# Development diary

|  |  |  |
| --- | --- | --- |
| **Date** | **Progress** | **Modifications made** |
| 11/06/2024 | * Created unity project * Installed CUDA development kit from Nvidia | N/A |
| 12/06/2024 | * Set up the python machine learning requirements through a python virtual environment | Downgrade from python 3.12.1 to 3.10.11 to get packages to install correctly, required CUDA development toolkit |
| 13/06/2024 | * Player movement and aiming to cursor has been implemented * Informally tested that the player movement is set up correctly and performs properly | N/A |
| 14/06/2024 | * Implemented basic firing functionality for the player including: | N/A |
| 19/06/2024 | * Added main AI implementation, it can now fire projectiles after adding a separate projectile summoning script. | Pseudocode for AI changes |
| 20/06/2024 | * Implemented AI training bots into the game, able to record them but running back the bot in game is yet to be implemented. * Checked that training bot files saves properly after recording them * Figured out how to change the fire rates for the boss when training | N/A |
| 25/06/2024 | * Created a bot re-run script that reruns the inputs done that are stored in files * Added text mesh pro (TMP) for UI elements * Ensured that the inputs recorded and saved are able to be read back by playback system. | N/A |
| 27/06/2024 | * Changed recording system to record player velocity instead of player inputs as more reliable when playing inputs back * Changed script execution order so input script always gets loaded first * Updated Gantt Charts to be more up to date | N/A |
| 29/06/2024 | * Added a configuration file that expands how long the ml agent can train for * Created separate boss projectile script * Timer script now ends the training episode * Made the boss AI add a reward or punishment whether it hit a wall or player (-0.2 and 7 respectively) * Started training AI | N/A |
| 09/07/2024 | * Altered rewards given for AI feedback by increasing player hit reward * Trained AI further | N/A |
| 11/07/2024 | * Changed how boss AI reward is given, now given through an immediate raycast instead of when projectile lands * Trained AI further * Updated windows to latest patch | Criterion & FR03 |
| 13/07/2024 | * Added more bots that move in different directions * Made boss AI choose different bot to train on each session * Timer script resets systems to default after timer runs out * Trained AI on updated movement bots (movement\_test\_1-3) | N/A |
| 16/07/2024 | * Added player health bars and scripts to deal with player health running out * Added boss health and logic to reset the game when boss health runs out * Revised system that resets the game when time/health runs out | N/A |
| 17/07/2024 | * Added a preliminary pause and settings menu to the game to allow the user to navigate back to main menu * Added a main menu scene into unity * Fixed the settings loading * Offloaded resets to a proper game manager script instead of the timer script * Tested that game resets still worked | N/A |
| 18/07/2024 | * Added settings menu UI * Created settings save load system to load settings the user set up * Created a main menu for the user to navigate to settings, quit or into the game | Settings UI Design |
| 23/07/2024 | * Added volume controls to the settings * Tested that volume controls were correctly working by adding test audio and adjusting volume | N/A |
| 25/07/2024 | * Added a loading screen to the project to indicate that the AI is loading as it pauses the application while loading * Checked the loading screen was rendered when playing the game scene | N/A |
| 27/07/2024 | * Created sprites for the player and boss to fit the theming better for the project | N/A |
| 29/07/2024 | * Nintendo switch pro controller has been fully supported through adding the controller to the input map and changes in aiming code. * Created other sprites for the game including a boss health bar UI design, pause and main menu background, sprites for projectiles from the boss and player. | Pause and Main Menu  Player game UI |
| 31/07/2024 | * Changed colours on main title to match design for the main menu * Fixed bug where menu navigation on controller did not work properly * Updated settings menu so that it fits theming in the design * User acceptance testing was performed today * Testing table of various data was performed * After testing aiming on controller, changed it so that the aim indicator will hold last angle input from either controller or mouse (see testing table) | N/A |
| 01/08/2024 | * Performed user acceptance testing on v1.1 of the game * Added audio to the fireballs when you fire the fireballs | N/A |
| 05/08/2024 | * Updated internal documentation * Fixed several typos in the in-game settings * Created testing summary * Added range checks to inputs in the game | N/A |
| 06/08/2024 | * Did more testing * Added range checks to settings inputs | N/A |

# Design alterations

**Technical environment**

Original:

|  |
| --- |
| The environment the software will run in is on a computer that runs windows 11 OS. Installed there will need to be unity 2023.2.12f1 and python 3.12 with pytorch installed via pip. This is because pytorch is the framework used to support machine learning in the unity game engine and can be easily tested on windows OS as it is accessible. To make the AI learn more efficiently, a Nvidia RTX graphics card will be required as pytorch can more efficiently use a graphics card. I do acknowledge that there is a central processing unit (CPU) version of pytorch available, however the environment will run more efficiently if the calculations for the AI can be offloaded to the graphics card. The recommended amount of memory that a computer should have to run the application will be 16gb as a lot of data needs to be stored due to the AI and for game assets. |

Altered:

|  |
| --- |
| The environment the software will run in is on a computer that runs windows 11 OS. Installed there will need to be unity 2023.2.12f1 and python 3.10.11 with all python requirements installed via pip inside of a virtual environment. This is because pytorch is the framework used to support machine learning in the unity game engine and can be easily tested on windows OS as it is accessible. It will also require CUDA development kit to be installed prior to use CUDA instead of the CPU. To make the AI learn more efficiently, a Nvidia RTX graphics card will be required as pytorch can more efficiently use a graphics card. I do acknowledge that there is a central processing unit (CPU) version of pytorch available, however the environment will run more efficiently if the calculations for the AI can be offloaded to the graphics card. The recommended amount of memory that a computer should have to run the application will be 16gb as a lot of data needs to be stored due to the AI and for game assets. |

These changes come about as I have discovered that installing an older version of python was necessary because of errors that occurred. The Numpy module needed to be on a specific version that only existed around version 3.10, as well as some installation steps refusing to correctly install if it was in a later python version. It was also recommended in forms that 3.10 was more stable which is why it was selected. I also discovered during installation that to run on CUDA cores in an RTX graphics card the CUDA development toolkit was required to be installed which is why it is now required in the technical environment.

Original pseudocode for AI:

A screen shot of a computer program

Description automatically generated

Modified:

A screenshot of a computer program

Description automatically generated

The pseudocode has to be modified due to how unity structures the machine learning (agent) class that is being inherited. It separates out the observations and action methods into separate scripts. Additional functionality is also capable of being implemented with the OnEpisodeBegin method being used to reset the game in code, requiring the design to reflect this change accordingly. As opposed to before where we add inputs and perform the calculations in one function which, due to how the MLAgents classes were created is impossible.

Original settings menu:

A screenshot of a video game

Description automatically generated

Modified settings menu:

A screenshot of a video game

Description automatically generated

Due to limitations in modifying mouse controls, an option to change sensitivity has been removed. However, a drop down to change the quality of the graphics as well as a section describing the control layouts has been planned. This is due to sensitivity not making sense as the original mouse movement is not modified, and so, to make the game easier to use, a description of the controls will replace the sensitivity and an option to change graphics quality has been added.

# Evaluation criteria modifications

Original:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AI is quickly able to form decisions, not slowing down the game and is able to learn quickly too (efficiency) | The training and decision process of the AI has minimal impact on the framerate of the game and uses little resources | Some impact on framerate but not too noticeable to cause stutters or skipped frames | Noticeable stuttering and framerate drop is occurring due to the AI needing to be computed | User is unable to play the game as the solution freezes when attempting to process the AI decision or train it. |

Modified:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AI is quickly able to form decisions, not slowing down the game and is able to learn quickly too (efficiency) | The training and decision process of the AI has minimal impact on the framerate of the game apart from some occasional freezes that are unavoidable | Some impact on framerate but not too noticeable to cause stutters or skipped frames often that it impacts gameplay. | Noticeable stuttering and framerate drop is occurring due to the AI needing to be computed | User is unable to play the game as the solution freezes when attempting to process the AI decision or train it. |

The evaluation criterion relating to the AI has to be modified as well. During testing and training the AI I noticed that there were times that the AI caused the game to freeze occasionally even if there was only 1 instance of the AI running, there was not an increase in resource usage. However, to reflect this nature, the evaluation criterion now reflects this behaviour.

Original background:

A screenshot of a video game

Description automatically generated

Altered background:

Red lines on a black background

Description automatically generated

To keep with the higher detail pixel art that can be seen in places such as the level background, I felt it would be more fitting to set the image above to be the background that encompasses the main menu and pause menu, as it sets the theme of the game better than the original.

Original game design:

A diagram of a computer game

Description automatically generated

Modified:

A screenshot of a video game

Description automatically generated

To make the game more readable and information friendly, the health text has been moved onto the health bar and the timer has been added above. Outlines have also been added to the UI elements to make them highly visible.

# Contingencies

* Had to change the technical requirements. This was due to unforeseen issues with installing the files themselves, resulting in more being required for the technical environment.
* The nature of the game made it difficult to add a sensitivity section, so instead that area was repurposed into a description of the controls to ensure that they are even more usable.
* The background of the settings and main menu did not fir the theme of higher detailed pixel art and was adjusted to fit the theme and art style better.
* Some of the UI layout was altered to make it easier for the user to read and fit much better within the user requirements

# Screenshots

13/06/2024

InputLoader.cs: Assigns variables to be for player movement or to fire a projectile which is taken from a created input map. The upside of using a script is that it is very simple to add to the inputs and implement it in appropriate scripts.

A screenshot of a computer program

Description automatically generated

PlayerMovement.cs: The players movement has been programmed to simply apply the direction pressed and move in a specific direction.

A screenshot of a computer program

Description automatically generated

ShootProjectile.cs: Today the logic for finding the angle between cursor and player was written out and just applied to a temporary aim indicator

A screenshot of a computer program

Description automatically generated

14/06/2024

ShootProjectile.cs: This script handles all calculations in regard to the projectiles fired from the player. It first calculates the angle to the mouse (and soon to be controller as well). It then checks if the fire button was pressed and launches the projectile from the specified spot.

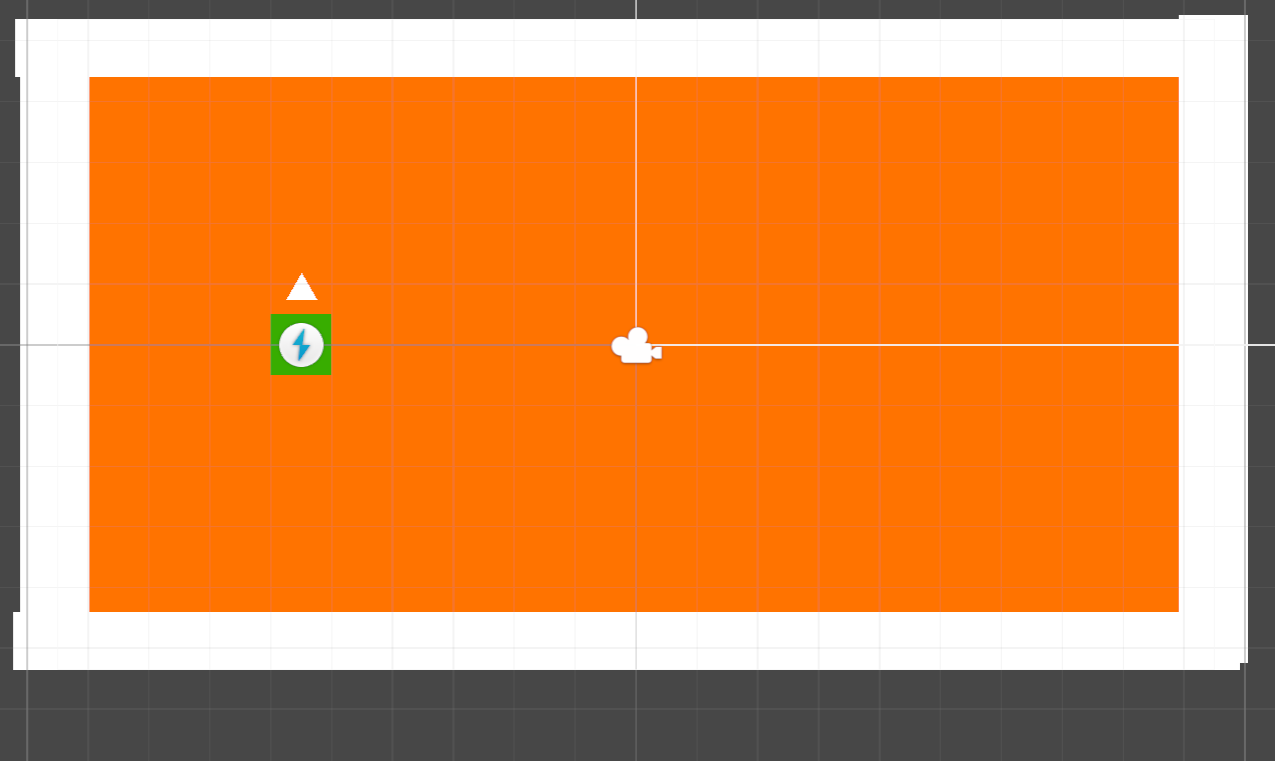


Projectile.cs: The work that has been done here is to do with the logic each projectile follows when spawned into the scene. As shown below right now the script only applies velocity to a given direction from the script that provides projectile calculations.

A screen shot of a computer code

Description automatically generated

After all this code has been implemented, the player now has a triangle to indicate where they are aiming at as well as the ability to move freely in the white boxed area.



19/06/2024

BossMLAgent.cs: This script handles all main decisions. Right now, what has been implemented is the logic for inputs into the AI (CollectObservations) and also to output a direction to fire the projectile in. This has been done so that FR03 can be achieved.

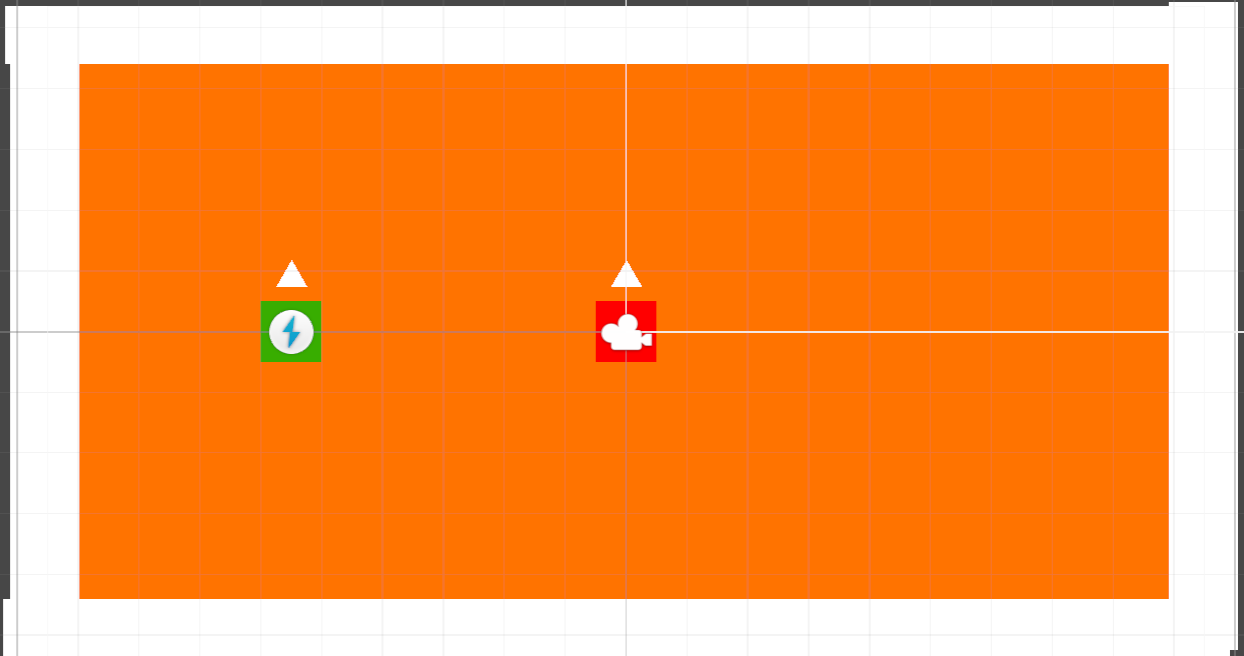
A screen shot of a computer program

Description automatically generated

BossProjectile.cs: This script is similar to the player projectile.cs script as it provides a way for the boss to summon a projectile from within the BossMLAgent.cs script. This is because it provides a class that can be referenced as an instance from the other script. A screenshot of a computer program

Description automatically generated

A boss indicated by a red square was added with an arrow/triangle to indicate where they are aiming for debugging purposes. The boss has the BossMLAgent.cs script attached that handles the aiming logic.



20/06/2024

RecordTrainingBots.cs: This has been created to set up the training bots to train the machine learning boss to target them. It records players movements into a plain text file to be re-ran.



25/06/2024

RunBot.cs: Runs back the files saved by the RecordTrainingBots.cs script. So, the AI can play back the players inputs that have been pre-recorded.

A computer screen shot of a program code

Description automatically generated

Timer.cs: Displays a timer at the top of the screen with the number of seconds remaining in the fight. Will in future reset the training environments so the next round can be run but stops recording the player movements when enabled for training.

A computer screen shot of a program code

Description automatically generated

27/06/2024

RecordTrainingBots: Altered the code to ensure that the recording is velocity based instead of input based. This is due to issues with replicating the inputs from the player. Velocity based is far more reliable.

A computer screen shot of a program code

Description automatically generated

RunBot.cs: Added existence validation to check if the input file exists as well as ensuring that velocity is input into the bot instead of adding force from inputs.

A computer screen shot of a program code

Description automatically generated

29/06/2024

BossMLAgent.cs: Changes made were to clearing the projectiles fired by the player each time a training episode ended so that the reward won’t carry over into the next episode. When a new training episode begins it calls ClearProjectiles that loops through all projectiles in the scene and deletes them.

A screen shot of a computer code

Description automatically generated

Timer.cs: Existence validates that a bossMLAgent instance exists to reset the reward when the timer runs out. Done to ensure the AI can progress in its training.

A screen shot of a computer program

Description automatically generated

The machine learning training has been scaled up to speed up the time training takes.

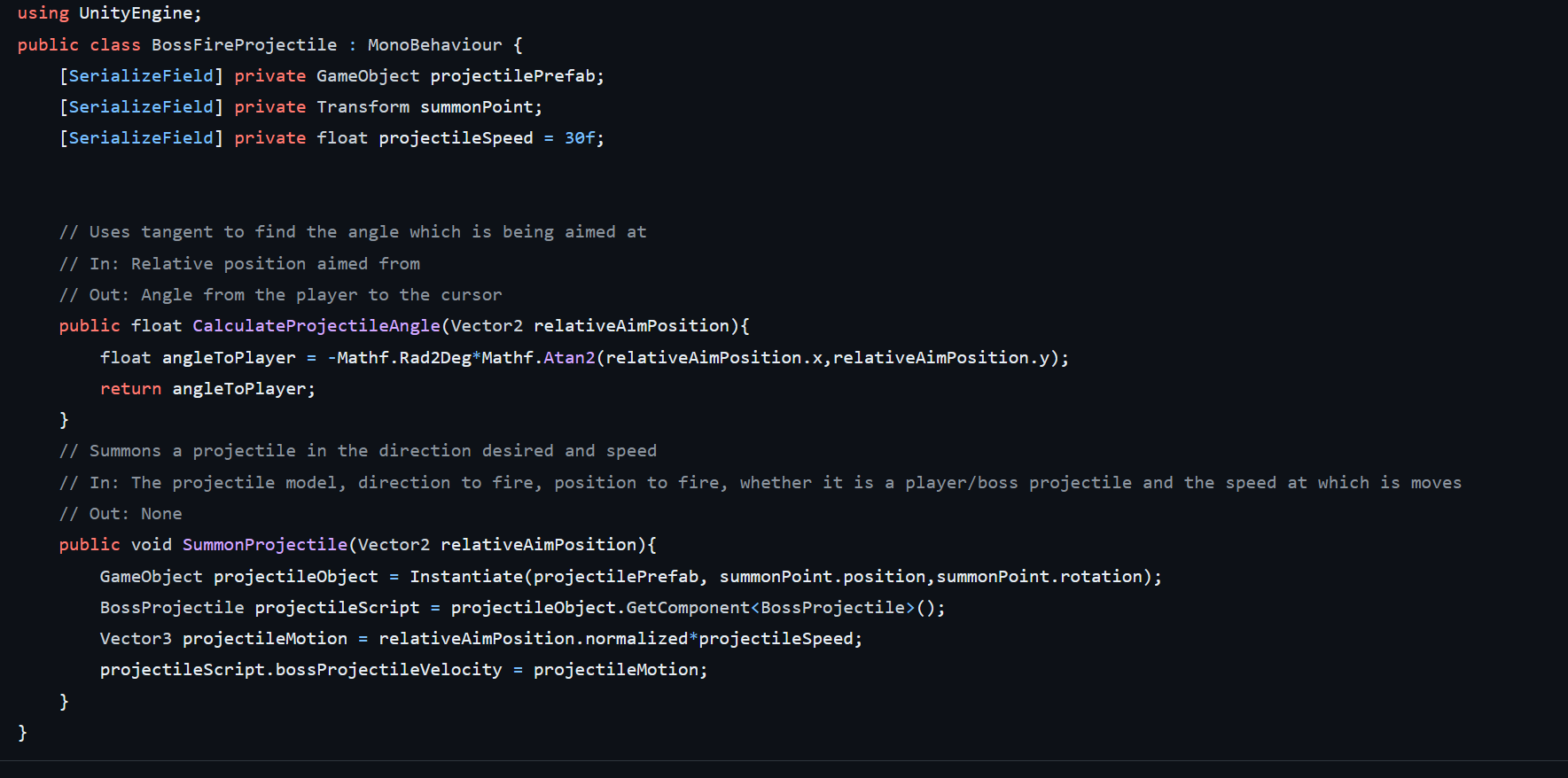


BossProjectile.cs: Checks whenever the projectile hits an object such as wall or player and then now adds the reward to the boss AI after it has validated that it exists.

A screen shot of a computer program

Description automatically generated

BossFireProjectile.cs: Handles the calculations for summoning the boss projectile logic. Separates out complexity from the player fire projectile. Is referenced by the BossMLAgent script.



11/07/2024

BossMLAgent.cs: Removed a few observations to make it easier for the AI to understand as well as implemented raycasting for the reward so it can be more accurate.

A computer screen shot of text

Description automatically generated

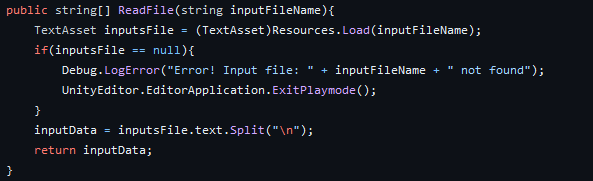
BossProjectile.cs: Removed the AddReward as it would interfere with the raycast rewards.

A computer screen shot of text

Description automatically generated

13/07/2024

RunBot.cs: The changes made were to put the file replay logic into a separate function that returns a string array if the file passes an existence check. This is then run in the timer script to load the inputs to play back.



Timer.cs: Changes have been made so that the training bots properly reset when a new training episode begins. The timer has a method that iterates over all the bots that reads a new training file and resets the position of all training bots.

A computer screen shot of a program code

Description automatically generated

BossMLAgent.cs: When the Timer script finishes countdown and calls a new episode then the script calls the reset method in Timer.cs, since the timer is a global script instead of attached to multiple objects.

A screen shot of a computer code

Description automatically generated

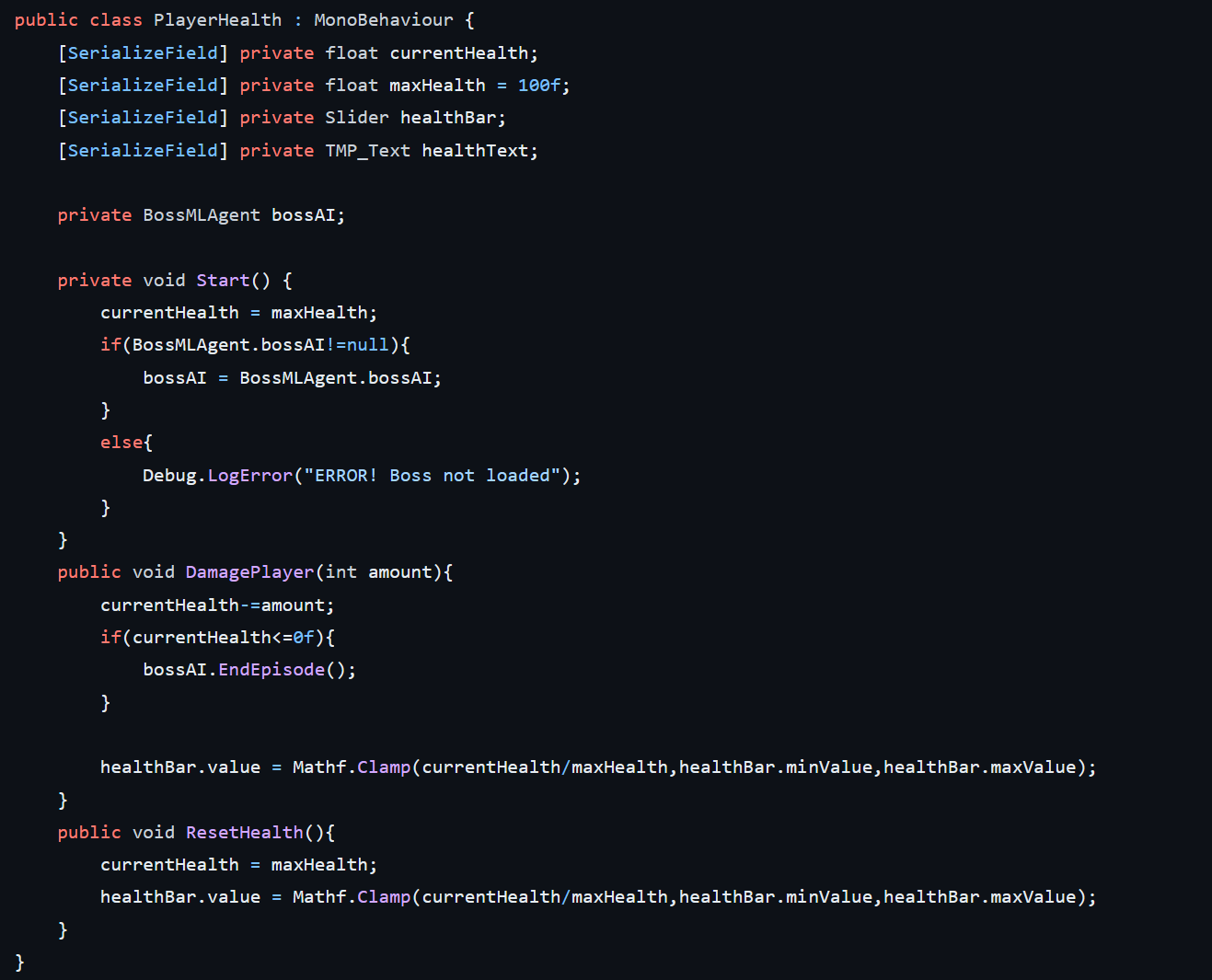
Due to the code now being capable of resetting the training bots’ position locally and without any bugs this time. The same setup still applies and does not need to be changed.

A screenshot of a video game

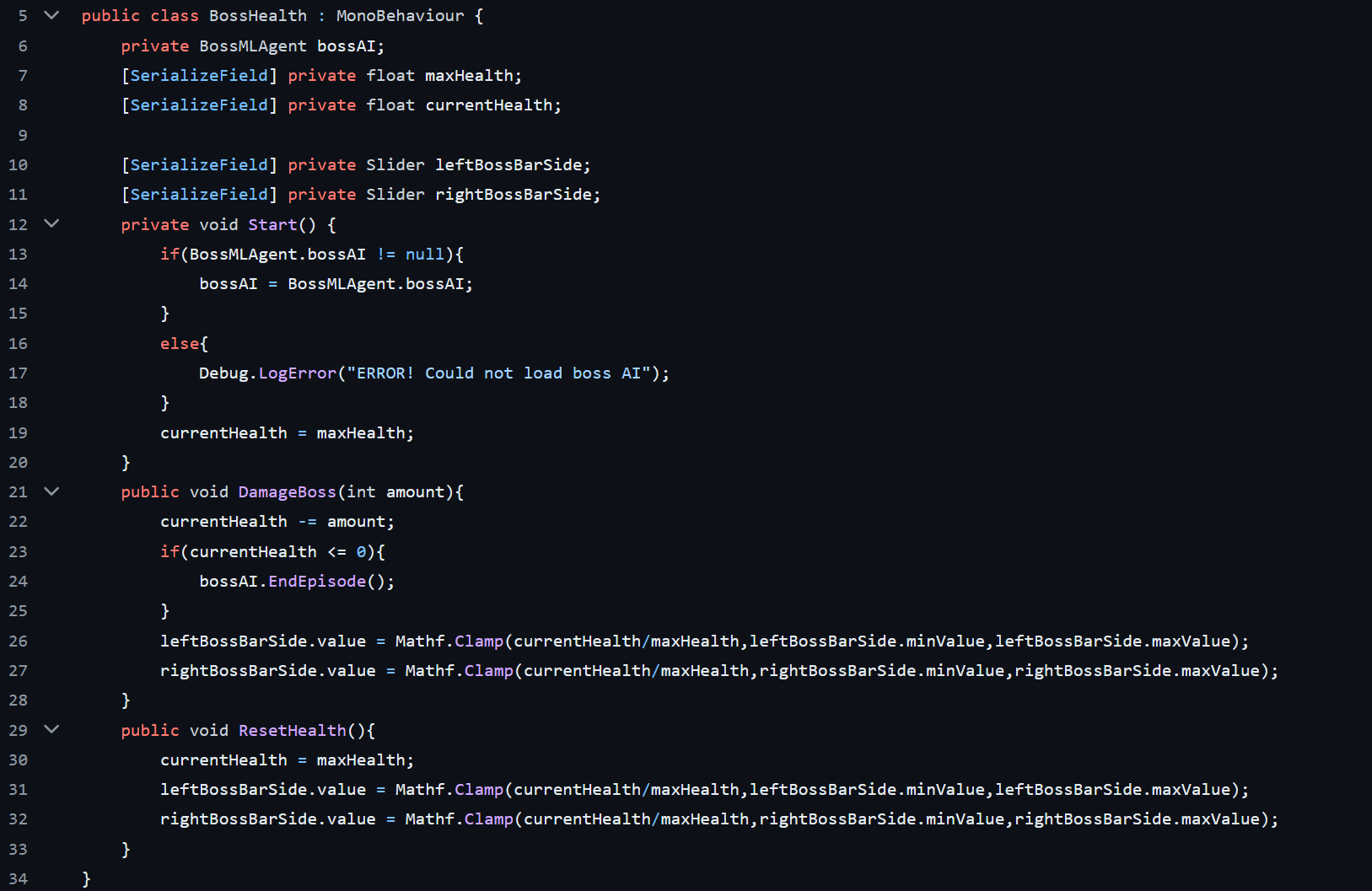
Description automatically generated

16/07/2024

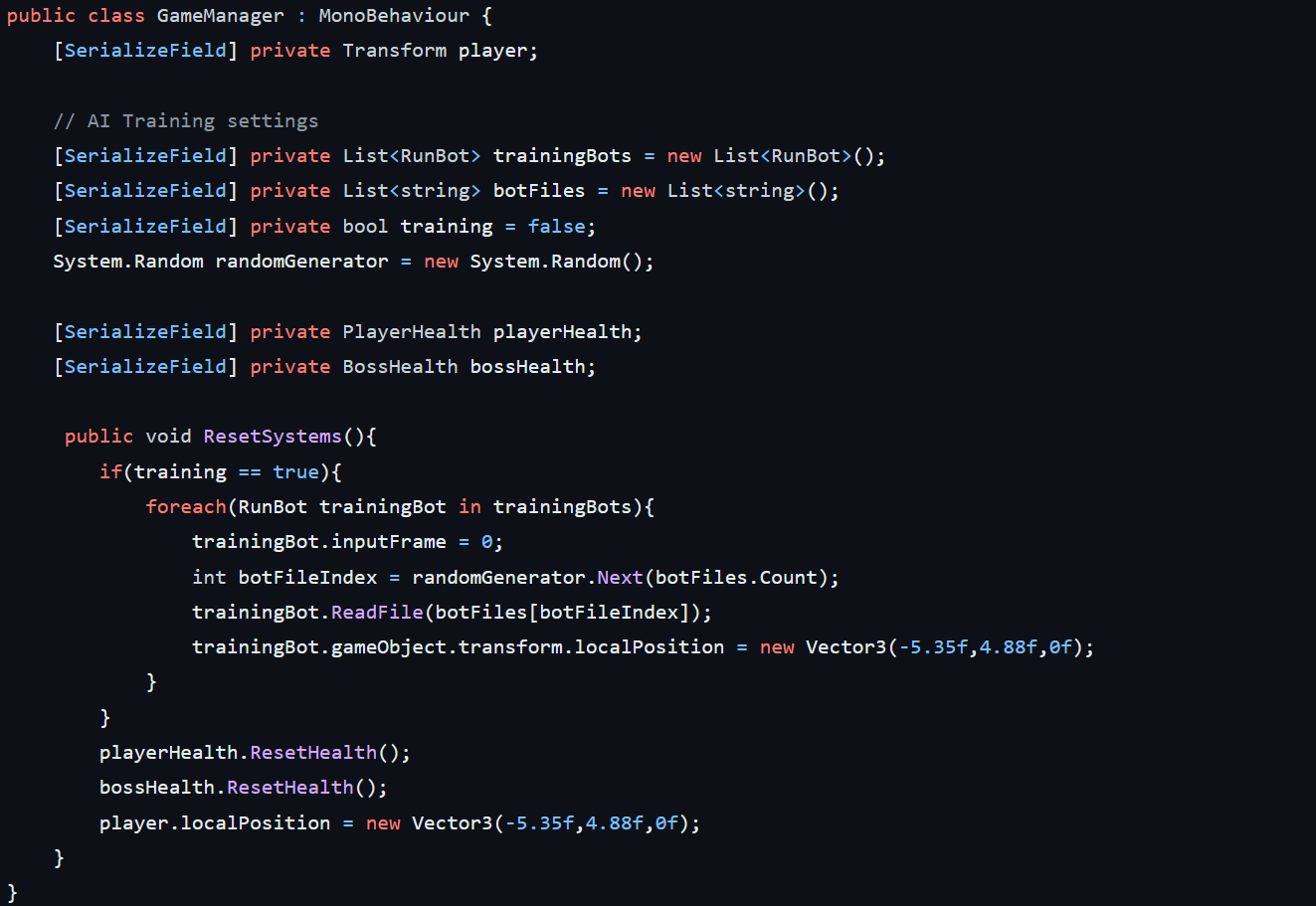
PlayerHealth.cs: To handle the player being damaged by the boss’s projectiles, this script is referenced to deal damage to the player but also reset the scene by ending the episode. Through a range check the EndEpisode method in the boss AI invokes a system reset from GameManager.cs.



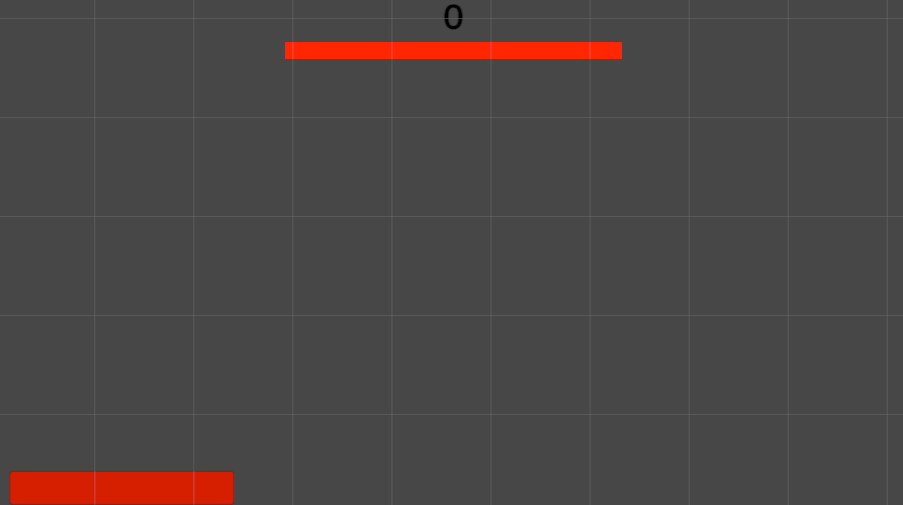
BossHealth.cs: This script was created to be in charge of damage the player would deal to the boss, and accordingly perform range checks to ensure that the bosses health is decreased in the code and on the health bar visual. If it does reach 0 health points, then the game is reset currently by triggering the GameManager.cs when the AI episode ends, which is done after checking the boss AI exists.



GameManager.cs: This script is invoked from the EndEpisode method in the BossMLAgent.cs script and resets all necessary systems to default. It saves having to spend resources by simply reloading the scene and can be modified easily to add other systems.



Player UI layout: Below, the boss health bar can be seen at the top with the players health in the bottom left, similar to the design. Above the health bar the timer can be seen to indicate how long the player has left to defeat the boss.



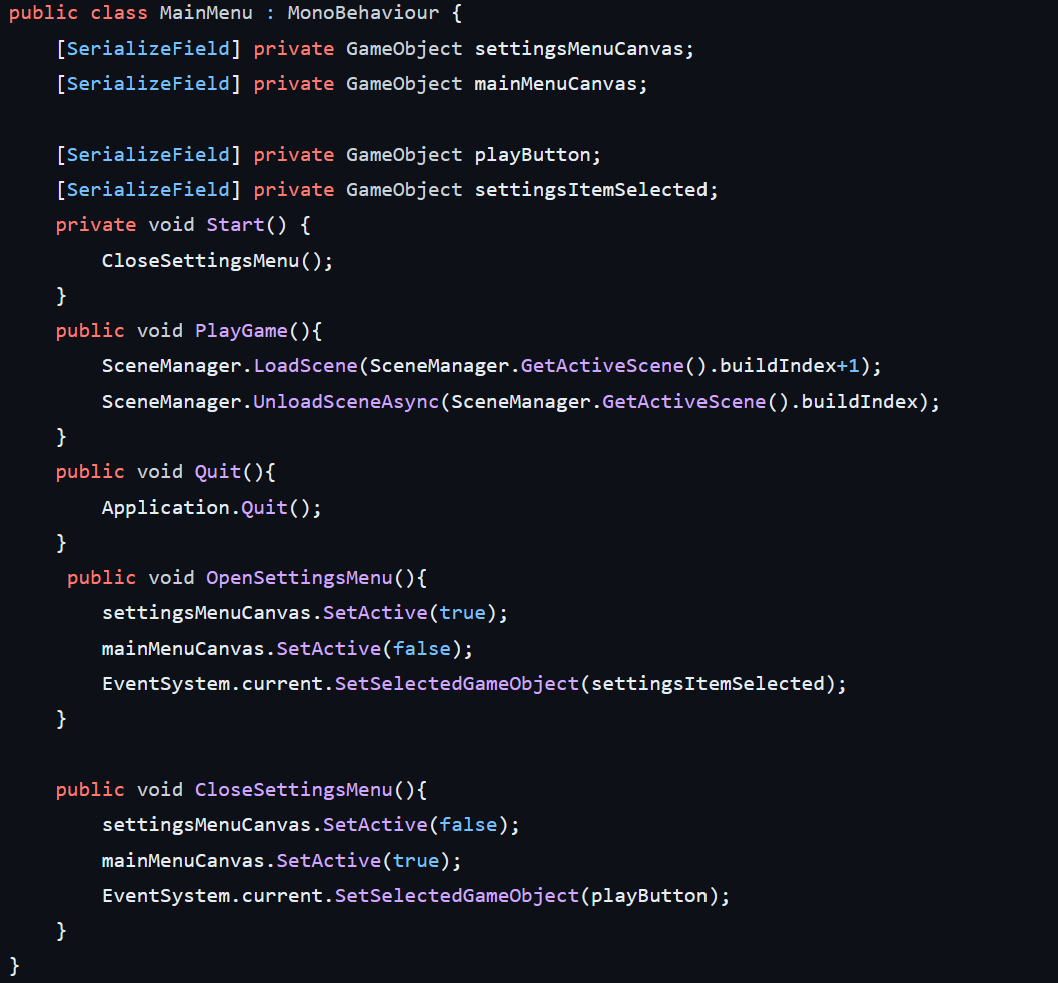
17/07/2024

PauseMenu.cs: The pause menu implementation first stops all the player logic from running when the game is paused, as well as set the time that passes to be 0. From there it then opens a pause menu (that can be seen below). We then can load into the main menu if it exists or into the settings menu. It has been informally tested that the game resumes properly from different paths.

A screen shot of a computer code

Description automatically generated

MainMenu.cs: To handle the button inputs in the main game, this script was created. The relevant functions open the menus, quit the game or go to the game scene.



Following the design plan for the main menu, the buttons are allocated appropriate names that correspond to the functions in MainMenu.cs



18/07/2024

LoadSettings.cs: To ensure the player has their settings saved and loaded, the functionality was implemented in this script. To add the relevant data types for each setting, a class at the bottom called Settings is created that contains the data to be saved and loaded. The LoadSettings class handles applying the settings, however when testing there was a bug with the .value variable in the drop-down menus that caused the code to stop running. To get around that, the code uses SetValueWithoutNotify, which relieves the issue as I believe it overwrites the method execution in the call stack. The SaveLoad class is responsible for saving and loading, if it cannot find a settings file it returns null, which the LoadSettings class performs an existence check and creates a fresh settings file if it could not be found, adding to the security of the files as it ensures that if the settings file is destroyed then a new one will always be present.

A screen shot of text

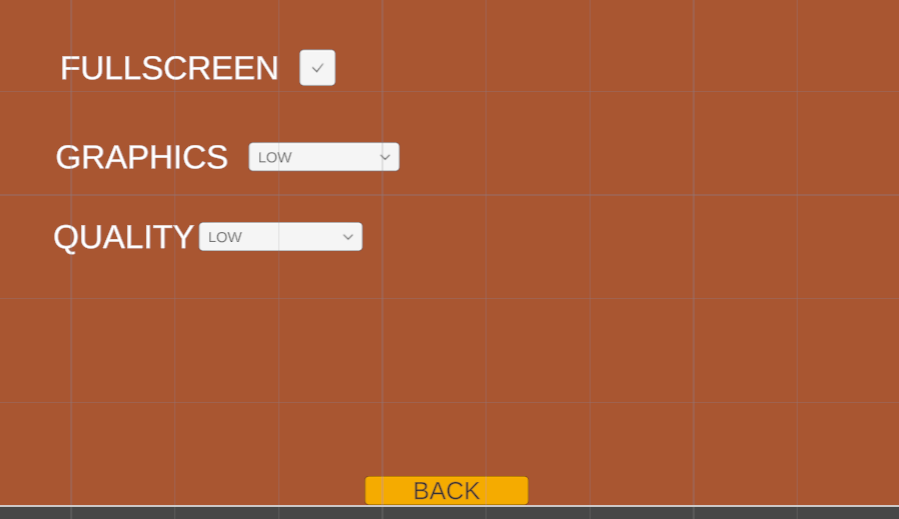
Description automatically generated

SettingsMenu.cs: To change settings in the menu, these methods were created to set specific functionality on different settings elements. First it also checks the settings are loaded properly through an existence check and modifies the settings and saves them whenever a setting is changed.

A computer screen shot of text

Description automatically generated

This results in a settings menu that looks like this:



23/07/2024

SettingsMenu.cs: A volume slider was added to adjust the output sound to the user preferences, similar to other methods, the volume is then saved after it has been adjusted.

A screen shot of a computer program

Description automatically generated

LoadSettings.cs: In the settings class, we add a floating-point number for the volume so it can be saved and loaded later, and then we apply the update to the audio mixer that controls the volume of the sound.

A screen shot of a computer program

Description automatically generated

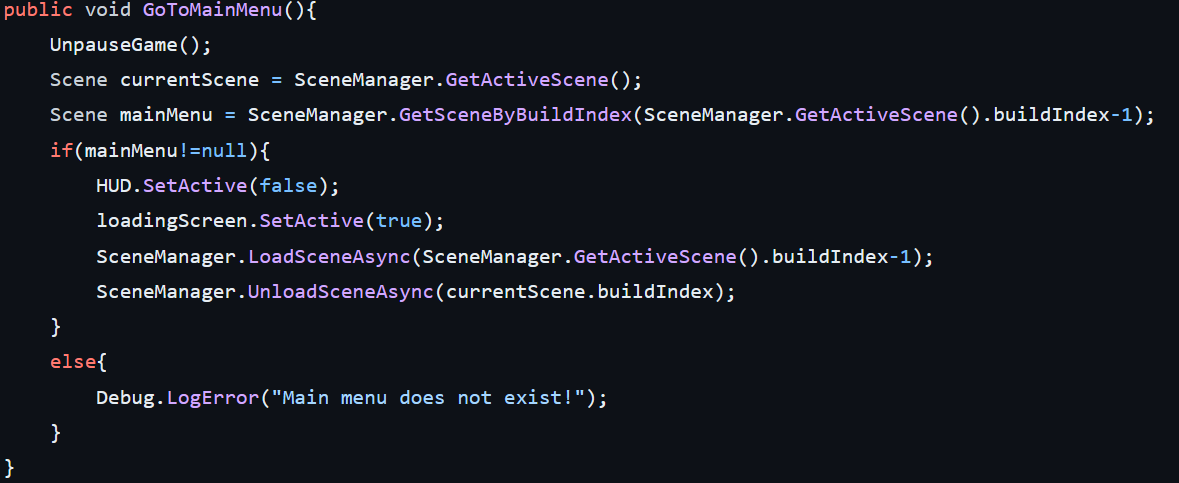
The settings menu now looks like with the volume added:

A screenshot of a computer

Description automatically generated

25/07/2024

PauseMenu.cs: Since loading between scenes can take some time, a loading screen has been added that is set to active as the main menu is being loaded in and the game scene is being unloaded.





MainMenu.cs: The reverse is true as the loading screen needs to be added as the AI takes some time to load up, so to indicate to the player this is the case a loading screen has been added.

A computer screen shot of a program code

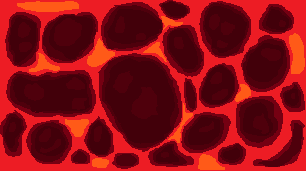
Description automatically generated

This is what the basic loading screen will look like. It provides indication to the player that the game is currently being loaded. Since the machine learning does take the most time when loading into the scene, it is displayed in the loading screen.

A grey screen with white text

Description automatically generated

The background below has been created as well to provide more theming to the volcano theme of the game.



27/07/2024

I feel that incorporating themes from another theme that was suggested to potential users (that being an office theme) feels fitting despite it going against the volcano theme. It also provides some intrigue and contrasts against the lava background to easily identify the important components of the game. The green that can be seen on the right hand is also to indicate the colour of projectile the player fires.

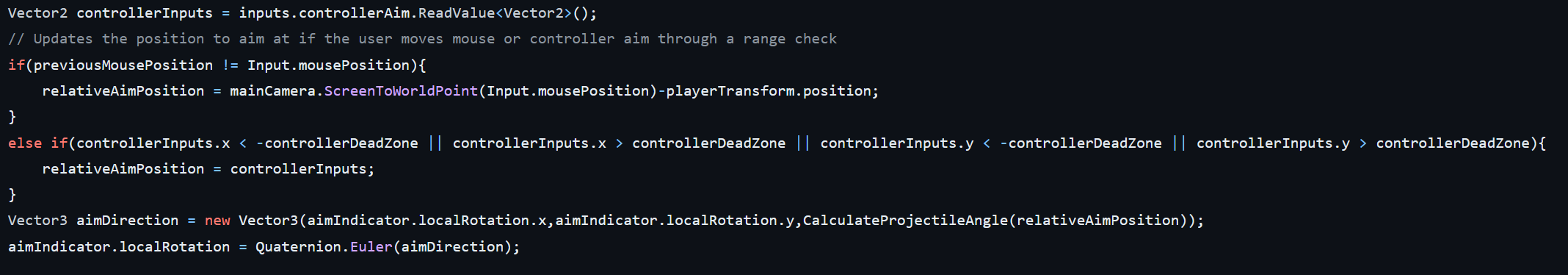


To make the boss feel like a big bad evil guy, I chose to represent him as a wizard. This not only represents the decisive nature of the machine learning underneath as some sort of magic, but also visually stands out to the player and gives context for what to aim for.



29/07/2024

FireProjectile.cs: To ensure that proper aiming works. We first make sure that input from the controller is outside of a dead zone through a range check or if the mouse position has moved, if either of the inputs has been altered then we update the position to aim to.



InputLoader.cs: To facilitate inputs being added to make the controller fully integrated into the program, they are also loaded in by the input system.



To further invest in the volcano theme, the borders have been made to look like dark, ash covered rock that meets the lava. Creating a barrier visually for the player as well as fitting in with the volcano theme. It is also tile-able so it can stack together without looking too repetitive or out of place.

A black and red background

Description automatically generated

To make the boss health bar stand out more, this design was produced to make it appear more villainous with the points on the end while leaving room for the health bar fill in the centre portion.



To correlate the boss’s colours, similar coloured projectiles have been added to the boss projectile.



As the projectiles need to be differentiated then two projectiles, the player projectiles are coloured green to indicate that they are from the player.



A main menu theme that fits volcano is required. I created a background that fits the volcano theme with lava tubes/cracks.

Red lines on a black background

Description automatically generated

The information section is created to indicate how much time is left on the clock before it runs out. It is coloured white to stand out from the dark background of the game scene.



31/07/2024

GameManager.cs: To make sure the player knows when they have beaten the boss, a method was created to activate the win screen

A computer code on a black background

Description automatically generated

BossHealth.cs: To activate the win screen, we check if the boss has been defeated and then invoke the method in the game manager script.

A black background with white text

Description automatically generated

FireProjectile.cs: After testing the aiming, the aiming snapped back to the mouse when not using the aiming joystick on the controller. This has been remedied and will now stay in a specific orientation unless a new input is added.

A computer screen with text

Description automatically generated

LoadSettings.cs: To ensure that loading quality will be within a specified range, mathf.clamp is used when loading in settings as a form of range checking to ensure that all settings are within bounds.

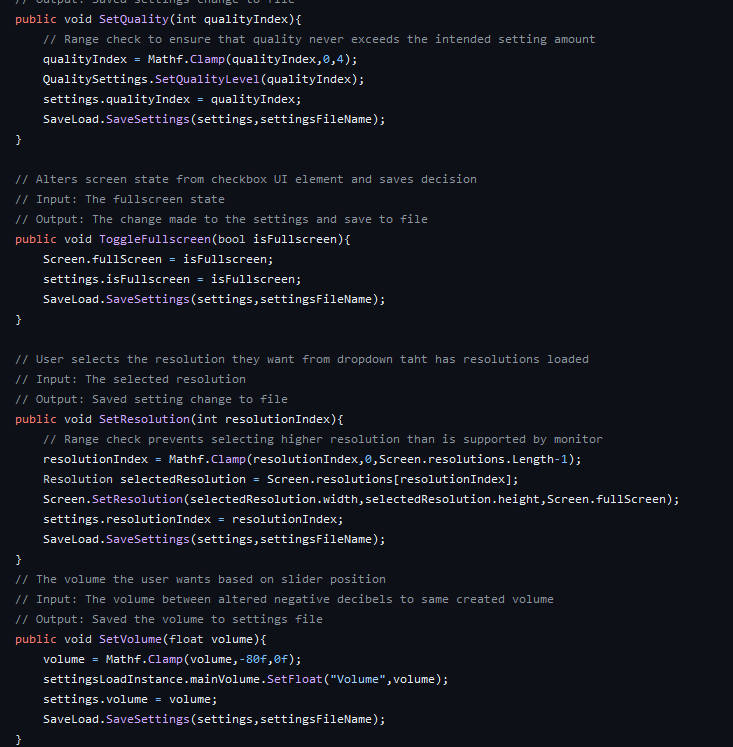
A screenshot of a computer program

Description automatically generated

05/08/2024: Today comments were added or updated in all files to ensure that internal documentation exists inside all scripts to make more sense to users (see codebase for further updates)

06/08/2024:

SettingsMenu.cs: Added range checks through using Mathf.Clamp which automatically prevents the value from passing outside of a certain range in the SettingsMenu.cs file



# Machine learning training progress

Training scene setup

A screenshot of a computer game

Description automatically generated

There are multiple “environments” that are separated from each other. This provides a faster training process as multiple different AI configurations can be trained at once. The red is the AI and green block is the training bot that will replay actions to train the boss.

Test 2 (29/06/2024)

Test 2 was a 33-minute run. This comes right after a quick test to confirm that the unity mlagents was running. Throughout the run the AI showed no meaningful improvement and only fired at the walls. This training test was conducted using projectile reward feedback, which likely confused the AI as it took a while before the projectile hit an object to get any feedback.

Test 4 (09/07/2024)

A graph with lines and numbers

Description automatically generated

Test 4, similar to test 2 produced little meaningful improvement in training despite altering rewards to give a higher reward when the boss projectile hits the player. This confusion is likely caused by the projectile having to move to the target and then the training AI not being able to correlate the position it is aiming and the training target location.

Test 7/9 (11/07/2024)

A graph with lines and numbers

Description automatically generated

The tests for training after test 4 were done using raycast technique to get rid of the confusion caused by the projectile taking time to move. The downside is that when the projectiles do fire, they will be less accurate as the training is done instantly.

Test 7 (orange) was done using SetReward method provided by the mlagents documentation, which set the reward for the training episode instead of adding them, this was done as a control to ensure that mlagents would train properly. As can be seen, the combination of raycast and set reward aided the AI in improving aiming until it could no longer reach any more reward by aiming at the target. It is also important to note that the training time only took around 10 minutes to complete as well.

Test 9 was changed to AddReward which adds up the reward for a training episode before the countdown finishes in the timer script. After this, we can see that test 9 (blue) followed similarly to test 7 and was only trained for around 5 minutes with constant improvement.

Movement test 1-3 (13/07/2024)

A graph on a black background

Description automatically generated

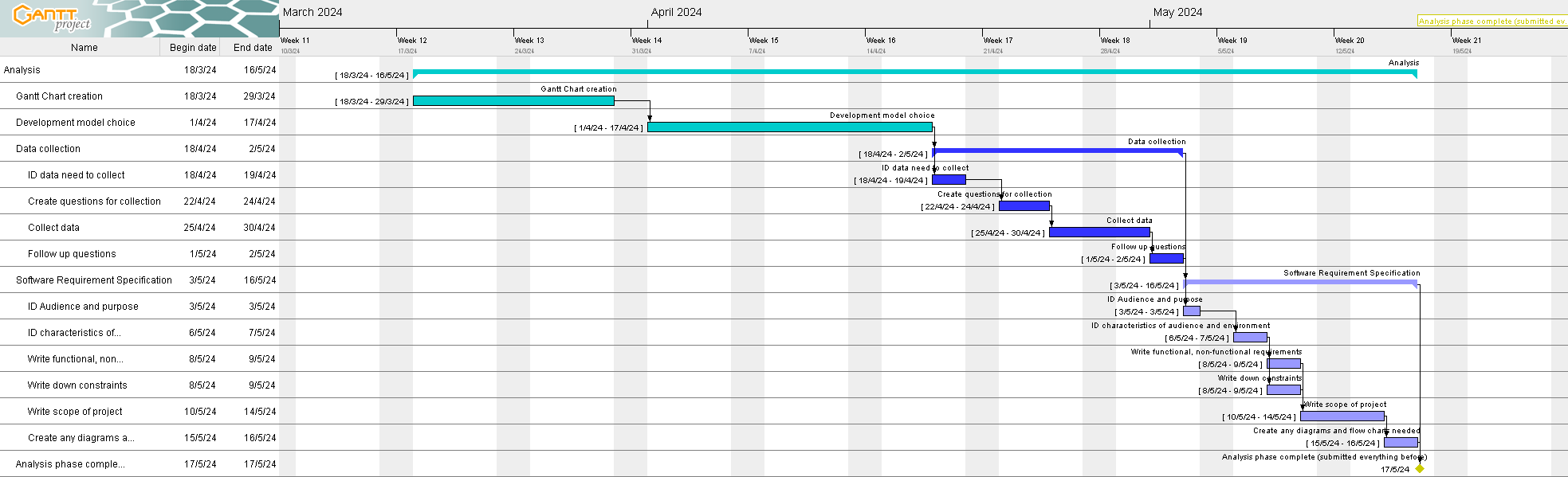
After successfully training on a stationary bot in test\_9, training was changed to a moving bot to reinforce the direction the boss should shoot in. Test 1 (pink) was to check that the AI can train on up and down movement.

Test 2 (orange) was expanded to a larger set of movements with clockwise, counterclockwise and movements on the other side of the boss included. However, this test was cancelled due to complications with the training bots not resetting to their original position. Even though it indicates that training was going as expected with the reward increasing, to ensure that the AI makes the best decision this problem needed to be resolved.

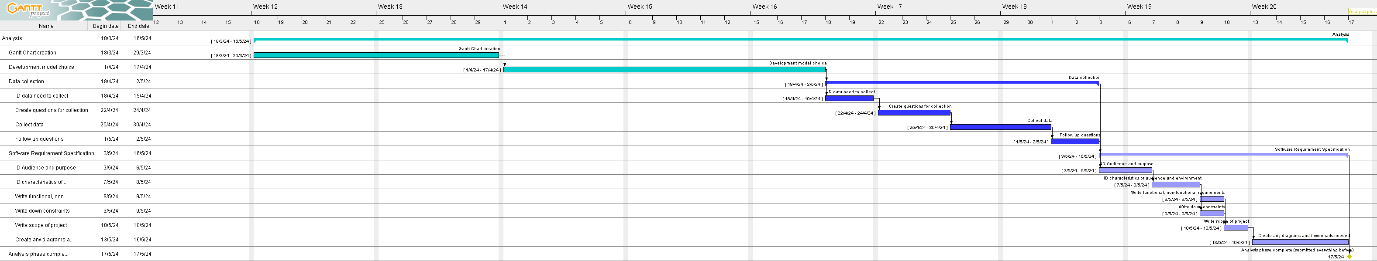
Movement\_test\_3 (green) was run after all problems were resolved by iterating over all training bots manually to reset their position. This test was run for over an hour and a half, but tapered off in improvements, due to the accuracy this training session will be used for the game.

# Gantt chart alterations

**Original analysis Gantt chart**

****

**Modified analysis Gantt Chart**

****

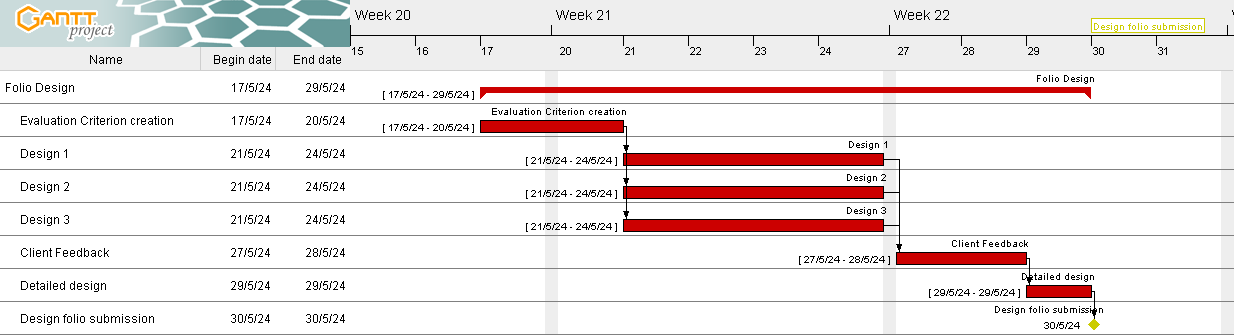
Over the course of the analysis, very little was changed as the deadlines were stuck to with a lot of the work being done in order seen in the Gantt Chart. However, there were a few modifications with the SRS. A modification made included the length of time to write the scope being shorter than what was expected. The diagrams and charts creation also took a while to complete compared to the original design as well.

**Original design Gantt Chart**

**A screenshot of a computer

Description automatically generated**

**Modified design Gantt Chart**

****

Over the design period. Some of the expected times were changed. This extends to the length of time feedback from clients were taken to improve the designs. This in turn shortened the time the detailed design was taken across. As well as this lengthening, the time spent to create the initial designs themselves.

**Original development plan (before 11/06/2024)** A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Date of modifications** | **Gantt chart** |
| **11/06/2024** | Due to exams for year 11 and the king’s birthday, starting the development phase has been pushed back to the 11th of June to start. This has pushed everything back by several days which means that some work will have to be cut in order to meet the deadline. |
| **14/06/2024** | The time to set up movement for the player was shorter than expected. I also managed to implement the player firing projectiles within this. We also removed validation as that will be handled during the testing phase. |
| **19/06/2024** | The implementation of AI has taken a lot less time than expected as well, this is due to the expectation of complexity being a lot higher than reality. However, the rewards implementation is yet to be seen. I also removed the validation as that will be done during testing. |
| **20/06/2024** | Some modifications have had to be made to the machine learning and are expected to be made until 25th of June. However, an initial training bot’s system has been created with the bots recording the players inputs. Since we are in the training creation task, it is difficult to say when it will end. |
| **25/06/2024** | The setup for training will occur during this week, as well as continued modifications to the boss AI. This is to ensure proper training. I will also be going camping so I have to take time out of development to do that. Also implemented a playback of the recorded inputs. |
| **27/06/2024** | Similar to the 25/06/2024, we are still in the modification and training setup tasks. Changed from input to velocity based bot inputs as it was more reliable when playing back the bots. |
| **29/06/2024** | Start AI training is meant to be on the 29/06/2024. However Project Gantt refuses to allow tasks on the weekend. The end of the forseen modifications happened. The first training tests were performed on static training bots that did not move. The AI did not improve much, suggesting further changes that will be performed apart of the AI training task. |
| **11/07/2024** | After getting back from camping, AI training resumed, however the UI has not been developed as of yet so is being pushed back as well as art work. The AI training has involved several changes to how the reward is handled, most notably changing to raycast rewards instead of projectile rewards as it confused the AI. |
| **13/07/2024** | AI training has finished today. The rapid training of the AI was not expected but it has now been finished. |
| **16/07/2024** | After training the AI, work has begun on the UI elements. This is expected to take a little time but not too long. It will only take a few days due to unity having default UI elements that can be used and altered to fit the design. |
| **17/07/2024** | The player game UI was finished and game reset functionality was implemented to match the timer in a proper game manager that handels when player, boss or timer is depleated. Concurrently, a settings and pause menu can begin being created as well as a main menu scene that is expected to be worked on. |  |
| **18/07/2024** | Game reset functionality only took 1 day to be completed. As well as this the unity menu did not start work until today with the scene being created yesterday. The settings save load has been created today after figuring out issues with the value of settings not being applied correctly. The menu UI is also starting to look like the design. |  |
| **23/07/2024** | Today the settings menu got worked on with the volume setting being added to the scene. Since development has only occurred on the settings menu, it has been added as a different task that will be worked on over more time. |  |
| **25/07/2024** | Finished up the programming for the user interface as well as UI element creation, and have started on creating artwork today. It is expected to go over several days. A background for the main game scene has also been created over the past few days and updates to the settings menus as well occurred |  |
| **27/07/2024** | The boss and player sprite are still being worked on today with them being finished but the white background yet to be removed. |  |
| **29/07/2024** | Other art work such as UI borders, projectile art and settings/main menu UI background were worked on during this period. Also a negligable amount of work was done on the projectile scripts to ensure that the projectile sprites rotated properly, but not enough to warrant a task being created as it was only a few lines changed. |  |
| **31/07/2024** | Other artwork elements have been completed. Testing can now begin with user acceptance testing and a testing table being used during the testing phase. The controller was a massive issue and has already been fixed as it would impede the user acceptance testing. |  |
| **01/08/2024** | It is expected that bug fixes will be performed as well as testing during the second week with user acceptance testing continuing throughout the week. |  |
| **05/08/2024** | Version 1.1 of the program has been created with several bugs being fixed and further user acceptance testing occuring as well as more testing within the testing table. |  |
| **06/08/2024** | Testing has continued along with further touch ups to the development diary. As well as this enhanced internal documentation has been added. User acceptance testing has been finished with no more formal testing conducted on users. I would still like to perform one last test in the testing table. |  |

# Security of files

To securely save files, I have decided to use a private GitHub repository to store files. The repository is private to ensure that no unauthorized users change or modify the software, and potentially harming the data integrity within the backup itself. Even if this scenario did occur, or others similar to it, GitHub is a combination of incremental and differential backups that ensure these changes can be rolled back to a previous version.

Furthermore, GitHub requires valid login information through multi-factor authentication. As it at bare minimum needs a username and password. However, confirmation to login is also needed through the GitHub mobile app in order to proceed to the repository. These authentication techniques further reinforce the integrity of data stored within the project as they are preventative measures against unauthorized access.

Locally as well, I am ensuring to stay up to date with the updates windows puts out on Windows 11, as it minimizes the risk of corruption and security vulnerabilities that are potentially present within Windows 11 Operating System or on my device. Being up to date also ensures that the software is stable and won’t cause harm to the environment it is running on.

My aim is to continue to update windows when notified of any releases within the development phase of the project, and to back up by pushing any changes to the project to GitHub as soon as a session of coding is finished.

To further ensure the security of files locally, the settings file has an existence check performed on it to ensure that the settings can be applied and read back. If the settings file does not exist, then the LoadSettings.cs script will create a new settings file with default settings applied. This detection provides further security to files as there is tolerance if the settings data is destroyed, allowing for normal operation of the application.

A screenshot of a computer program

Description automatically generated

# Testing table v1.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function tested | Test data | Expected results | Actual result | Fixes |
| AI tracking player | Players moves upwards | The ai tracks to the players position when they can next fire/ aim indicator pans upwards | Aim indicator pans upwards towards the player | None  (see machine learning training section for more details) |
| In game Movement | Keyboard move upward (w key) | Character in game moves upwards when the w key is pressed | Character does move upwards, sometimes when running unity, the internal speed does change however | None, |
| In game movement | Keyboard move right (d key) | Character moves right when d key is pressed | Character does move right when d key pressed.  The character speed does change which is not intended when project is re-opened | None |
| In game movement | Keyboard move down (s key) | Character moves down when s key is pressed | Character does move downwards.  Reopening the project alters the speed the player moves at however. | None, |
| In game movement | Keyboard move left (a key) | Character moves left when a key is pressed | Character does move left.  upon reopening the unity project, for some reason the movement is either sped up or slowed down sometimes | None |
| In game movement | Keyboard input (q key) | Nothing happens | Nothing happens | None |
| In game movement | Controller left joystick movement | The player moves left in game, corresponding to the action on the joystick | Attempting to move the player is not responding | Added support for the controller in the input map under the movement input action |
| Player aiming | Controller right joystick moved far right | The aim indicator follows along with where the firing | Aim indicator follows when the right joystick is moved, however snaps back to the cursor when no input is given | Create a Vector3 that is |
| Player aiming | Mouse position moved right | The aim indicator follows the cursor to the right | Aim indicator tracks to mouse position | None |
| Settings save load | No settings.json file exists in project | New settings file is created | New settings file is created | None |
| Settings save load | Value is changed in settings menu (e.g. volume db is moved from -40 to 0) | Volume is changed to 0 in game and is saved to the settings.json file | The audio volume is changed and saved to the settings.json file. | None |
| Settings save load | Change full screen state | Nothing happens as no implementation exists within the unity editor | Nothing happens since nothing happens within the unity editor | None |
| Menu navigation | Controller left joystick | Controller navigates up and down menu elements properly | Controller jumps between options without any inputs | Added controller to the input map and updated the input settings to only support Nintendo Switch Pro controller |
| Menu navigation | Escape key pressed on keyboard | Opens the pause menu in the pause settings | Opens the pause menu | None |
| Menu navigation | Home button pressed on controller | Opens pause menu in game | Nonresponsive | Add controller to input map |
| Boss health | currentHealth = 0 | Game win screen is activated, and other inputs are disabled | Game win screen is activated, and other inputs are disabled | None |
| Boss health | currentHealth = 1 | The main game loop is ran as normal | The main game loop is ran as normal | None |
| Player health | currentHealth = 0 | Game resets to original state | Game resets to its original state | None |
| Load AI training properly | AI start up on game play button pressed | The training process is started properly | The AI does start properly by booting up a terminal. However, mlagents alters the timescale and how physics and collision functions inside of unity. Speeding up everything | Disabled the training, so it will not be capable of learning from the player in real time. However, the pretrained AI still exists and will be used. |

## Testing table v1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function being tested | Test data | Expected result | Actual result | Fixes |
| Load AI training | AI starting when pressing play | The pre trained AI starts properly | The pretrained AI starts properly | None |
| Load settings () | Resolution index = 9999 | The load settings system clamps the value at 16 (maximum setting) | The load settings system properly detects and loads value as 16 | None |
| Aiming (input) | Mouse has moved | The aim indicator tracks to where the new position is | The aim indicator tracks to the new mouse position | None |
| Aiming (input) | Mouse is currently still | The aim indicator is pointed towards where the mouse position is last | The aim indicator is pointed towards where the mouse position was last | None |
| Aiming | Controller right joystick is being moved around outside of deadzone (>0.2 or <-0.2) | The aim indicator tracks to the corresponding location | The aim indicator updates properly with joystick | None |
| Aiming | Controller right joystick not used (-0.2 < joystick value < 0.2) | The aim indicator is stuck at the last updated position from the controller | The aim indicator is not moving from last controller update | None |
| Player movement (input) | Controller left joystick moved around | The player character is moved in the direction the joystick is moved in | The player does move in the direction. However, the speed of the player is altered when reopening the unity project | None |
| Installing the game files | Copying the Unity project over to another device through downloading the GitHub project | The game is loaded into the unity editor properly and runs without errors | Unfortunately, the unity version specified in the technical environment was not installed and hence the game failed to open properly in unity. However, I am confident that the project will open if unity | None |
| Resolutions indices of a list | Index of screen resolutions array = 99 | An error should print in console, but overall game should not crash unity | An error was printed to the console but did not end up breaking the game, still able to be played. |  |

# Testing summary

In the first round of testing, several problems were discovered with the software relating to the controller and how it functioned properly. The problems affected the controller’s ability to navigate menus and hindered how the user would have interacted with the game through aiming and movement as well. Version 1.0 of testing also revealed that real time training on the AI was not feasible as it altered Unities physics engine and time scale that made it difficult to change back. I opted to remove that feature and rely on the pre-trained AI instead for the boss. Another bug with unity was also revealed during testing where the speed of the character would change when the unity project is opened at different times. I tried comparing the project settings and seeing which setting required altering but could not find a fix for this specific issue. However, for the issues that were able to have fixes applied for version 1.1. Other unintended bugs included settings not working due to project being ran in the Unity editor. If given a longer time constraint, these issues could be quickly resolved by replicating how the settings would alter the game by altering the layout and components of the Editor UI in the settings code.

(User Acceptance Testing (UAT) has been removed for testers privacy)

# To what extent have requirements been met?

FR01 – Inputs that are not mapped to any functionality are ignored

From the testing table, unmapped inputs are ignored as can be seen when pressing “q” on the keyboard, demonstrating that any input that has not been implemented will not interfere with the application. Hence this requirement has been met.

FR02 – Out of range on arrays should fail but not crash the program

The manual override of an array index proved useful information for what might happen if a fault like that did occur. The outcome was far better than expected with the program not only being responsive but also managing to operate as normal, as indicated in the testing table.

FR03 – AI system has data properly input into it and the most “correct” aiming result is output

Throughout training and testing, the AI has improved its aiming and code has been created to check the AI inputs. The AI has been tested on the testing table to correctly aim at the player whenever it is able to shoot projectiles.

FR04 – Switching between keyboard and controller should be handled appropriately, switching over to controller and back to keyboard proved successful during the testing table process. During the User Acceptance Testing, one of the users did report the controller taking some time to take inputs from the controller, however it did end up accepting the input.

FR05 - Program should be capable of running without updates needing to be fixed to operate as intended.

The program does not need intervention from the developer during the user acceptance testing to run the game properly. Meaning that this requirement has been met.

FR06 – Varying technical environments are able to have the solution installed

Unfortunately, this was not able to be tested due to the technical environment on the other device not correctly installed. However, I am highly confident it will run properly if the correct unity version was installed.

NFR01 – Intuitive layout of player functionality

The user acceptance testing proved that all users were able to use the control inputs and found them very familiar. Displaying that this requirement has been filled.

NFR02 – User interface is capable of being navigated effectively

The users that testing occurred on were able to navigate the menus to where they wanted to. All of the users filled out the menu layout criterion as being very high, illustrating the effectiveness of the menu layout.

NFR03 – Desired inputs from the player should produce consistent results

The users found the inputs mostly the same with a bit of difficulty to produce consistent results but eventually got used to them and learnt how to reproduce inputs.

NFR04 – The art style of the game should follow the theme of fighting in a volcano

Every user that participated in the user acceptance tests found that the art style mimicked a volcano very well. The users ended up scoring the art style very high or high on the criterion and were able to distinguish the important elements from theming or background art.