); // add to list so you can clear the leaderb
    StudentObj.name = "studentDetails";
    StudentObj.transform.SetParent(parentObj);
```

I then tried a different approach as the one above did not work. My new approach was to destroy the object then remove it from the list. I put it inside a for loop where J never increased, and this only stopped when the length of studentObjs was 1 or less. The code destroys the object at the first index position (0) and then the remaining items in the list are renumbered to replace the removed item.

```
scroll.value = 1;
for (int i = 0; i <= (details.Count)/2 + 2; i = i +2)
{
    if (first)
    {
        for (int j = 0; StudentObjs.Count > 1;)
        {
            Destroy(StudentObjs[j]);
            StudentObjs.RemoveAt(j);

        }
    }
    StudentObj = Instantiate(studentDetails, pos, Quaternion.i
```

However, this gave me an error saying the index was out of range as shown below. I originally thought that it was due to the for loop I just made however this was not the case as the problem lied in the for loop iterating through details





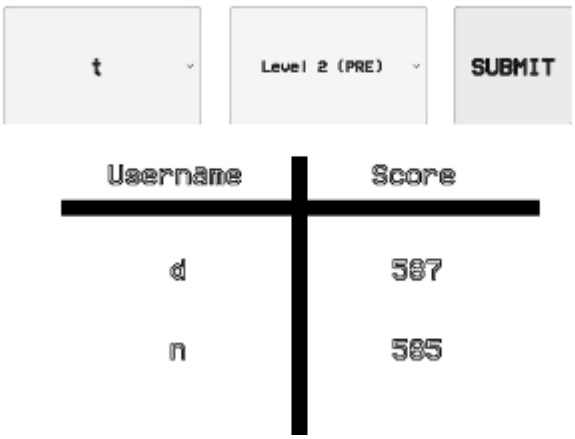
I moved the for loop outside the details loop and changed the <= (less than or equal too) condition to only be < (less than). I then removed the halving of the list.

```
}*/
for (int i = 0; i < (details.Count) ; i = i +2)
{
    StudentObj = Instantiate(studentDetails, pos, Quaternion.identity);
    StudentObjs.Add(StudentObj); // add to list so you can clear the leaderboard when changing classes or levels
    StudentObj.name = "studentDetails";
    StudentObj.transform.SetParent(parentObj);
    StudentObj.transform.SetAsLastSibling();
    texts = StudentObj.GetComponentsInChildren<TextMeshProUGUI>();
    Debug.Log(i);
    foreach (TextMeshProUGUI t in texts)
    {
        if(t.name == "user")
        {
            t.text = details[i];
        }
        if(t.name == "score")
        {
            t.text = details[i + 1];
        }
    }
}
details.Clear(); // clear details list
```

I then changed the for loop to be more efficient by removing the j variable altogether since I can just use 0 as shown below

```
Vector3 pos = new Vector3(0, 0, 0);
/*float height = ((details.Count) / 2) * 106.2
content.sizeDelta = new Vector2(800, height);*
scroll.value = 1;
if (first)
{
    for (; StudentObjs.Count > 0;)
    {
        Destroy(StudentObjs[0]);
        StudentObjs.RemoveAt(0);
    }
}
for (int i = 0; i < (details.Count) ; i = i +2
```

Below is a screenshot of my code working. It also clears itself and refills when changing the class or level.

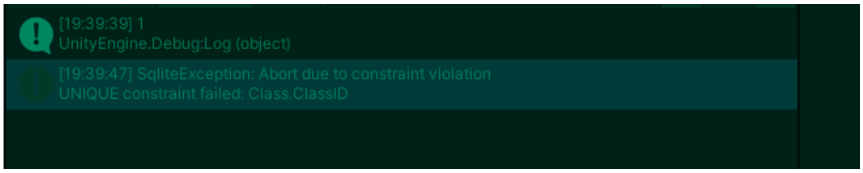


After further consultation, my client is content with the implementation and making use of databases within the program for the teacher data.

Post Development Testing

This section will be bug fixes that happened during testing:

When I was testing to see what happened if you did not enter a code, I got a SqliteExcepetion Error where the unique constraint on the classID primary key failed.



To fix this I added an if statement to check if the class name input is empty or if the code is null which means nothing has been generated. I then changed the submit button to tell the user to fill in all the inputs and change proceed to false.

Else it inserts the values as expected

```
0 references
public void AddClass()
{
    Debug.Log(code);
    if (classNameINP.text == "" || code == null)
    {
        submitStext.text = "Fill in all inputs";
        proceed = false;
    }
    else
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Class(ClassID,TeacherID, className) " +
                    "VALUES ('" + code + "','" + readID + "','" + classNameINP.text + "');";
                command.ExecuteNonQuery();
            }
            connection.Close();
        }
    }
```

Next when testing I noticed the studentID was 0

StudentID	LevelNum	Score
0	0	555
0	1	583

This was due to the code below executing the inserting into the database but did not execute the reader bit in the database since there was no Select statement.

```
//Adds a new player into the Student table
0 references
public void AddStudent()
{
    if (proceed)
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Student(ClassID,Username, Password, Sensitivity, Graphics) " +
                    "VALUES ('" + code + "','" + username + "','" + password + "','" + 2.5 + "','" + 0 + "');";
                command.ExecuteNonQuery();
                using (IDataReader reader = command.ExecuteReader())
                {
                    //Every line the SQL code outputs is read
                    while (reader.Read())
                    {
                        object x = reader["StudentID"];
                        readID = Convert.ToInt32(x);
                        Debug.Log(readID);
                    }
                    connection.Close();
                }
            }
            connection.Close();
        }
    }
}
```

To fix this I then added a new SQL query that selects the data from the student table where the username and password match the inputted values.

```
//Adds a new player into the Student table
0 references
public void AddStudent()
{
    Debug.Log(proceed);
    if (proceed)
    {
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Student(ClassID,Username, Password, Sensitivity, Graphics) " +
                    "VALUES ('" + code + "','" + username + "','" + password + "','" + 2.5 + "','" + 0 + "');";
                command.ExecuteNonQuery();
                command.CommandText = command.CommandText = "SELECT * FROM Student WHERE Username = '" + username + "'AND Password ='" + password + "';";
                command.ExecuteNonQuery();
                using (IDataReader reader = command.ExecuteReader())
                {
                    //Every line the SQL code outputs is read
                    while (reader.Read())
                    {
                        object x = reader["StudentID"];
                        readID = Convert.ToInt32(x);
                        Debug.Log(readID);
                    }
                    connection.Close();

                }
            }
            connection.Close();
        }
    }
}
```

Further testing led to me finding another problem. Whenever I logged in since I changed the procedure to be moved within the student table studentIn never became true, this variable allows me to access the pause menu only when I have logged in and I am not in a log in or sign up or teacher screen.

To fix this problem I created setter method for studentIn which is a static variable. This would change studentIn to true

```
0 references
public void SetIn()
{
    studentIn = true;
}
```

I then accessed this procedure and called within the Main Menu procedure within the student script

```
0 references
public void MainMenu()
{
    if (proceed)
    {
        SceneManager.LoadScene(22);
        gameObject.GetComponent<ScreenDirecter>().SetIn();
    }
}

0 references
```

Next, during more testing I found out that the graphics quality when initialised do not affect the actual game quality only the drop-down menu.

To fix this I set the quality level in start.

```
Unity Message | 0 references
private void Start()
{
    QualitySettings.SetQualityLevel(playerGraphics);
    if (gameObject.name == "Manager")
    {
        if (playerSens != 0)
        {
            GameObject[] gameObjects;
            gameObjects = GameObject.FindGameObjectsWithTag("settings");
            if (gameObjects.Length != 0)
            {
                gameObjects[0].GetComponent<Slider>().value = playerSens;
                gameObjects[1].GetComponent<TMP_Dropdown>().value = playerGraphics;
            }
        }
    }

}
```

Then I changed setGraphics to only set the playerGraphics variable to the passed in parameter

```
1 reference
public void setGraphics(int i)
{
    playerGraphics = i;
}
```

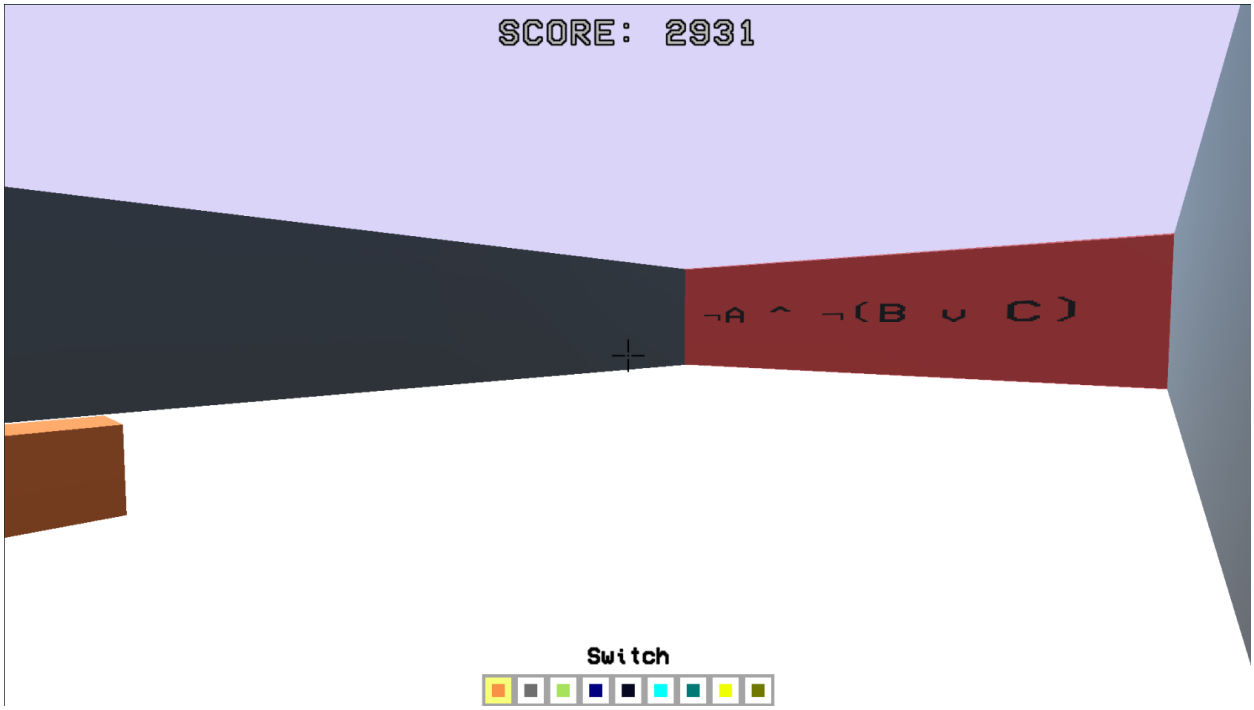
Since I forgot to set and update the graphics in student and playerInfo directly from the procedure within screen director I also added this. Which accesses both these scripts to update and set the graphics.

```
0 references
public void setQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
    gameObject.GetComponent<student>().UpdateGraphics(qualityIndex);
    gameObject.GetComponent<PlayerInfo>().setGraphics(qualityIndex);
}
```

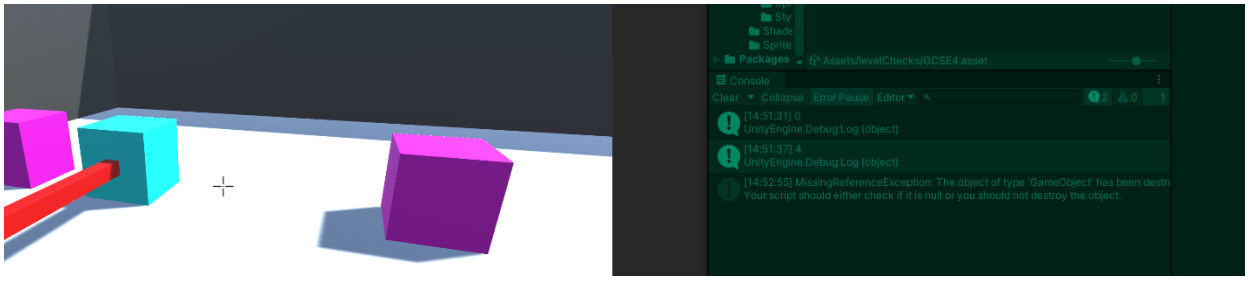
I then cleaned up some of my code with issues such as this from copy and pasting. This will take up less storage and is extra useless lines.

```
{
    //Inserts the new values into the database
    command.CommandText = command.CommandText = "SELECT ClassID FROM Class;";
    command.ExecuteNonQuery();
```

Next, I added the level design question into the 3DLevels. Below is an example of how it looks in game for PRE-GCSE question 6



When testing, I removed a not gate and it gave me an error that I am trying to access a destroyed object.

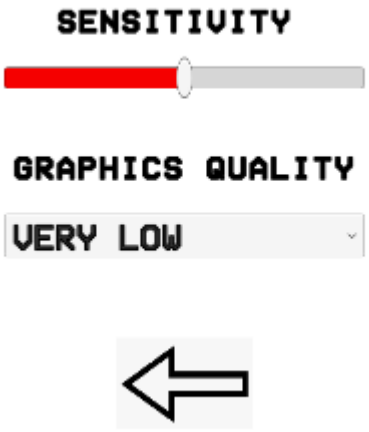


To fix this I added a check to see if the object in the current index position of 'i' within the circuit is different to null to then carry on executing the rest of the code that was within that if statement

```
if (GetComponent<power>() != null)
{
    for (int i = 0; i < GetComponent<power>().circuit.Count;)
    {
        if (GetComponent<power>().circuit[i] != null)
        {
            if (GetComponent<power>().circuit[i].name != "NOT")
            {
                GetComponent<power>().circuit.Remove(GetComponent<power>().circuit[i]);

            }
            else
            {
                i++;
            }
        }
    }
}
for (int i = 0; segs.Count>0; i++) {
```

This fixed the issue.

I also realised that the student would have no idea has to what the settings do despite them being able to infer. To fix this issue I added text to explain the purpose of the UI shown on screen.



Next, I had to do extra database validation to make the program more robust. Currently the program doesn't check if a table exists before the student tries to log in or a teacher.

To fix this I created a new procedure for the button to execute before it does anything else.

This connects to the database and tries to select the student table from a group of tables. It will then return true if it is found and false if not.

I then used a reader to retrieve the true or false value and then check it to see if the table exits. If not, then proceed is set to false so none of the other actions will happen and submit button text is set to sign up first.

```
0 references
public void checkDbStudent()
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            command.CommandText = "SELECT name FROM sqlite_master WHERE type='table' AND name= 'Student'";
            using (IDataReader reader = command.ExecuteReader())
            {
                tableExists = reader.Read();
            }
        }
    }
    if (!tableExists)
    {
        submitStext.text = "Sign up first";
        proceed = false;
    }
}
```

Below is a screenshot of me testing my code and it is working as expected.

I next did the same thing for the teacher login and sign up. I copied the code and then changed it so that it checks if a teacher table exists instead of a student table

```
0 references
public void checkDbTeacher()
{
    tableExists = false;
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            command.CommandText = "SELECT name FROM sqlite_master WHERE type='table' AND name= 'Teacher'";
            using (IDataReader reader = command.ExecuteReader())
            {
                tableExists = reader.Read();
            }
        }
    }
    if (!tableExists)
    {
        submitStext.text = "Sign up first";
        proceed = false;
    }
}
```

Below is a screenshot of my code working for the teacher log in screen.

I next started to work on checking if the student and level progress table have been made when trying to access the student's username and score for the leaderboard in the teacher script.

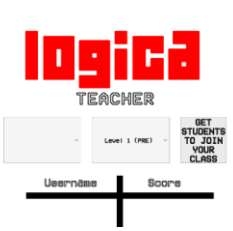
I first copied the same code again but changed he code to check for the LevelProgress table and Student table. I then added a an if condition to check if proceed is true so that the code will only run if there are tables.

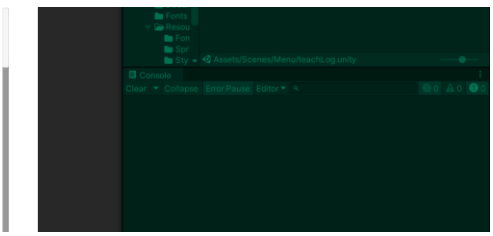
```
0 references
public void checkDbStudent()
{
    tableExists = false;
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            command.CommandText = "SELECT name FROM sqlite_master WHERE type='table' AND name= 'Student'";
            using (IDataReader reader = command.ExecuteReader())
            {
                tableExists = reader.Read();
            }
        }
    }
    if (!tableExists)
    {
        submitStext.text = "Get students to join your class";
        proceed = false;
    }
    else
    {
        checkDbLP();
    }
}

1 reference
public void checkDbLP()
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            command.CommandText = "SELECT name FROM sqlite_master WHERE type='table' AND name= 'LevelProgress'";
            using (IDataReader reader = command.ExecuteReader())
            {
                tableExists = reader.Read();
            }
        }
    }
    if (!tableExists)
    {
        submitStext.text = "None of your students have started to complete levels";
        proceed = false;
    }
}
```

Below is a screenshot of me adding the proceed condition

```
0 references
public void DatabaseLeaderBoard()
{
    if (proceed)
    {
        using (var connection = new SqliteConnection(
        {
            connection.Open();
```

In the screenshot below whenever I submitted, the code will change the submit message to output that text instead of ‘submit’. Hence my code works.





I then had to check if a class exist when a teacher tried to remove an empty class.

To do this, I first used the same code to check if a class exists.

```
0 references
public void checkDbClass()
{
    tableExists = false;
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            command.CommandText = "SELECT name FROM sqlite_master WHERE type='table' AND name= 'Class'";
            using (IDataReader reader = command.ExecuteReader())
            {
                tableExists = reader.Read();
            }
        }
    }
    if (!tableExists)
    {
        submitStext.text = "Add a class first";
        proceed = false;
    }
}
```

I then added a proceed function to check if the code says to proceed however this does not work.

```
0 references
public void RemoveClass()
{
    if (proceed)
    {
        string selected = selectClass.options[sel
        using (var connection = new SqliteConnect
```

Below is a screenshot showing that the code is working however, the logic behind it is wrong because I check if a class table exists however, I don't check if there is a class within the options. Also, the class table is created when a teacher logs in/ signs up technically there should be no reason to check for the existence of a class table.

To fix this I adjusted the code to try and find a record within the table. If the number of records, it reads is more than 0 then tableExists is true.

```
1 reference
public void checkDbClass()
{
    using (var connection = new SqliteConnection(dbName))
    {
        connection.Open();
        using (var command = connection.CreateCommand())
        {
            command.CommandText = "SELECT COUNT(*) FROM Class WHERE TeacherID = @TeacherID;";
            command.Parameters.AddWithValue("@TeacherID", readID);
            using (var reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    int count = reader.GetInt32(0);
                    if (count != 0)
                    {
                        tableExists = true;
                    }
                }
            }
        }
    }
    if (!tableExists)
    {
        submitStext.text = "Add a class first";
    }
}
```

I then added a submit text to check for the existence of the record and if doesn't exist then change the text of the submit button to tell the user to add a class first.

```
0 references
public void RemoveClass()
{
    if (classOptions.Count != 0)
    {
        string selected = selectClass.options[selectClass.value].text;
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                if (tableExists)
                {
                    //Inserts the new values into the database
                    command.CommandText = "DELETE FROM Class WHERE className = '" + selected + "'AND TeacherID ='" + readID + "';";
                    command.ExecuteNonQuery();
                }
            }
            connection.Close();
        }
        classOptions.Clear();
        DropDownSetUp();
        checkDbClass();
    }
}
```

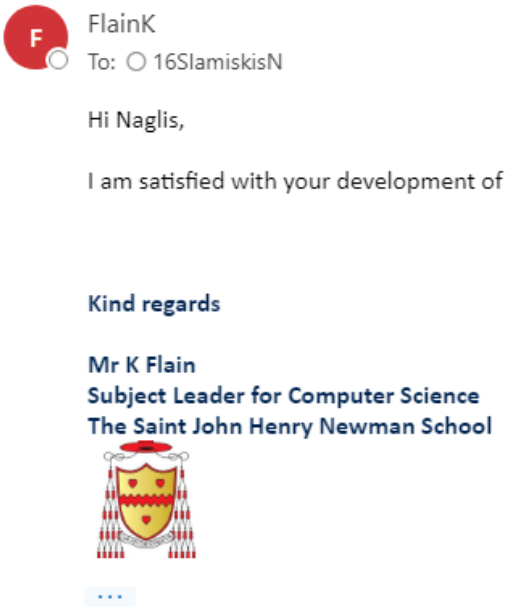
I then added extra code so that it doesn't run the removeClass procedure if the length of the options list is equal to 0. I also add an extra layer of condition to only remove the class of recordExists is true.

```
0 references
public void AddClass()
{
    if (classNameINP.text == "" || code == null)
    {
        submitStext.text = "Fill in all inputs";
        proceed = false;
    }
    else
    {
        submitStext.text = "Submitted";
        proceed = true;
        using (var connection = new SqliteConnection(dbName))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                //Inserts the new values into the database
                command.CommandText = "INSERT INTO Class(ClassID,TeacherID, className) " +
                    "VALUES ('" + code + "','" + readID + "','" + classNameINP.text + "');";
                command.ExecuteNonQuery();
            }
            connection.Close();
        }
    }
}
3 references
```

I then changed the addClass function to change the submit button text to submitted so the user knows when their new class is added.

After further consultation, my client is content with the process of post development testing that occurred

Client Sign Off



FlainK

To: 16SlamiskisN

Hi Naglis,

I am satisfied with your development of your program and the outcome.

Kind regards

Mr K Flain
Subject Leader for Computer Science
The Saint John Henry Newman School

Section 4: Testing

Videos

I made 3 videos:

I first video called LogicaTeacher1 is showcasing the sign-up process for a teacher, creating a class and removing a class. This also showcases the robustness of the code to not cause any errors within the database and not make the program crash.

The second video I made called LogicaStudent showcases the log in and sign-up process of a student. It then showcases the gameplay aspects and all the screens that can be accessed by the student and being able to select Levels. It also shows that each level can be completed and will show a game complete screen when finished. It also shows being able to go to the next level on this screen. The video then showcases the ability to change settings in game.

The third video that I made to showcase my program that is called LogicaTeacher2 showcases the log in process for the teacher and needing the correct username and password to log in. It then also shows the functionality of the leaderboard.

The fourth video I made called LogicaStudent2 is to showcase the gameplay aspects of my game. This video showcases features such as movement, placing and break blocks. It also showcases inventory management, furthermore, also shows all the inputs and outputs for each of the gates.

Test tables

Gameplay						
Function / method being tested	**Scenario**	**Input**	**Testing Type**	**Expected results**	**Actual Outcome**	**Improvements required**
Movement	In game	W (default)	Valid	Player will walk forwards, towards where the player camera is looking	Player will walk forwards, towards where the player camera is looking. Shown in LogicaStudent Video and LogicaStudent2 Video	n/a
		Other (e.g., H)	Invalid	The player will not move forward	The player will not move forward. User only moves when WASD is pressed	n/a

		A (default)	Valid	Player walks to the left, camera moves along with the player	Player walks to the left, camera moves along with the player. Shown in LogicaStudent Video and LogicaStudent2 Video	n/a
		Other (e.g., Y)	Invalid	The player will not move to the left	The player will not move to the left User only moves when WASD is pressed	n/a
		S (default)	Valid	Player will walk backwards, 180° where the player camera is looking	Player will walk backwards, 180° where the player camera is looking. Shown in LogicaStudent Video and LogicaStudent2 Video	n/a
		Other (e.g., G)	Invalid	The player will not move backwards	The player will not move backwards. User only moves when WASD is pressed	n/a
		D (default)	Valid	Player walks to the right, camera moves along with the player	Player walks to the right, camera moves along with the player. Shown in LogicaStudent Video and LogicaStudent2 Video	n/a
		Other (e.g., T)	Invalid	The player will not move to the right	The player will not move to the right. User only moves when WASD is pressed	n/a
Player Look	In game	Move the mouse	Valid	Camera rotation correlates to mouse movement	Camera rotation correlates to mouse movement. Shown in LogicaStudent Video and LogicaStudent2 Video	n/a

		Other (e.g., Mouse button)	Invalid	Camera will not rotate	Camera will not rotate	n/a
Player Jump	In game	Space (default)	Valid	Player jumps along the y axis with the camera following the player	Player jumps along the y axis with the camera following the player. Shown in LogicaStudent2 Video. (1:42)	n/a
		Other (e.g., J)	Invalid	Player does not jump	Player does not jump	n/a
Player Interacts with Switch	In game	E (default)	Valid	Laser Power turns on or off depending on the state of the Laser Power	Laser Power turns on or off depending on the state of the Laser Power. Shown in LogicaStudent Video (1:00-1:02) and LogicaStudent2 Video (1:40-1:41)	n/a
		Other (e.g., R)	Invalid	The laser power's state will stay the same and so will everything connected to it	The laser power's state will stay the same and so will everything connected to it. Shown in LogicaStudent and LogicaStudent2	n/a
Player access Inventory	In game and out of game	Q (default)	Valid	Inventory screen is brought up, black opacified canvas is also brought to cover the players camera view. Movement and freeze camera rotation. Gameplay continues as usual after Q is pressed again	Inventory screen is brought up, black opacified canvas is also brought to cover the players camera view. Movement and freeze camera rotation. Gameplay continues as usual after Q is pressed again. Shown in LogicaStudent2 (3:05-3:15)	n/a
		Other (e.g., F)	Invalid	Player's inventory	Player's inventory does not pop up.	n/a

				does not pop up. Player can move and look around as usual.	Player can move and look around as usual.	
Access inventory slots	In game	Scroll wheel	Valid	Access slots with their correspondin g numbers	Access slots with their corresponding numbers. Shown in logicaStudent and logicaStudent2 videos	n/a
		Other (e.g., F)	Invalid	Player's inventory does not pop up. Player can move and look around as usual.	Player's inventory does not pop up. Player can move and look around as usual.	n/a
Player Place block	In game	Right click	Valid	The selected block in the inventory array is placed from a prefab of that block	The selected block in the inventory array is placed from a prefab of that block	n/a
Player break block	In game	Left click	Valid	If the block being looked at is able to be broken (checked via tags or mask) Then block is destroyed and also destroys turns off the power	If the block being looked at is able to be broken (checked via tags or mask) Then block is destroyed and also destroys turns off the power. Shown in LogicaStudent2	n/a
		Other	Invalid	If block is highlighted and breakable inputs other than left click will not do anything, power stays on	If block is highlighted and breakable inputs other than left click will not do anything, power stays on	n/a

Pause menu	In game	ESC	Valid	Open Pause Screen, movement and being able to look around is disabled	Open Pause Screen, movement and being able to look around is disabled. Shown in LogicaStudent2(1:20)	n/a
Laser Power/ switch	In game	-	Valid	Can be placed (right click), broken (left click) and interacted with by pressing E, when allowed. this then turns the current on or off (thus influencing other blocks). Generates a laser in the z direction	Can be placed (right click), broken (left click) and interacted with by pressing E, when allowed. this then turns the current on or off (thus influencing other blocks). Generates a laser in the z direction. Shown in LogicaStudent2 (1:40-1:41)	n/a
Laser	In game	-	Valid	Is generated based on the rotation of the Laser Power + Carries the correct signal output between logic gates. laser power changes the laser output	Is generated based on the rotation of the Laser Power + Carries the correct signal output between logic gates. laser power changes the laser output. Shown in LogicaStudent2	n/a
AND GATE	In game	-	Valid	IF both inputs are true this gate will output positive and both inputs are from the x axis of the block	IF both inputs are true this gate will output positive and both inputs are from the x axis of the block. Shown in LogicaStudent2	n/a
OR GATE	In game	-	Valid	IF one or both inputs are true this	IF one or both inputs are true this gate will output	n/a

				gate will output positive and both inputs are from the x axis of the block	positiveand both inputs are from the x axis of the block. Shown in LogicaStudent2	
NOT GATE	In game	-	Valid	INVERSES the output (input = positive output = false, input false output positive) And the only input is from the opposite side of the output	INVERSES the output (input = positive output = false, input false output positive) And the only input is from the opposite side of the output. Shown in LogicaStudent2	n/a
XOR GATE	In game	-	Valid	IF only one input is true this gate will output true and both inputs are from the x axis of the block	IF only one input is true this gate will output true and both inputs are from the x axis of the block. Shown in LogicaStudent2	n/a
NAND GATE	In game	-	Valid	SAME as AND GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block	SAME as AND GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block. Shown in LogicaStudent2	
NOR GATE	In game	-	Valid	SAME as OR GATE but outputs the opposite output due to the not function and both inputs are from the x	SAME as OR GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block. Shown in LogicaStudent2	n/a

				axis of the block		
XNOR GATE	In game	-	Valid	SAME as XOR GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block	SAME as XOR GATE but outputs the opposite output due to the not function and both inputs are from the x axis of the block. Shown in LogicaStudent2	
Mirror	In game	-	Valid	When a laser hits the mirror depending on its rotation it will reflect the mirror 90°	When a laser hits the mirror depending on its rotation it will reflect the mirror 90°. Shown in LogicaStudent2	n/a
PlayerScore	In game	-	Valid	Player score is updated in top right of gameplay in the overlay. Player score is measured by the time taken to complete the level	Player score is updated in top right of gameplay in the overlay. Player score is measured by the time taken to complete the level. Shown in LogicaStudent2 and LogicaStudent	n/a

Other in-game features				
UI screens	**Action**	**Expected result**	**Actual Outcome**	**Improvements Required**
Start menu	Teacher button	Redirects to the Teacher login screen	Redirects to the Teacher login screen. Shown in LogicaTeacher1 (0:02)	n/a
	Student button	Redirects to the Student login screen	Redirects to the Student login screen. Shown in LogicaStudent(0:05)	n/a
	X Button	Exists to desktop	There is no X button	Add an X button

Teacher Sign Up and Log in	Username Input	Allows the user to enter an email	Allows the user to enter an email. Shown in LogicaTeacher1 and LogicaTeacher2	n/a
	Password Input	Allows the user to enter a password	Allows the user to enter a password. Shown in LogicaTeacher1 and LogicaTeacher2	n/a
	Confirm Password Input	Allows the user to enter a password and checks if it is the same as the original one.	Allows the user to enter a password and checks if it is the same as the original one. Shown in LogicaTeacher1	n/a
	Enter button	These inputs are submitted and checked in the database if successful redirects to Teacher Menu Screen, if not error is displayed	These inputs are submitted and checked in the database if successful redirects to Teacher Menu Screen, if not error is displayed. Shown in LogicaTeacher1 and LogicaTeacher2	n/a
	Back button	Returns to previous screen	Returns to previous screen. Shown in LogicaTeacher1 and 2	n/a
	X button	Exists to desktop	There is no X button	Add an X button
Student Sign Up and Log in	Username Input	Allows the user to enter an email	Allows the user to enter a username. Shown in LogicaStudent and LogicaStudent2	n/a
	Password Input	Allows the user to enter a password	Allows the user to enter a password. Shown in LogicaStudent and LogicaStudent2	n/a
	Confirm Password Input	Allows the user to enter a password and checks if it is the same as the original one.	Allows the user to enter a password and checks if it is the same as the original one. Shown in LogicaStudent and LogicaStudent2	n/a
	Code Input	User can enter a code that will	User can enter a code that will join	n/a

		join the student to the class screen	the student to the class screen. Shown in LogicaStudent and LogicaStudent2	
	Enter button	These inputs are submitted and checked in the database if successful redirects to Teacher Menu Screen, if not error is displayed	These inputs are submitted and checked in the database if successful redirects to Teacher Menu Screen, if not error is displayed. Shown in LogicaStudent and LogicaStudent2	n/a
	Back button	Returns to previous screen	Returns to previous screen	n/a n/a
	X button	Exists to desktop	There is no X button	Add an X button
Teacher Menu	Add Button	Redirects to Add class screen and creates a new class database	Redirects to Add class screen and creates a new class database. Shown in LogicaTeacher1 (0:23)	n/a
	Remove Button	Redirects to Remove class screen	Redirects to Remove class screen. Shown in LogicaTeacher1 (0:20)	n/a
	Leaderboard Button	Redirects to Leaderboard class screen	Redirects to Leaderboard class screen. Shown in LogicaTeacher1 (0:46)	n/a
	X button	Exit to desktop	There is no X button	Add an X button
Teacher Add	Generate Button	Generates a random number in a 4-digit format that leads with 0s if its less than 1000.Also checks to see if its unique and regenerates	Generates a random number in a 4-digit format that leads with 0s if its less than 1000.Also checks to see if its unique and regenerates. Shown in LogicaTeacher1 (0:26)	n/a

	Code text	Generated number shows up here	Generated number shows up here. Shown in LogicaTeacher1 (0:26)	n/a
	Class name input	Teacher can enter the class name	Teacher can enter the class name. Shown in LogicaTeacher1 (0:27)	n/a
	Submit button	Submits all data to the database and inserts the class into the database	Submits all data to the database and inserts the class into the database. Shown in LogicaTeacher1 (0:28)	n/a
	X button	Exit to desktop	There is no X button	Add an X button
Teacher Remove	Class Drop Down Menu pt1	Menu drops down with correct classes groups	Menu drops down with correct classes groups	n/a
	Class Drop Down Menu pt2	Select the class and it saves correctly this is then submitted and connects to the database	Select the class and it saves correctly this is then submitted and connects to the database	n/a
	Submit button	Submits all data to the database and deletes the selected class from the database	Submits all data to the database and deletes the selected class from the database	
	X button	Exit to desktop	There is no X button	Add an X button
Leaderboard	Class Drop Down Menu pt1	Menu drops down with correct classes groups	Menu drops down with correct classes groups. Shown in LogicaTeacher2	n/a
	Class Drop Down Menu pt2	Select the class and it saves correctly this is then submitted and connects to the database	Select the class and it saves correctly this is then submitted and connects to the database. Shown in LogicaTeacher2	
	Level Drop Down Menu pt1	Menu drops down with	Menu drops down with correct Levels	n/a

		correct Levels from pre gcse – gcse and finally alevel	from pre gcse – gcse and finally alevel. Shown in LogicaTeacher2	
	Level Drop Down Menu pt2	Select the level and it saves correctly this is then submitted and connects to the database	Select the level and it saves correctly this is then submitted and connects to the database. Shown in LogicaTeacher2	n/a
	-	Shows the correct class, names and scores in the correct order and places	Shows the correct class, names and scores in the correct order and places. Shown in LogicaTeacher2	n/a
	Scroll Wheel	Moves the page down	Moves the page down.	n/a
	Scroll Slider	Updates depending on how far down the page you are and also moves the page down when held and dragged down	Updates depending on how far down the page you are and also moves the page down when held and dragged down	n/a
	Back button	Returns to previous screen	Returns to previous screen	n/a
	X button	Exit to desktop	There is no X button	Add an X button
Main Menu	Level Selection Button	Redirects you to Level Selection Screen	Redirects you to Level Selection Screen. Shown in the student videos	n/a
	Settings Button	Redirects you to Settings Screen	Redirects you to Settings Screen. Shown in the student videos	n/a
	Help Button	Redirects you to Help screen	Redirects you to Help screen.	n/a
	X button	Exit to desktop	Exit to desktop	n/a
Settings	Sensitivity slider	Student can move the slider around and this will change the sensitivity in the	Student can move the slider around and this will change the sensitivity in the 3D levels in	n/a n/a

		3D levels in game	game. Shown in the student videos	
	Graphics Drop Down Menu pt1	Menu drops down with correct classes groups	Menu drops down with correct classes groups. Shown in the student videos	
	Graphics Drop Down Menu pt2	Select the class and it saves correctly this is then submitted and connects to the database	Select the class and it saves correctly this is then submitted and connects to the database. Shown in the student videos	
	Back button	Returns to either Pause Menu or Main Menu depending on the previous scene	Returns to either Pause Menu or Main Menu depending on the previous scene. Shown in the student videos	
	X button	Exit to desktop	There is no X button	Add an X button
Pause Menu	Main Menu button	Redirects you to Main Menu Screen	Redirects you to Main Menu Screen	n/a
	Settings button	Redirects you to Settings Screen	Redirects you to Settings Screen	n/a
	Resume button	Returns to previous screen	Returns to previous screen	n/a
	X button	Exit to desktop	There is no X button	Add an X button
Help	Content Image	Contains an image of how to play and the keybinds	Contains an image of how to play and the keybinds. Shown in the image below	n/a
	Back button	Returns to previous screen	Returns to previous screen	n/a
	Scroll Wheel	Moves the page down	Moves the page down	n/a
	Scroll Slider	Updates depending on how far down the page you are and also moves the page down when held and dragged down	Updates depending on how far down the page you are and also moves the page down when held and dragged down	n/a

	X button	Exit to desktop	There is no X button	Add an X button

Database Inputs and Outputs

UI screens	Action	Input	Input Data	Testing Type	Expected result	Actual Outcome	Improvements Required
Creating Tables	Pressing Student button in the Start Menu	User clicks the student button	n/a	Valid	Redirects to Student sign up or login in screen and creates the student table if it does not exist	Redirects to Student sign up or login in screen and creates the student table if it does not exist. LogicaStudent (0:05)	n/a
	Pressing Teacher button in the Start Menu	User clicks the Teacher button	n/a	Valid	Redirects to Teacher sign up or login in screen and creates the student table if it does not exist	Redirects to Teacher sign up or login in screen and creates the student table if it does not exist. LogicaTeacher1 (0:02)	n/a

	Pressing Level Selection button in the Main Menu	User clicks the Level Selection button	n/a	Valid	Redirects to PRE-GCSE Level selection and creates the LevelProgress table if it does not exist	Redirects to PRE-GCSE Level selection and creates the LevelProgress table if it does not exist. LogicaStudent2 (0:18)	n/a
	Pressing Add Class button in the Teacher Menu	User clicks the Add Class button	n/a	Valid	Redirects to Teacher sign up or login in screen and creates the Class table if it does not exist	Redirects to Teacher sign up or login in screen and creates the Class table if it does not exist. Shown in LogicaTeacher1 (0:23)	n/a
	A student trying to log in without signing up beforehand / Student table does not exist yet	User clicks the submit button on the log in screen	n/a	Erroneous	No actions should be performed to access the database since it does not exist, and text should display telling the user to sign up first	No actions should be performed to access the database since it does not exist, and text should display telling the user to sign up first. Shown in the student videos (LogicaStudent and LogicaStudent2)	n/a
	A teacher trying to log in without signing up beforehand	User clicks the submit button on the log in screen	n/a	Erroneous	No actions should be performed to access the database	No actions should be performed to access the database	n/a

	/ teacher table does not exist yet				since it does not exist, and text should display telling the user to sign up first	since it does not exist, and text should display telling the user to sign up first	
	Pressing submit with something not filled in	User clicks the submit button on the sign-up screen with nothing inputted in any of the fields	Username = “” Password = “” Confirm Password = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.	Text is shown to the user to enter something, the code stops since the validation has failed.	n/a
	Password is 8 characters long or less	User clicks the submit button on the sign-up screen with the password being less than 8 characters	Password = “passwor”	Erroneous	Text is shown to the user to enter a longer password, the code stops since the validation has failed.	Text is shown to the user to enter a longer password, the code stops since the validation has failed.	n/a
	Password does not contain a special character	User clicks the submit button on the sign-up screen with the password not containing a special character	Password= “password”	Erroneous	Text is shown to the user to enter a longer password, the code stops since the validation has failed.	Text is shown to the user to enter a longer password, the code stops since the validation has failed.	n/a
	Pressing submit with password and confirm password not matching	User clicks the submit button on the sign-up screen with the password and confirm	Password = “password” Confirm Password = “possword”	Erroneous	Text is shown to the user to say that the passwords do not match, the code stops	Text is shown to the user to say that the passwords do not match, the code stops	n/a

		password inputs not matching			since the validation has failed.	since the validation has failed.	
	Pressing submit without typing in the class code (student sign up only)	User clicks the submit button on the sign-up screen with nothing inputted in any of the fields	Code = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.	Text is shown to the user to enter something, the code stops since the validation has failed. Shown in LogicaStudent video (1:30)	n/a
	Pressing submit when entering non numbers into the class code (student sign up only)	User clicks the submit button on the sign-up screen with letters inputted in the class code	Code = “091H”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.	Text is shown to the user to enter something, the code stops since the validation has failed.	n/a
	Pressing submit when entering code that is not 4 digits in length (student sign up only)	User clicks the submit button on the sign-up screen with the length of the inputted class code is not equal to 4 digits	Code = “01234”	Erroneous	Text is shown to the user to enter a longer code or the correct code, the code stops since the validation has failed.	Text is shown to the user to enter a longer code or the correct code, the code stops since the validation has failed.	n/a
	Entering every field, reaching every validation requiremer t(student) and	User clicks submit when all validation is reached	Username = “student” Password = “student1 @” Confirm Password = “student1 @”	Valid	Redirects to main menu while also inserting the data as a new entry into	Redirects to main menu while also inserting the data as a new entry into the student	n/a

	pressing submit		Code = “0123”		the student table	table. Shown in the student videos (LogicaStudent and LogicaStudent2)	
	Entering every field, reaching every validation requirement (teacher) and pressing submit	User clicks submit when all validation is reached	Username = “teacher” Password = “teacher@” Confirm Password = “teacher @”	Valid	Redirects to teacher menu while also inserting the data as a new entry into the teacher table	Redirects to teacher menu while also inserting the data as a new entry into the teacher table. Shown in the teacher videos	n/a
	Pressing submit with something not filled in	User clicks the submit button on the Log in screen with nothing inputted in any of the fields	Username = “” Password = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.	Text is shown to the user to enter something, the code stops since the validation has failed.	n/a
	Pressing submit and the Password and Username match not found within the database	User clicks the submit button on the Log in screen however their entry does not exist in the database since they have not signed up or their details are incorrect.	Username = “Josh” Password = “wrongpassword”	Erroneous	Text is shown to the user to enter the correct details, the code stops since the validation has failed.	Text is shown to the user to enter the correct details, the code stops since the validation has failed. Shown in LogicaStudent video	n/a
	Entering every field, reaching	User clicks submit in the log in	Username = “naglis”	Valid	Redirected to Main Menu	Redirected to Main Menu.	n/a

	every validation requirement (Student) and pressing submit	screen and passes all validation	Password = “password @”			Shown in student videos	
	Entering every field, reaching every validation requirement (teacher) and pressing submit	User clicks submit in the log in screen and passes all validation	Username = “flain” Password = “teachers @”	Valid	Redirected to Teacher Menu	Redirected to Teacher Menu. Shown in teacher videos	n/a
	Graphics drop down new option selected	User selects new option from the drop-down section	Graphics Drop Down Option = “Low”	Valid	Edits the graphics field data for the student that is changing the graphics so that it can be retrieved from the database when loading a 3d level to initialise the graphics settings	Edits the graphics field data for the student that is changing the graphics so that it can be retrieved from the database when loading a 3d level to initialise the graphics settings. Shown in the LogicaStudent video	n/a
	Change in sensitivity slider	User slides the slider to pick a new sensitivity	Sensitivity slider value = 3.4	Valid	Edits the sensitivity field data for the student	Edits the sensitivity field data for the student	n/a

					that is changing the sensitivity so that it can be retrieved from the database when loading a 3d level to initialise the player sensitivity	that is changing the sensitivity so that it can be retrieved from the database when loading a 3d level to initialise the player sensitivity. Shown in the LogicaStudent video	
Student Score	Student completes a level for the first time	When win for that level is true.	n/a	Valid	It will check if there is already an entry since there is not, it will enter a new entry into the LevelProgress database table putting in the correct data.	It checks if there is already an entry since there is not, it will enter a new entry into the LevelProgress database table putting in the correct data. Shown in the LogicaStudent video	n/a
	Student completes a level again but with a better score	When win for that level is true.	n/a	Valid	It will check if there is already an entry since there is, it will edit the entry from the LevelProgress	It checks if there is already an entry since there is, it will edit the entry from the LevelProgress	n/a

					database table changing the data to its new value only if the new value is larger than the old value in the database	database table changing the data to its new value only if the new value is larger than the old value in the database. Shown in the LogicaStudent video	
	Student completes a level again but with a worse score	When win for that level is true.	n/a	Valid	It will check if there is already an entry since there is, however since the new value is lower than the value in the database nothing updates or changes.	It checks if there is already an entry since there is, however since the new value is lower than the value in the database nothing updates or changes. Shown in the LogicaStudent video	n/a
	Pressed submit button with no inputs filled	Click submit on the add screen with no code generated or class name inputted	Code = “” Class Name = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.	Text is shown to the user to enter something, the code stops since the validation has failed.	n/a
	Pressed submit with only	Click submit on the add	Code = “” Class Name = “Year 13”	Erroneous	Text is shown to the user to	Text is shown to the user to	n/a

	the class name filled	screen with no code generated			enter something, the code stops since the validation has failed.	enter something, the code stops since the validation has failed.	
	Pressed submit with no class name and only a code	Click submit on the add screen with no class name inputted	Code = “5126” Class Name = “”	Erroneous	Text is shown to the user to enter something, the code stops since the validation has failed.	Text is shown to the user to enter something, the code stops since the validation has failed.	n/a
	Reaching every validation requiremer t and pressing submit	Click submit with all validation reached	Code = “5126” Class Name = “Year 13”	Valid	It will add the class into the database. It will also tel the user that their entry has been submitted	It adds the class into the database. It will also tel the user that their entry has been submitted/ . Shown in the LogicaTeac her1 video	n/a
Teacher Remove Class	Initiating the drop-down Menu	When first loading in the screen, the drop-down menu must be initiated with the correct values /classes	n/a	Valid	Will retrieve all classes from the database for that teacher and to initialise the drop-down options	Will retrieve all classes from the database for that teacher and to initialise the drop-down options. Shown in the LogicaTeac her1 video	n/a
	Pressing Submit	Click submit on	Class Drop down	Valid	Will delete the class	Will delete the class	Cascade the

		the teacher remove screen	option = “Year 13”		entry from the database and will also update the drop-down menu	entry from the database and will also update the drop-down menu however it should also cascade. Shown in the LogicaTeacher1 video	deletion to remove unnecessary data
	Initiating the drop-down Class Menu	When first loading in the screen, the drop-down menu must be initiated with the correct values/classes	n/a	Valid	Will retrieve all classes from the database for that teacher and to initialise the drop-down options	Will retrieve all classes from the database for that teacher and to initialise the drop-down options. Shown in the LogicaTeacher1 video	n/a

Client testing

To further test my game post development to gain feedback as mentioned in my design stage I brought in my game into a year 9 class for them to test and give feedback on the functionality of the game. Here is the feedback I received from the students:

Positives:

- The students enjoyed the simplistic look of the game
- They also enjoyed how bug-free the program was due to my thorough testing before giving it to the class
- They enjoyed the blocky sandbox aspect since it reminded them of Minecraft which made them enjoy the game a lot.

Negatives:

- When changing the sensitivity of their mouse the slider felt extremely laggy
- Add models to make the gates clearer to the students and will depend on them recognising the gates by their symbols instead of colours.

I fixed one of the negative issues quite quickly by adjusting the code that is being accessed by the slider

```
0 references
public void sensitvity(Slider sensSlide)
{
    playerSens = Mathf.Round(sensSlide.value * 100) / 100;
    gameObject.GetComponent<student>().UpdateSens(playerSens);
}
```

Before the code updated every time the slider changed the sensitivity as seen above, this causes lag as it keeps updating the database multiple times which results in more FDE cycles than needed

To fix this I will need to check for when the user stops changing the slider value and then update the sensitivity and set playerSens to the slider value

I first created a new script to attach onto the slider called slider.

This script contains a Boolean that tests for a release of a mouse click from the slider via the OnPointerUp method.

Next the code changes the endDrag variable to the opposite of its current value.

```
Unity Script (1 asset reference) | 3 references
public class slider : MonoBehaviour, IDeselectHandler
{
    public bool drag;

    0 references
    public void OnDeselect(BaseEventData data)
    {
        drag = true;
        Debug.Log("dragging");
    }

}
```

I then adjusted the sensitivity void code to always set the drag value to false and then check if it has been changed to true. If so, the slider has been declared (the sensitivity procedure has been called via the slide) it will then check to see if drag is true. If it is true, it then assigns the new value of the slider to be playerSens while also updating the database and setting drag back to false

```
}

Unity Message | 0 references
public void Update()
{
    if (slid != null)
    {
        sensitvity(slid);
    }
}
1 reference
public void sensitvity(Slider sensSlide)
{
    slid = sensSlide;
    Debug.Log(sensSlide.GetComponent<slider>().drag);
    if (sensSlide.GetComponent<slider>().drag)
    {
        float temp = Mathf.Round(sensSlide.value * 100) / 100;
        setSens(temp);
        gameObject.GetComponent<student>().UpdateSens(playerSens);
        sensSlide.GetComponent<slider>().drag = false;
    }
}
```

This successfully fixes the problem and shows that I cater towards client feedback.

Overall, most negatives that the students found in their playtest such as the models will be added into future updates. They also enjoyed the experience very much since they do not get to play games that are so resemblant of other popular games in class since most other games are not educational however, my game being educational allowed my program to be a suitable solution to enable the students to think computationally and advance their problem-solving skills.

I next tested my game even further with my client, Mr Flain, to see what possible feedback on the student and teacher side that he could give me.

Positives:

- The colours used within the game were vibrant and eye catching
- The game looked played well as the game was simple to understand and it did not need instructions
- UI navigation is extremely clear for both students and teachers
- Most requirements have been met

Negatives:

- Add a for fun section, to enable more replay ability and more customisation for the teacher and students
- Also add a way for multiple lasers to come out of a source.

In conclusion, after testing, Mr Flain thought that the game is good but needed a few extra additions rather than changes to the game. This can be implemented in future updates that I will release for the game so that it can successfully reduce its limitations and solve the problem to its best ability.

Client Sign Off

FlainK
To: 16SlamiskisN

Hi Naglis,

I am delighted with the testing process of your program and the robustness

Kind regards

Mr K Flain
Subject Leader for Computer Science
The Saint John Henry Newman School

...

Section 5: Evaluation

Requirements

Menu General Criteria		
Testing Criteria	**Requirements Met**	**Evidence/ Improvements**
Logica title must be in 04b_19 font, #FF0000 no outline, lowercase	This requirement has been fully met.	This has been met since every scene that includes the logic title is in the 04b_19 font, the colour of the title is #FF0000. It has no outline and is lowercase. Shown in all videos
Other text must be VCR OSD Mono font (same for every menu or text that is on screen). Colour will usually be #FFFFFF. Usually, will be uppercase and have an outline	This requirement has been partially met.	This is because half of the text follow these criteria, but some text is #000000 instead of plain white, for this reason this criteria has been partially met. Shown in all videos
White background for all menus apart from inventory	This requirement has been fully met.	This has been met since every menu apart from the Inventory screen has a white background. Shown in all videos
All buttons can be pressed and execute the correct action	This requirement has been fully met.	This is because every button can be pressed and when pressed it will do the correct action. Shown in all videos

All input boxes can be written in	This requirement has been fully met.	This has been met due to every input box can be written in, and the data inputted can be retrieved. Shown in all videos
All scrollable screens or sections must have a scrollbar and be able to scroll via moving the page with your mouse or via the scroll wheel or via moving the scrollbar	This requirement has been fully met.	This requirement has been achieved since every scrollable page contains a scrollbar. These scrollable pages can be scrolled via the scroll wheel on the mouse, by dragging the page up and down or dragging the scroll bar up and down. Shown in all videos
All scrollbars must be dependant to where the user is in the page and update the scrollbar	This requirement has been fully met.	This is because each scrollbar shows the position of the user on the current page. Shown in all videos
All drop down menus must show the correct options assigned to them and must also have a smaller menu that will drop down	This requirement has been fully met.	This has been met since all drop down Menus output the correct options needed for the player to interact. All drop down Menus also show the options when the user clicks on it. Shown in all videos
All drop down menus should be able to select each option	This requirement has been fully met.	This is because all drop down menus, you can select all the options. Shown in all both teacher videos and the LogicaStudent video
Cursor in menus	This requirement has been fully met.	This requirement has been achieved since each menu contains the mouse cursor. Shown in all videos

Game General Criteria		
Testing Criteria	**Requirements Met**	**Evidence/ Improvement**
Each Level must be able to be completed	This requirement has been fully met.	The requirement has been fully met since each level has a win condition that can be met hence causing the game to understand that the player has won. Shown in the student videos
Each Level must have a score that is stopped when the level is completed	This requirement has been fully met.	This requirement has been achieved because when a player has won the score will stop. During the game the

		score decreases so that the player knows what their score is. Shown in the student videos
Each level must have a game over screen if the student does not complete it	This requirement has been fully met.	This has been met since when the score reaches 0 the game will load the game over scene. Shown in LogicaStudent2
Each Item in the inventory must be able to be placed and broken via left click and right click respectively	This requirement has been fully met.	This has been met due to the code instantiating the prefab object of the current selected item. Shown in the student videos
When a block is broken and is part of a circuit it will update the circuit	This requirement has been fully met.	This requirement has been achieved since when breaking a block, it will remove itself from every list to not cause any errors. Shown in LogicaStudent2
When a block is placed in the middle of a circuit it will update the circuit	This requirement has been fully met.	This is because the code checks for if the user is pressing either mouse button attempting to place or break an object in return. It calls the genUpdate function to update the circuits and lists necessary. Shown in LogicaStudent2
Each item in the inventory should be accessible via the scroll wheel	This requirement has been fully met.	The requirement has been fully met since, the code checks for an input from the scroll wheel and when it does it will change the index position. Shown in the student videos
The selected item must be visually represented	This requirement has been fully met.	The requirement has been fully met since the current selected item has a selected item image overlay to symbolise that it is selected, hence meeting the requirement. Shown in the student videos
Pressing WASD allows you to move forward left backward and right respectively	This requirement has been fully met.	This requirement has been achieved due to whenever the player inputs the WASD keys it will do the corresponding action to enable movement for the player within the game. Shown in the student videos

Moving your mouse around will be affected by the user's current sensitivity	This requirement has been fully met.	This has been met since moving your mouse around it will take in the mouse X and Y input and translate this to the players camera rotation while also being influenced by the sensitivity. Shown in the student videos
3d levels will be affected by the users currently selected graphics option	This requirement has been fully met.	This is because I have created a settings menu where the user can select a graphics option to determine the graphics for the 3D levels. Shown in the LogicaStudent video
User can press escape to bring up the pause Menu	This requirement has been partially met.	The requirement has been partially met since the user can still access via the escape key, but it only happens when the student is logged in. Shown in the student videos
Whenever walking next to a switch, a pop up will be shown that says how to interact with the switch	This requirement has been fully met.	This requirement has been achieved since being next to/ close enough to a Switch object will enable the pop up to be visible to the player. Shown in the student videos
User can press E when the pop up is active to interact with the switch and it will produce a laser	This requirement has been fully met.	This is because the Switch will generate a laser if the onOff variable is true and if there are lasers and E is pressed again then it will degenerate. Shown in the student videos
User can press Q in the 3D level to bring up their inventory and close it	This requirement has been fully met.	This requirement has been achieved since pressing Q will bring up an overlay of your inventory and stopping all actions within the game when it is brought up. When Q is pressed again it will close the. Shown in the LogicaStudent2 video
User can rearrange their inventory when it is open	This requirement has been fully met.	This has been met since the student is able to drag items and drop click on other items to swap their positions. Shown in the LogicaStudent2 video
All gates must do their corresponding action with the inputs that are received	This requirement has been fully met.	This is because I have a gates script checking the inputs that

		are going into the gate. Shown in the LogicaStudent2 video
Mirrors will bounce the laser in a 90-degree rotation based on its own rotation	This requirement has been fully met.	The requirement has been fully met since when lasers hit a mirror it bounces in the 90 degrees in the direction that it is meant to which is based on the rotation of the mirror. Shown in the LogicaStudent2 video
Lasers will stop when it hits an object unless it is a player	This requirement has been fully met.	This is because lasers will stop generating in the same direction when hitting an object. Shown in the LogicaStudent2 video
All levels must have the correct answers to win	This requirement has been fully met.	This requirement has been achieved from the win condition that each level contains. Shown in the LogicaStudent video
Wrong answers will either be told to the user that it is wrong, or nothing will happen	This requirement has been fully met.	This has been met since all levels whenever there is a wrong answer there are not punishments and nothing happens. Shown in the LogicaStudent2 video
Each 3D level must have a laboratory aesthetic	This requirement has been partially met.	This is because the aesthetic is not exactly what I wanted however, I still maintained the white and grey simplistic look which fits the aesthetic that I envisioned the game to look like. To improve this, I can create more models and add more details in the level to fit the aesthetic. Shown in the student videos
Crosshair in each 3D level	This requirement has been fully met.	This requirement has been achieved since I have an image that is at the centre of the screen. Shown in the student videos
Cursor in 2D levels	This requirement has been fully met.	This has been met since cursors are enabled in all 2d levels. Shown in the student videos

Database Criteria		
Testing Criteria	**Requirements Met**	**Evidence / Improvement**

Each table must be created if it does not exist	This requirement has been fully met.	The requirement has been fully met since when creating a table in SQL I use ‘CREATE IF NOT EXIST', this creates the table only if it does not exist already. This has been met as shown in the development stage for databases
Each primary key must be unique	This requirement has been fully met.	This is because I use autoincrement for Teacher and Student IDs but for ClassIDs I make sure it is unique when generating the code. The composite key for LevelProgress is always unique since if the player plays the same level twice it will update the score and not create a new entry into the database. This has been met as shown in the development stage for databases
As little redundant or useless data as possible	This requirement has not been fully met.	This has not been fully met since when deleting classes I do not cascade and delete the students and other data too. Once a student leaves school or a teacher stops using the program, their data is never deleted and hence stores unnecessary data. However, the program does try to keep redundant data to a minimum. To improve this, I will add options to delete your account and cascade the deletion of entries from the database
Each field is atomic, it cannot be broken down further	This requirement has been fully met.	The requirement has been fully met since each field in the database cannot be broken down. This has been met as shown in the design stage for databases
The teacher table must be able to store the teacher's data	This requirement has been fully met.	This requirement has been achieved since the teacher's data is inserted into the teacher table. This has been met as shown in the development stage for databases

Teachers must be linked/ associated with a class	This requirement has been fully met.	The requirement has been fully met since the class table has a foreign key called TeacherID which is the primary key in the teacher table. This has been met as shown in the development stage for databases
Student table must be able to store the student's data and settings	This requirement has been fully met.	This is because the student's data is inserted into the student table. This has been met as shown in the development stage for databases
Students must be linked/ associated with a class	This requirement has been fully met.	This has been met since the student table consists of a ClassID foreign key which is the primary key in the class table. This has been met as shown in the development stage for databases
Students must be linked/ associated with many levels	This requirement has been fully met.	This requirement has been achieved since the LevelProgress table has a foreign key that is part of a composite key called StudentID which is the primary key in the student table. This has been met as shown in the development stage for databases
Passwords must be hashed	This requirement has not been met.	To improve this, I will add hashed passwords in future updates
Passwords must be more than 8 characters long	This requirement has been fully met.	The requirement has been fully met since the code validates if the password is more than 8 characters long if not then it tells the user to make it longer. This has been met as shown in the development stage for databases
Passwords must contain a special character	This requirement has been fully met.	This has been met since the code validates if the password entered contains a special character. This has been met

		as shown in the development stage for databases

Changes made during development

The changes I made during development were changing the models since I did not have time to create the models.

I also changed some text to be all black / be #000000. I chose to do this since it made some text contrast to the background more making it more accessible.

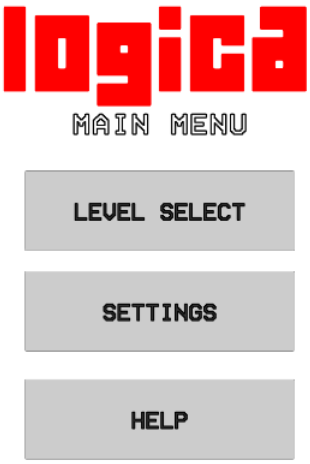
I also did not add the functionality of hashing passwords before adding them to the database. This is because I did not enough time to develop this and because I felt like it was not currently needed, and I may add it in future updates.

I removed the ability to access the pause menu from any screen as it does not make logical sense for the teacher to access the pause menu which contains levels. This also causes a bunch of errors within the code such as a teacher being able to access and complete a level will result in an error as the readID is not declared in the student script. Hence, I made the pause menu only accessible by the student when they are logged in.

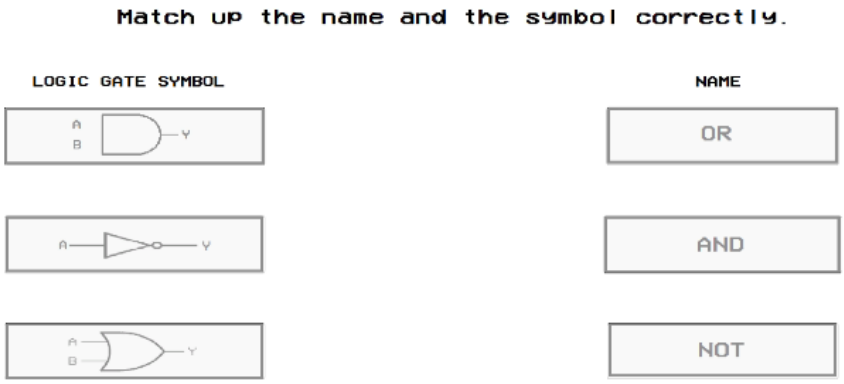
Usability

Every screen I made is easy to use and understand and is easy to navigate. As seen in the images of my screens shown in development.

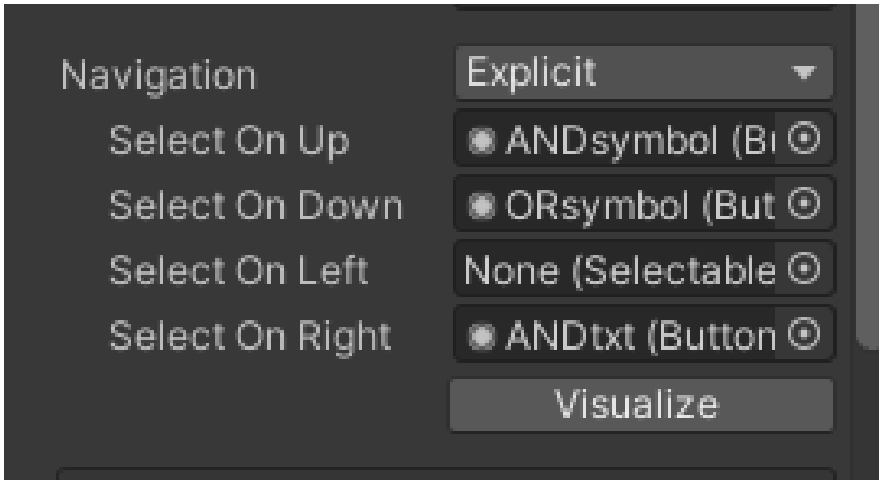
My main aim within screen designs and the criteria was to make sure all the text was visible and contrasts the white background while also making it simple and easy to understand.



I made sure that all the text on buttons clearly explained what their purpose was so that the user understands what each button does. I also added text when validation is required for example passwords to show an error message and allow the user to enter their password again. I also made sure that the title and text clearly contrasted the background letting the user to be able to see the text even with limited vision. I also made sure that most of the text used in screens are in uppercase and bold to make sure that the text pops out more and contrasts the button, text box or drop-down menu letting the user know the purpose of it.



Note: Navigation shown below is for the Not symbol button.



I also included navigation using the Tab button however I only included this for the match up level. An improvement I wish to make is to allow the user to access each button and interactable element using the Tab key to navigate. This would be a good quality of life change to make for the program.

Other improvements I can make is adding settings such as colourblind settings to make certain colours be visible for students with a colour blind disability since when placing down the logic gates and the fact that they currently do not have models and are only distinguishable from the colours makes it way harder for colour blind people to engage in this game especially at its current state. Some people with vision disabilities may need a yellow background instead of a white background to be able to see properly. I can add settings to adapt the game to suit this need and make the program more accessible.

Other students with physical disabilities in their hands may need controller support or the ability to change key binds to make the game more accessible since a keyboard might feel uncomfortable despite me grouping the current key binds all together making it accessible for most people. Key binds also allow customisation for users to make the interact and inventory button to whatever they want. I can add extra settings to change key binds however, I will also have to change the help menu to mention that the key binds mentioned are the default ones.

I would like to test this game with a group if students with a range of disabilities and discuss how to make the game more accessible for them.

Code

During the development stage, I made sure that all variable names were suitable hence making my code easier to read and more manageable. I also used indentation despite C# not needing indentation to keep my code more readable and manageable. I also used modular programming to divide the problem into subroutines that I can call multiple times due to needing to reuse the same lines of code multiple times. Using modular programming also made maintaining the code much easier as there is less code and another benefit to modular programming is that it is also quicker to write. Furthermore, I separated different code into different scripts depending on their purpose and which part of the program the code came under for example for the leader board code I put that in the teacher script, so it is easier to access teacher details for SQL statements and the leader board is for teachers to view their classes progress when they log in which makes logical sense to add it under this script. Additionally, I used Object Oriented Programming to create my game as it is easier to create my solution. This type of programming paradigm allows for better reusability and enhances the ability to upgrade my program and code in the future. Using C# made it easier to implement OOP into my solution due to the ability of accessing the methods and attributes of each logic gate and player object, furthermore the use of an OOP approach also allowed me to visually represent logic gates using the 3D environment provided by Unity to allow for an immersive game to be.

However, I would like to improve my code by adding more comments to code. This is important as this describes the code using human readable descriptions hence making the code easier to read and trace when debugging. The use of commenting will make code maintenance much easier as well. Furthermore, if I ever make a team to work on this project, I will need to include comments to make the code understandable to the other people working with me

Limitations/ Future Updates

My current limitations in relation to the old design are still the same. However, I have a clearer idea as to how to approach implementing ways to prevent these limitations of the program and allow more players to play the game and make the game more enjoyable and flexible both for the students and the teachers.

In the future, I would like to add quality of life updates to make the code more efficient, to add more accessibility for students with disabilities, add better maintenance for databases and add new features to improve the program.

Some of the features in these future updates will include changes to improve accessibility, such as compatibility with controllers. This allows for people who are not comfortable with using keyboards or have physical disabilities to still be able to engage and learn and develop their logical thinking skills. Furthermore, I would like to add colour blind settings to appeal towards people with colour blind disabilities and add the ability to change a background to a yellow colour to provide a better and more accessible format for people with eye disabilities and create more of a contrast between text.

Other things I would like to update is the efficiency of some of the code and how I approached the problem to solve it. For example, whenever I am ticking a multiple-choice question box instead of deleting and then re-instantiating, I should instead teleport the pre-existing tick object to the new position. This will take less lines of code to program and will save on lag since you are not creating

and destroying another object only moving one object. Small changes like these can greatly increase player enjoyment due to less lag.

Some other small features I would like to add are the models for the different gates to stop making the student rely on colours to differentiate the gates and rather identify the gates through their shape which will be recognised in exams hence helping them learn about logic gates and improving their grades. This change will also help colour blind people since some gates don't look similar anymore hence making it more accessible to everyone. I would also like to add a way to know when a light is being powered or any gates which I will be able to implement using the new models. I would like to also add a for fun section to add more replay ability as the student will be able to explore the possibilities of logic gates while also allowing the teacher to ask questions that are not already in the game and go around the class checking if they built the correct circuit.

I would like to also add better maintainability and use of the databases within the project as currently most data cannot be deleted and the data that is deleted such as a class record will not cascade and delete other students that have the classID linked to their account. Additionally, there is no way for teachers or students to delete their class so if a teacher or a student stop using the program their data is still stored which is no longer being used since it has no purpose and hence is breaking GDPR laws. To fix this I will add the ability to delete accounts, and this will cascade to ensure integrity within the databases.

Furthermore, my client suggested to allow the lasers to branch off and not just have a single line from a source of power. This will allow for new complex circuitry such as a half adder and a full adder to be made within this program. This is important as it will be an immersive way to teach a hard topic within A-Level.

Maintenance

I will maintain this program myself as I currently do not have enough comments to allow other people to be able to understand the code, I also do not need any other people to maintain this as it is a small project that is not super complex to deal with as a single developer. Despite this, I will try to add more comments to my code to improve maintainability and debugging. I will be able to update and maintain my code using Unity and C#. The use of an OOP language was intentional to allow me to be able to maintain my program easier since OOP promotes a modular, organized, and reusable approach to programming, which can help to make maintenance easier by reducing the amount of code that needs to be modified and minimizing the impact of changes on other parts of the code. It also allows me to debug the code easier to find what module of the code has an error.

I will make the updates mentioned in the Future Updates section above. There will also be typical updates to fix bug fixes and add small quality of life changes into the program and add any more features that my client wants

Client Sign Off

FlainK
To: 16SlamiskisN

Hi Naglis,

I am content with the evaluation of the program and I agree with the improvements that need to be made

Kind regards

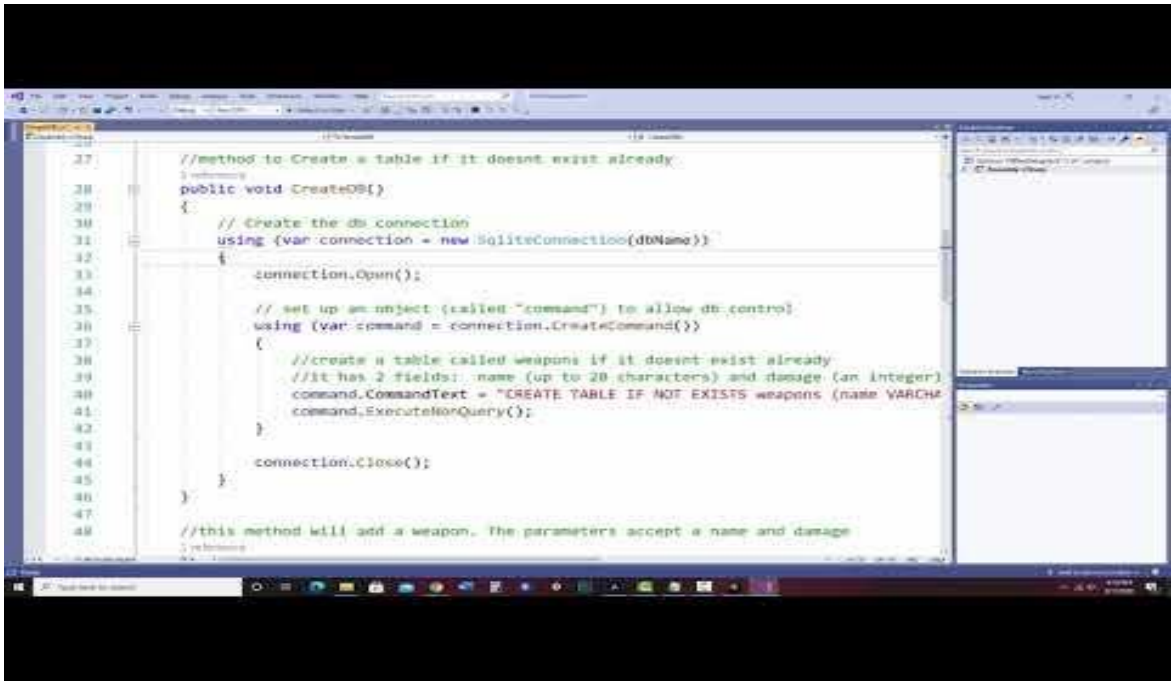
Mr K Flain
Subject Leader for Computer Science
The Saint John Henry Newman School

Links and References

[FIRST PERSON MOVEMENT in Unity - FPS Controller]

I used this tutorial to learn how to create a character controller.

[Using SQLite with Unity]



I used this playlist to better understand how to use SQLite within Unity