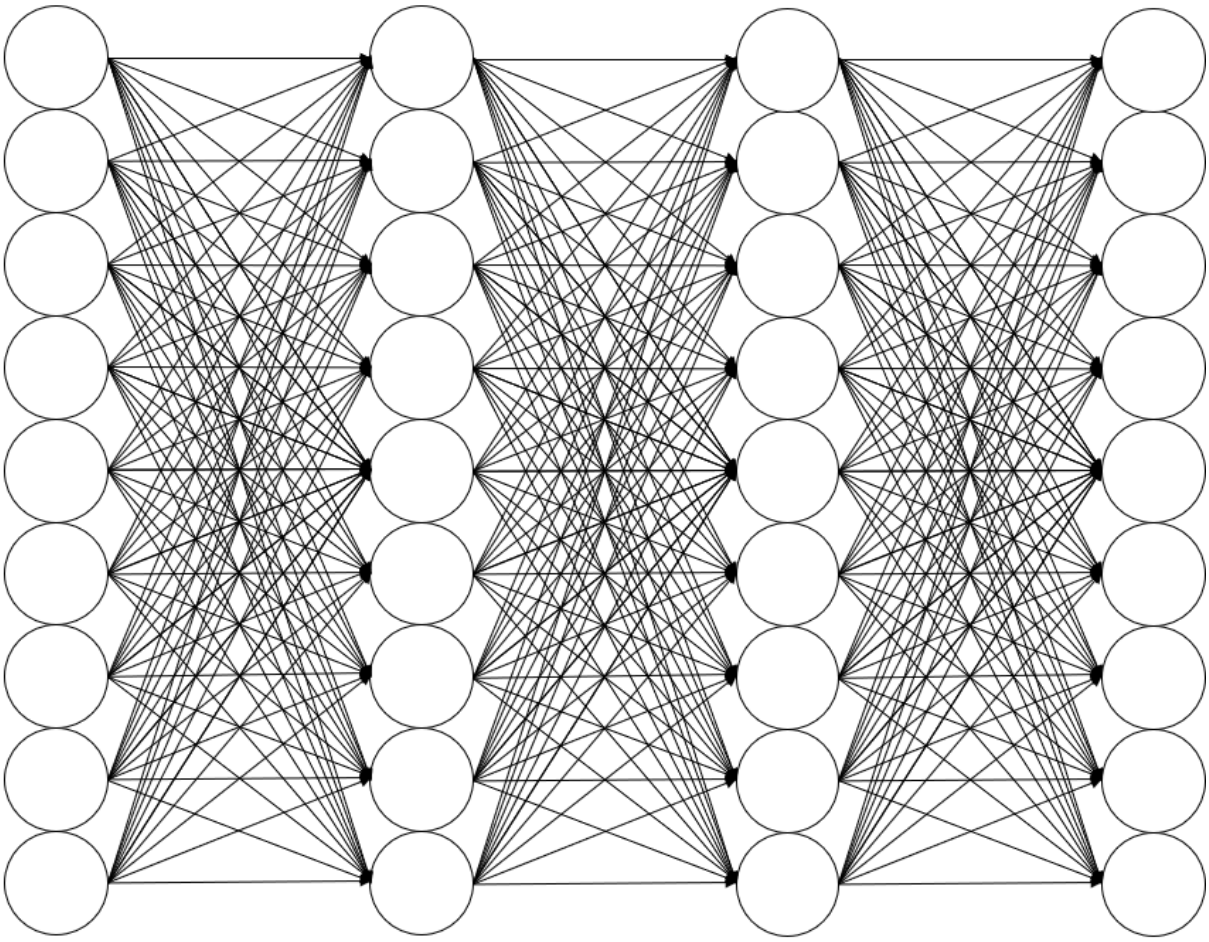


# NEA REPORT

## Missile Command With A Neural Network

William Chittick

# Contents

# Analysis

## Outline

The game Missile Command is an old video game made in 1980 where a player must stop the incoming missiles from reaching the ground by shooting with their own bullets.

My client remembers playing the game missile command when he was a child and decided that he would like to play it again, although unfortunately with it being such an old game he found it very difficult to find an emulation or a decent replica of the game. He also thought it could be interesting to have some system to be able to play against an artificial intelligence that has the same move options as a human player and which tries to destroy as many missiles as it can, with whichever of the human and the AI shoots the most missiles being declared as the winner. He saw a video of the game pong being played by a **neural network** with **deep Q**. As a veteran player of the game, he wonders if he would be able to beat such an AI and so he has asked me to help him create some software with a working replica of missile command and the option to play against such an **artificial intelligence**.

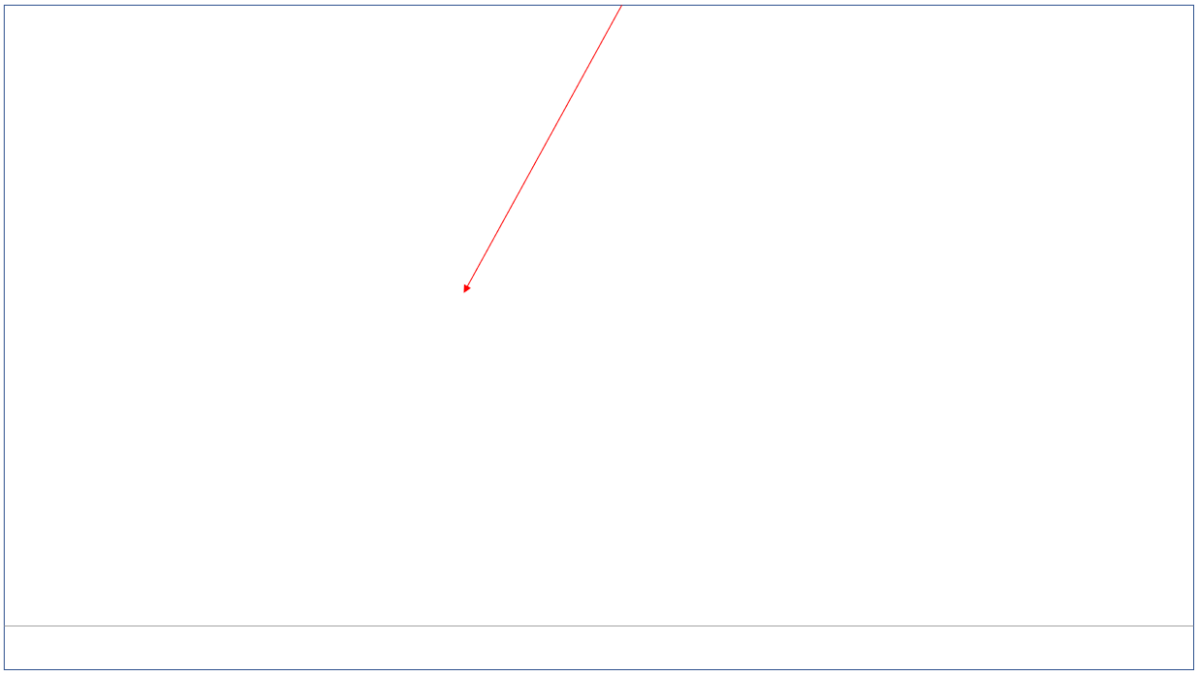
It would be possible to program an AI that simply shoots bullets at missiles by performing simple mathematical computations, using the speed of the missile and the speed of their bullets to predict the path of the missile, calculating the time it will take the missile to reach certain points on the screen and shooting bullets at those points accordingly. However, aside from the fact that this computer player would be quite boring, I do not think it will be able to play very optimally, as it will not look very far into

the future and it does not take into account that another player is shooting at the same missiles. So I think that using a **neural network with deep Q-learning** would be better.

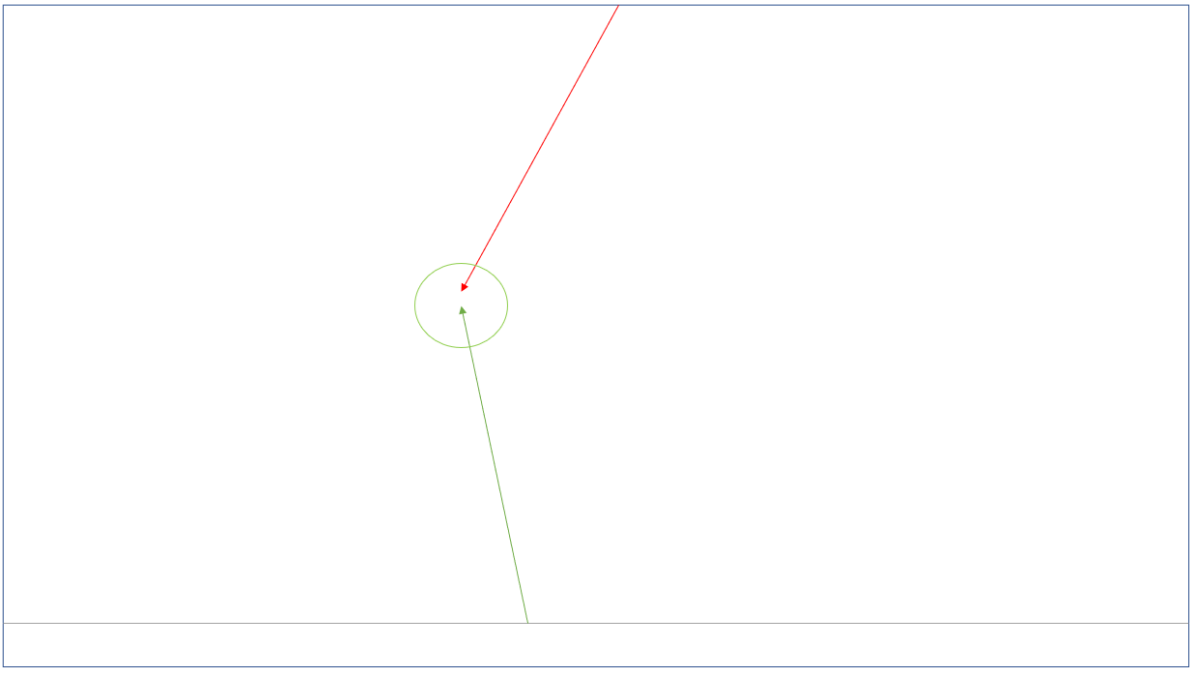
I am going to interview my client to gain his views on what he wants with the piece of software I am going to develop and to help me write down some objectives that need to be met throughout the development to make my program sufficient for his use.

## Description of Missile Command

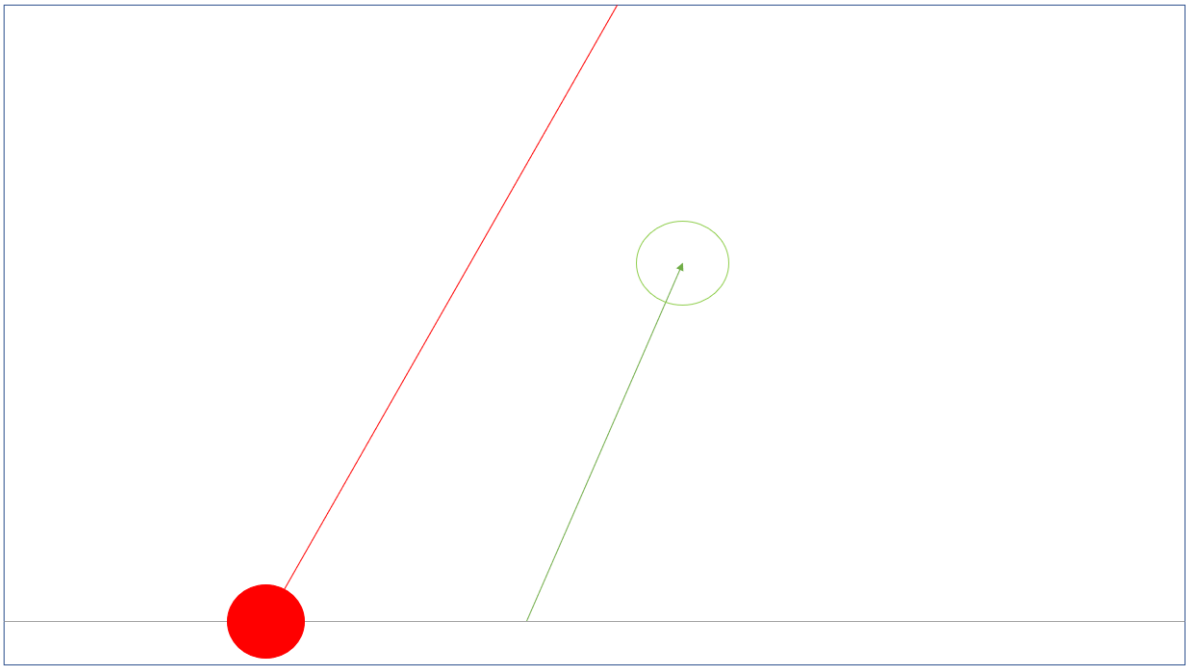
The game missile command works as follows. Once the game starts missiles will start coming down from the top of the screen to the bottom of the screen.



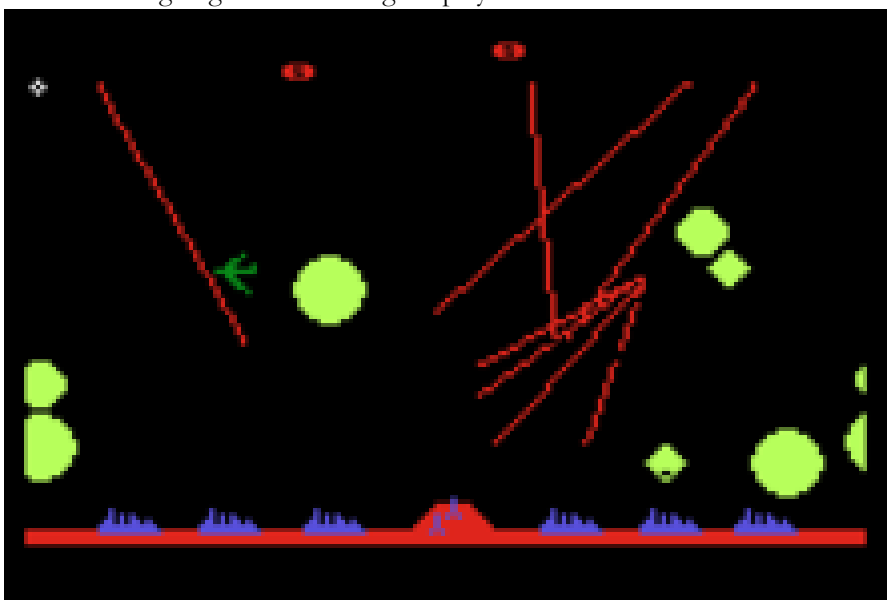
The player can destroy the missile by predicting the path of the missile and shooting their own bullets where they think the missile will be. The bullet will then explode, and the missile will be destroyed if it is within a certain range of where the bullet exploded, and the player's score will be incremented by 1.



If the player were to have missed that missile and then it hit the ground the player will lose one of their lives, and if they reach 0 lives then they will lose the game and their final score will be the score they had before their lives hit 0.



Below is a picture of the gameplay of the original version of missile command. I will use this when it comes to designing the GUI and gameplay.



## Users

The primary user and my client for this software will be Craig Chittick. However, he would also like to share this project with other veteran Missile Command friends who would also like to play Missile Command against an **artificial intelligence**. Thus, to make it easily accessible, I will write my project in python as it is an interpreted language so there will be no issues of having to make a compiled version for every operating system that a user may have.

## Project requirements

So that I can make sure that I am meeting the requirements of my client, I have decided to conduct an interview with them so that I can fully grasp what they want and to help me create a set of objectives that I must meet during this process.

- ***Question:*** *"What is it that you would say makes Missile Command fun for you?"*
  **Client's response:** "I would say that the strategies that one must build in order to play the game is what makes it fun. For example, how a player should always prioritize the missiles that are the closest to the ground to avoid getting overwhelmed later. Aside from that, I find the competitive nature of solo games to be very enjoyable, in how one can compete against friends and compare highest scores."
- ***Question:*** *"What is the problem with the current system?"*
  **Client's response:** "The main problem that I have is that the only versions of Missile Command that I can find online are not very good and the emulations of the original game that I have found online do not run very well on modern hardware. They also do not play very well with mouse and keyboard as they rely on using the buttons and joysticks of an arcade machine."
- ***Question:*** *"Are there any essential features you would like to see in the final product?"*
  **Client's response:** "Firstly, I think it is essential that the final product can store the highest scores achieved on that device. I have intentions of sharing the software with some of my friends and I want to be able to compare my highest scores with them. I also think that the game should get harder as time passes. I think that a lot of solo games have the issue that if a player is sufficiently good at the game then they can in theory play the game forever and then getting high scores becomes more of a problem of being able to endure playing the game for a long time and less about being more skilled. Also, to make it easier for me to compete against my friends I think there is a need for some kind of seeding system that can use a single number to determine the whole game. However, the game should still feel random to the player and the next missile's path should not be predictable without having played the seed before."
  **Key Objective:** The game needs to be able to store the highest scores achieved.
  **Key Objective:** The game should get more difficult as time passes.
  **Key Objective:** The game should support a seeding system that lets the game be deterministic yet seemingly random.
- ***Question:*** *"How do you think that the new version of the game could potentially be an improvement over the original game?"*
  **Client's response:** "Well in the modern day, we obviously have new hardware and I think honestly that a mouse would be a better input device for Missile Command as it would allow for more precision and would not have a cap on how fast it can move like the joysticks of the original game. This would allow for a higher potential skill ceiling of the game, which is ideal for competing against my friends. Also, the Artificial Intelligence that I suggested to be played against I think would be very cool to see."
  **Key Objective:** the game should use mouse input for letting the player choose where to shoot their missiles.
- ***Question:*** *"Can you expand on what you want to see with the Artificial Intelligence?"*
  **Client's response:** "Firstly, the game should be able to run the Artificial Intelligence without there being any sort of lag. I know that neural networks can be quite computationally expensive so I think the system should have the best possible neural network without sacrificing the speed of performance of the game. I also think that it is important that the neural network is not unfairly good, it would obviously be possible to simply allow the neural network to shoot as many times as it wants per second, which would be quite unfair to the human player. So, I think the neural network should shoot in fixed intervals at a rate that the average human could achieve quite easily."
  **Key Objective:** the game should support the ability to play against a neural network

- ***Question:*** *"Do you have any idea what you think you would like to GUI to look like?"*
  **Client's response:** "I think to honour the original game, the system should have a fairly simplistic look, using only simple monochromatic geometric shapes. I also think that it is quite important for there to be some sort of menu system, allowing the user to navigate between the two modes and see their high scores. This I think should also have a very simplistic design to make it more intuitive for users to use. Also, during the execution of the game, the game should show the current score of the player along with the number of lives they have remaining, once the player has run out of lives they should be taken to a screen that displays their score, the seed used and a button that allows them to return to the main menu."
  **Key Objective:** the game should have a menu that allows the user to navigate between modes and view high scores.
  **Key Objective:** the game should have a simplistic design.

## Limitations

Time Limit

Although my client did not specify a hard time limit for when this should be done, other than it being done as soon as possible, I would rather finish before March 2023 as by then I will need to begin studying for my A-level exams. I have decided to set this as a final deadline for myself.

Hardware

I am aware that the neural network that I am going to make could potentially be better if I could use better hardware, but people who are going to use this software will not have the best hardware available. As such, I am going to make this project so that it can run smoothly on my laptop.

## Proposed Solution

I have decided that I will create a bespoke piece of software using Python with the pygame module. The alternatives to pygame would most likely be tkinter or pyglet. I have no experience with any of the three mentioned modules, so no matter which one I were to choose I would have to learn it from scratch. This is the main reason why I have decided not to use pyglet as it is a much lesser-known python module and so it has much less documentation to help me learn it. This means I will have less time to work on other parts of the system. The reason I have decided to not use tkinter is that it is quite slow and the neural network will already be very computationally complex so it would be ideal for the rest of the project to run as quickly as possible to allow for the best artificially intelligent player possible.

I have decided to use Visual Studio Code as my integrated development environment. I find it to be very friendly for newer programmers and it has a lot of useful functions which I would like to have, such as automatic indentation, informative colour coding and automatic bracket completing. Visual Studio Code is also a very popular IDE so it should be very easy for me to find help online if I run into an issue mid-development. It also has built-in debugging features such as stopping the program at whichever line is causing the program to terminate and underlining code that is going to run time error. It is also the IDE which I am the most familiar with so I will have to spend less time learning my way around it. This will all help significantly with development time which will allow me to add many more features within the time limit.

I have chosen to use a **feedforward neural network with deep Q learning** to play as the computer player. The other obvious alternative would be to simply use a mathematical based model which predicts the path of missiles and shoots in the correct place. The problem with this, other than the fact that it would simply be very boring to play against, is that it cannot take into account what the player is already shooting at and has no real way of destroying missiles with much plan for what will happen in the future. If I were to implement such a mathematical model to be able to look forward in this way it would firstly be extremely difficult to figure out every possible thing that I should take into account, but also it would

most likely just end being far worse than using a neural network to achieve the same thing as they naturally look into the future when making decisions.

*Advantages and disadvantages of python*

| Advantages | Disadvantages |
|---|---|
| Is interpreted thus more portable | Very slow |
| Lots of documentation online | |
| Large standard library and publicly available library | |

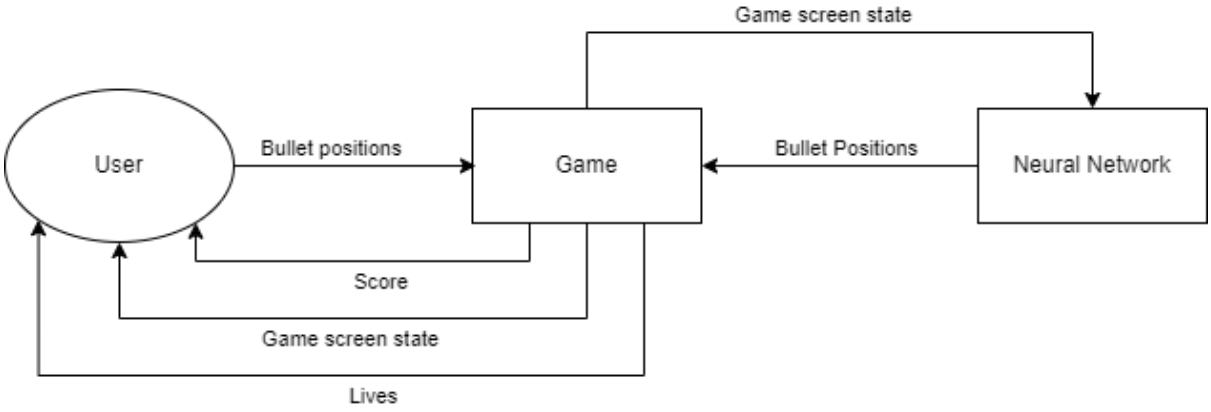
## Choice of the neural network model

I have chosen to use **deep Q learning with a feedforward neural network** as the computer player. This is because deep Q learning is very good for playing games such as this one where it must take into account how its current actions will affect its future ones. This is because of how the Q function formula has a recurrent relation with past states, so if I can get a feed-forward neural network to properly approximate such a function it would be able to play missile command with much regard for the future.

## Objectives

1. When the game is started the user should be presented with the options of viewing their high scores, playing the game solo, or playing the game against a neural network.
2. The game must have a seeding system that uses a single integer to determine the path of missiles and the time they will spawn so that the game appears random but using the same seed will always give the same time and location of missile spawns.
3. The game must spawn missiles for the user to shoot and as time passes during a game the number of missiles that spawn per second should increase.
4. The user must be able to spawn bullets by clicking on the screen and have a bullet travel to the location clicked. When a bullet explodes near a missile the missile should be destroyed and the player's score should be incremented by one.
5. The player must have a given number of lives. Every time a missile reaches the ground they should lose one life, if they run out of lives the player should lose the game and show a screen containing the score they achieved, the seed used and a button to return to the starting screen.
6. The program must be able to store the highest scores achieved by the player when playing solo. The player should also be able to view them after navigating to them from the start menu.
7. The program must also support the option to play against a **neural network**.
8. The program must have the **neural network** shoot bullets at a point on the screen in fixed time intervals.
9. The **neural network** weights must be stored between uses of the program so that they can be used the next time the program is run.
10. The game must show the current score of the player and of the **neural network** during the game's execution.

## Data Flow Diagram




## Critical path design

I have about 15 weeks to get this project finished and I then will have to hand it over to my client and will not be able to change it from then onwards. So, to keep myself on track with the project and make sure that I finish it on time I have decided it would be a good idea to create a high level timeline for what parts of the project I should work on and when I should have them finished.

| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Missile Command | Create a functional version of missile command that is playable by a human | | | | | | | | | | | Integrate the feedforward neural network into the game and train it. | | | Final bug fixing and testing |
| Menu | | | | | Create the menu that will be shown at the start of the game that will allow the user to navigate between modes | | | | | | | | | | |
| Neural Network | | | | | | | | Implement a general feed-forward neural network | | | | Integrate the feedforward neural network into the game and train it. | | | |

# Design

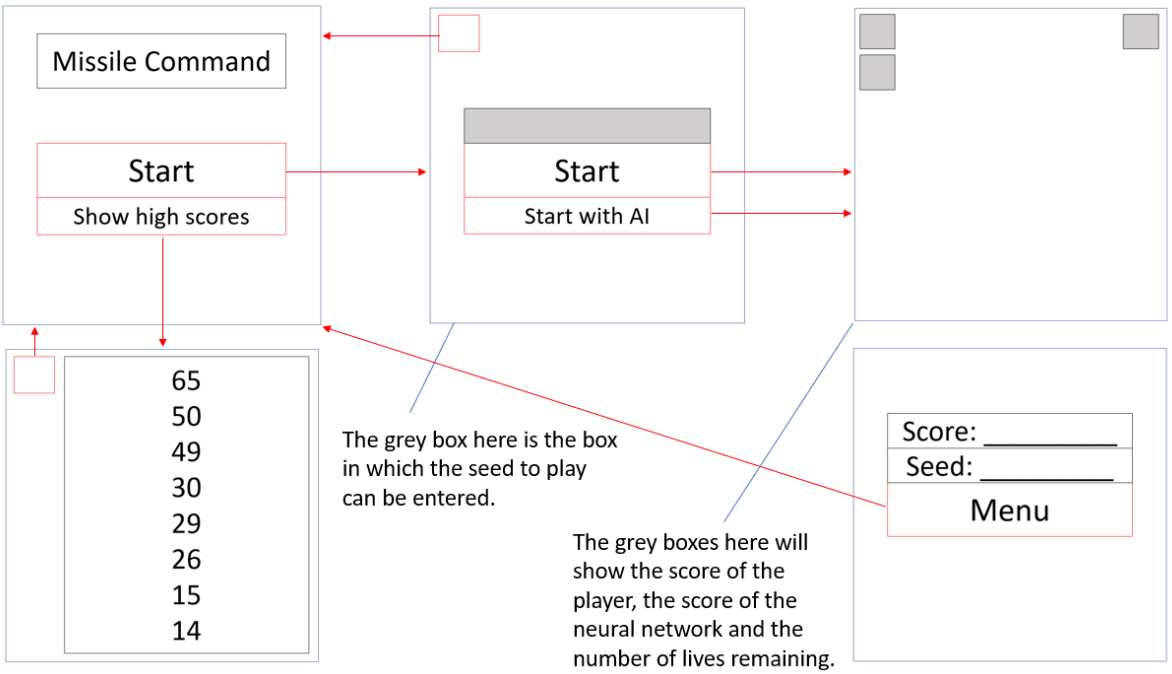
## Project Overview

I intend to make a playable version of Missile Command which displays information about the game on the user screen with a graphical user interface. The game should also respond to the user clicking on the screen to create defensive bullets. There should also exist another version of the game where the user can compete against an artificial intelligence which will be programmed to use a feed-forward neural network to calculate the best moves.

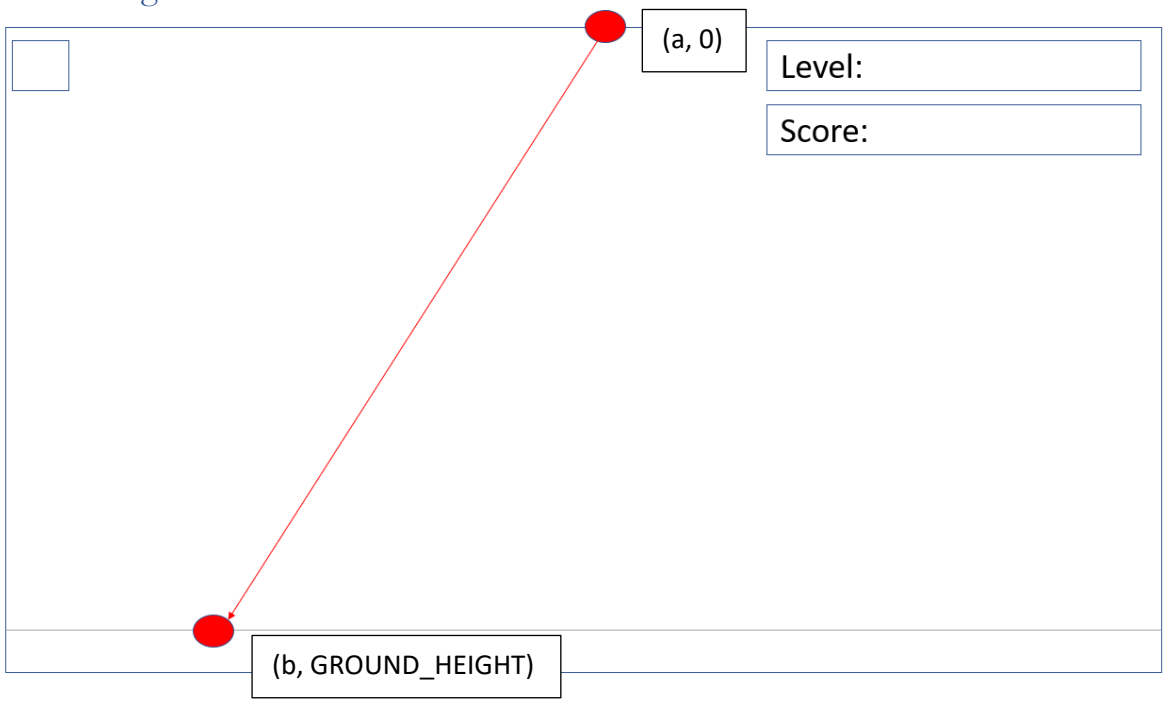
There also must be a menu which is first displayed when the program is run which will display a set of buttons and textboxes which can respond to the user clicking on the screen to create an intuitive way for the user to navigate between modes.

## Graphical user interface design

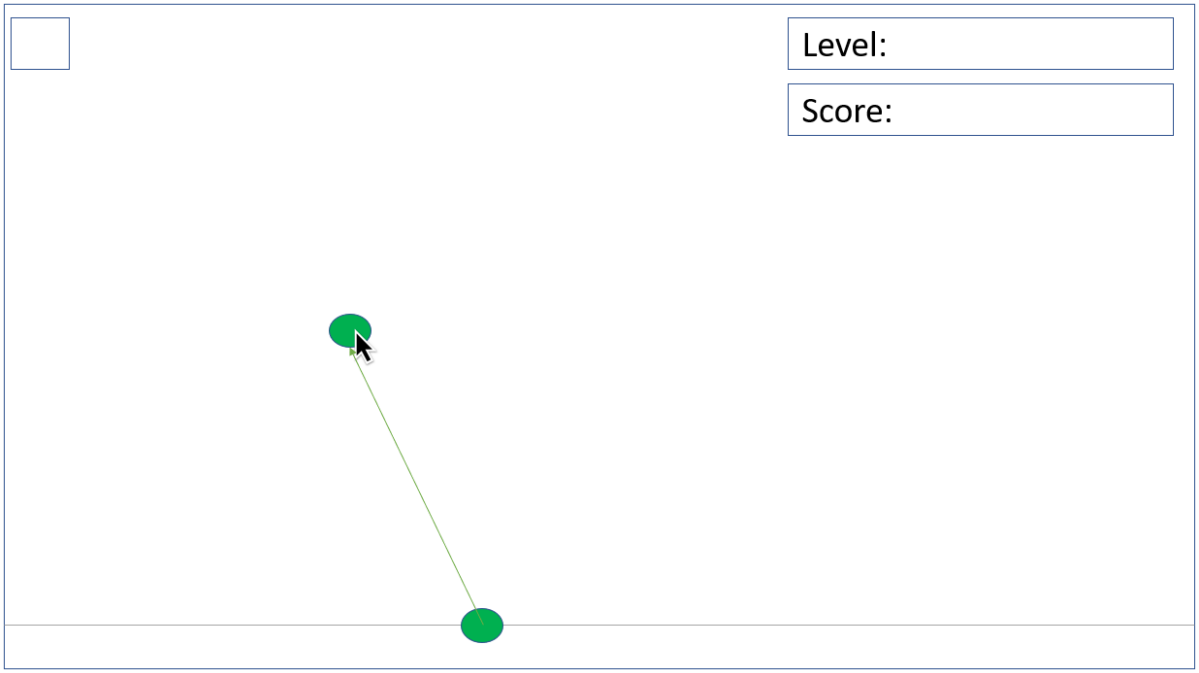
To help myself get a better understanding what of the graphical user interface will look like, I have decided to create a rough sketch of the design of the whole thing which is illustrated in the picture below.



## Game Design



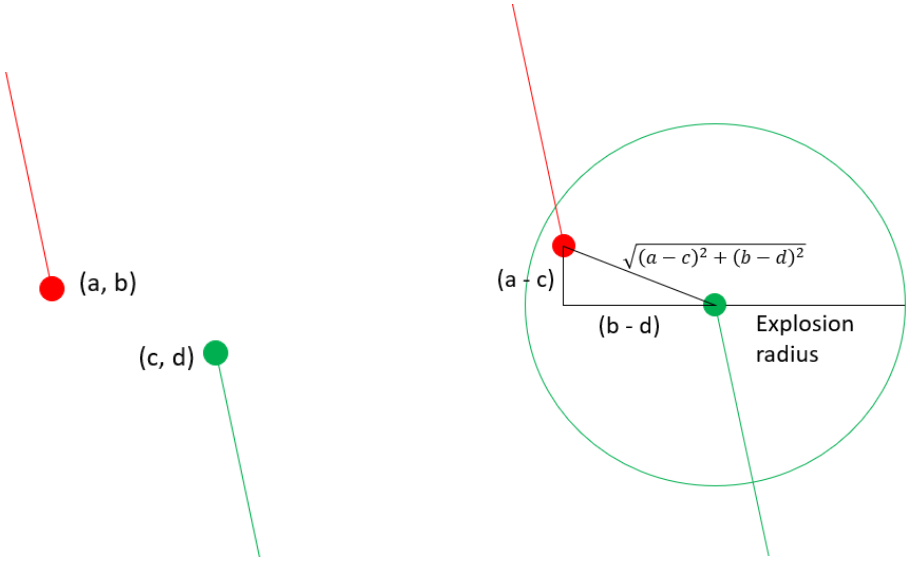

The path that the missile will take will be calculated as follows. Two random numbers a, b will be generated in the range of (0, screen width). The initial position of the missile will be the coordinate (a, 0), and the destination of the missile will be the coordinate (b, ground height). The path of the missile will be the line that connects the first point to the second point. The missiles will move at a constant speed and missiles will move at the same speed to allow the player to predict their paths more consistently.




When the player wants to fire a bullet, they can move their cursor to the position on the screen where they wish to fire. The bullet path will then be determined as the position in the centre of the ground to the position that the user has clicked. The bullets will all travel at the same speed.

To check whether the bullet has hit a missile. In each frame, each bullet that has reached its destination in that frame will be added to a list. Then for each missile on the screen, it will be checked whether that

missile is in the explosion radius of the terminated bullets (which can be checked as shown in the diagram below) and if so the missile will be destroyed.



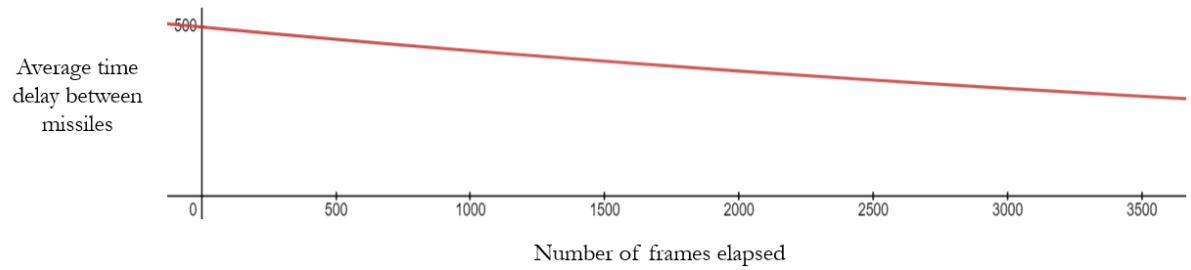

Pseudocode for such an algorithm (here terminated_bullets is this list of bullets that have reached their destination, missiles is the list of missiles currently in play and EXPLOSION_RADIUS is the radius of the explosion of the bullets):

```
for bullet in terminated_bullets
      for missile in missiles
            dx = missile.position_x - bullet.position_x
            dy = missile.position_y - bullet.position_y
            if (dx * dx + dy * dy <= EXPLOSION_RADIUS *
EXPLOSION_RADIUS)
                  del missile
            endif
      endfor
endfor
```

## Increasing difficulty

My client has specifically requested that he would like the game to get more difficult as time passes, I think that the easiest way to achieve this would be to make the time between missile decrease as time passes. I have decided to use the following formula to calculate the number of frames until the next missile should spawn. $t = 0.99985^{(frames\ elapsed)} * 500$. I think that this formula is ideal as the game will get steadily more difficult but will always be increasing in difficulty and thus the player will find a point where they are overwhelmed and lose the game. This eliminates the problem of a good enough player being able to play indefinitely that was outlined by my client. I will also add a random number between (-25, 25) to this so that the missile times are less predictable.

Here is a graph of how the time between missiles will vary as the game goes on. On the y-axis is the number of frames between missiles and on the x-axis is the number of frames elapsed.



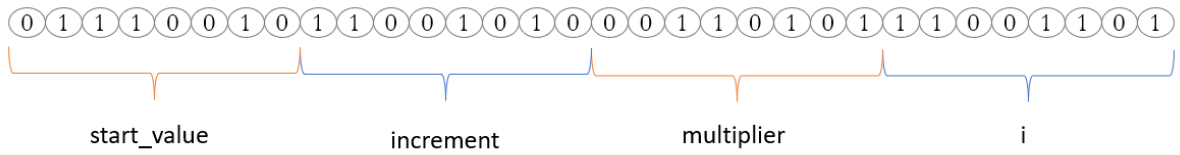

## Random number generator

My client made it clear that the missile generation system must have some sort of seeding system to make the missiles in the game spawn deterministically yet seemingly randomly to a human. Thus, it is necessary to make an RNG (random number generator) class that takes a seed on initialisation and then when a function is called on the object a number is returned which is seemingly random but if the same seed is used the same set of numbers will always be generated.

I will be using a Linear Congruential Generator which is defined by the following recurrence relation:

$$X_{n+1} = (aX_n + c) \mod m$$

Where a, c and $X_0$ are arbitrary integers and m is an arbitrary prime, m must be prime to provide the highest number of possible outcomes. The seed will be a 32-bit integer which will be split up as follows (the grey circles with binary numbers in them represent the bits of the 32-bit number):




m = p(i)
a = multiplier
c = increment
$X_0$ = start_value

p(i) represents the ith prime number bigger than 97000 (as starting at 2 would provide too few possible outcomes).

This allows for 2^32 (4294967296) seed inputs which should be sufficient. If the user enters a number bigger than the allowed seed range (0 – 2^32) that number modulo 2^32 will be used as the seed.

Here is pseudocode that could be used to set the seed in an RNG class, where seed is the variable of the entered seed, in the code I assume that the list PRIMES is an array of size at least 256 which contains large prime numbers:

```
class RNG
      private mod
      private multiplier
      private increment
      private current

      public procedure new (seed)
            mx = PRIMES[seed mod 256 + 1]
```

```
            seed = seed floor 256
            multiplier = (seed mod 256) + 12345
            seed = seed floor 256
            increment = (seed mod 256) + 23457
            seed = seed floor 256
            current = (seed mod 256) * 390
      endprocedure
```

For some of the values of the seed, I decided to add arbitrary large numbers so that the numbers feel more random and to allow for more possible outcomes.

The following pseudocode can be used to generate the next number in sequence in the same RNG class:

```
public procedure next()
      current = (current * multiplier + increment) mod mx
      return current
endprocedure
```

It is ideal to be able to generate any number in any range l to r where all outcomes are equally likely. Given a uniform random number generator $RNG()$ that can generate any non-negative real number less than a number $MAX$ where all outcomes are equally likely, it can be modified as follows to create the function $randrange(l, r)$.

$$randrange(l, r) = \frac{(l - r)RNG()}{MAX} + l$$

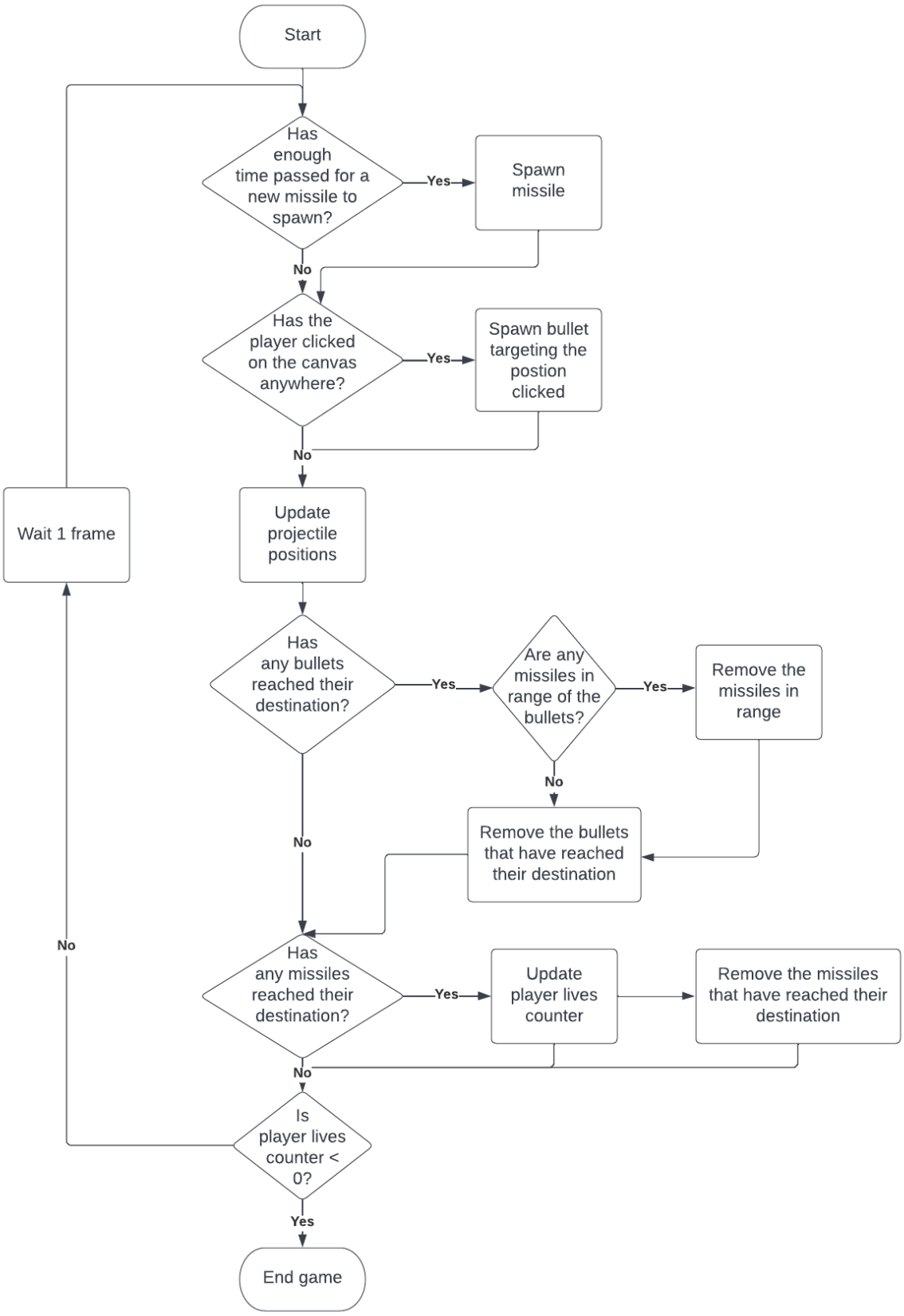
Thus the next() function for the RNG class can be modified as follows to allow there to be any range of numbers.

```
public procedure next(l, r)
      current = (current * multiplier + increment) mod mx

      dif = (r - l) / mx
      val = dif * current + l

      return val
endprocedure
```

## Game logic flow chart



## Neural Network Design

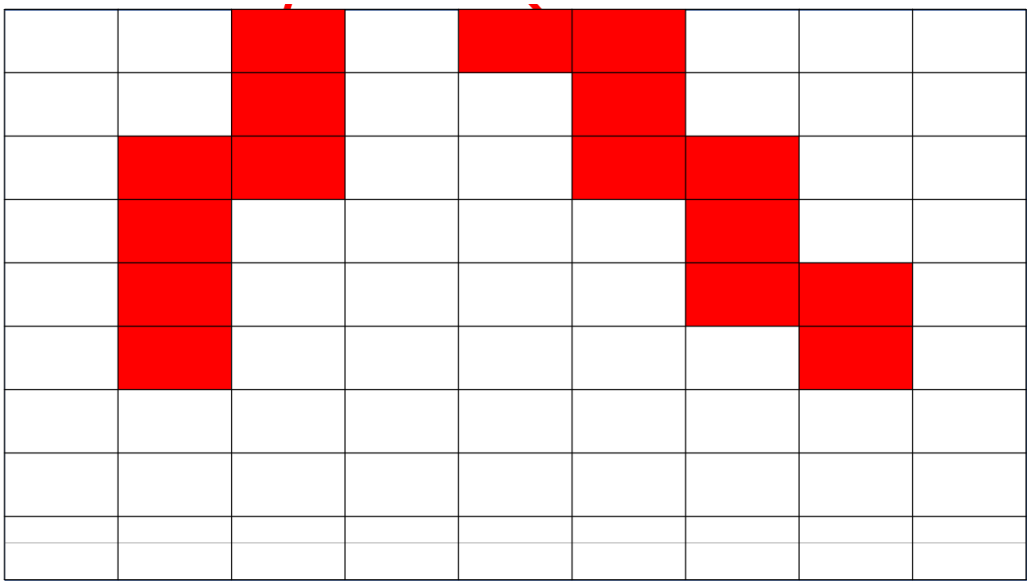
There needs to be some way to process the screen in an appropriate way such that the information on the screen can be fed as an input into a neural network in order to return moves. It would be possible to use a list containing the RGB values of all the pixels. However, as the screen has dimensions of 640x640, it would require 409,600 input nodes which is far too large and it would not be feasible to train such a network and have the system run smoothly with it running. So instead, I will split the screen up into regions and if there is a red pixel (the colour of missile trails) in that region the corresponding input node in the neural network will have value 1, if the region is green or blue (the colour of bullets) it will have a value of 0, otherwise it will have value -1. I will split the screen into 81 regions (9x9) which should be sufficiently small as inputs.

Below describes the process of getting the input of the neural network:

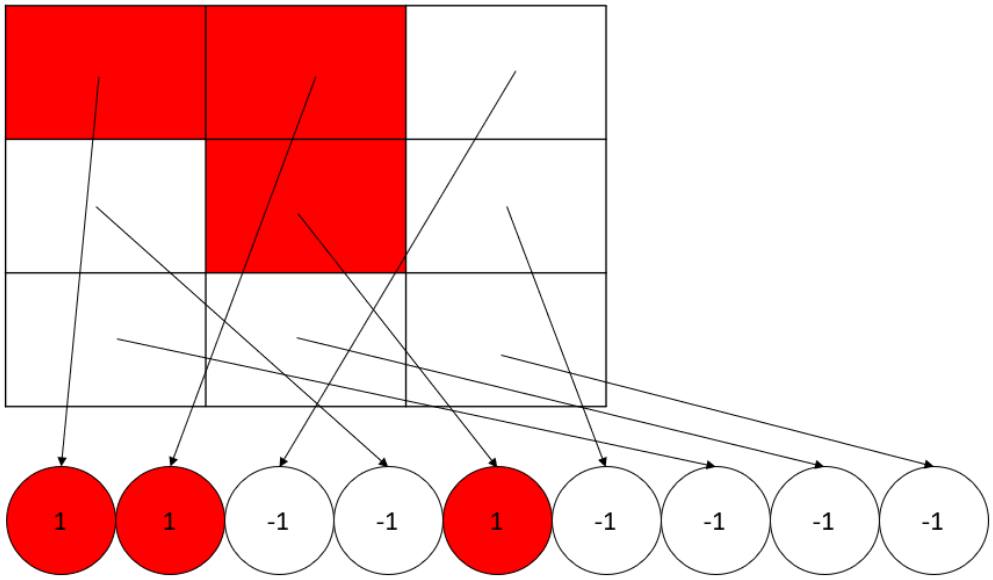
Suppose the current state of the screen looks like this.

The screens can be split up into regions as such.

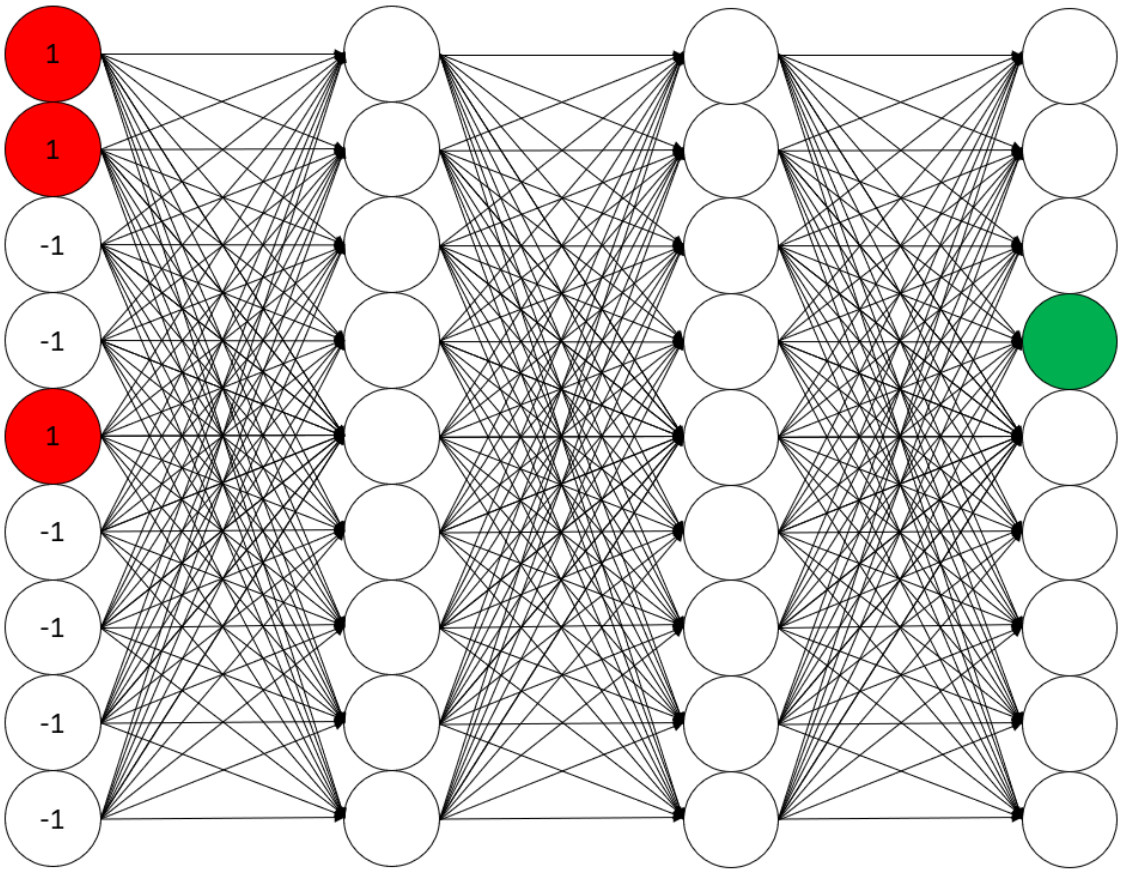
Then the regions with red pixels can be found.



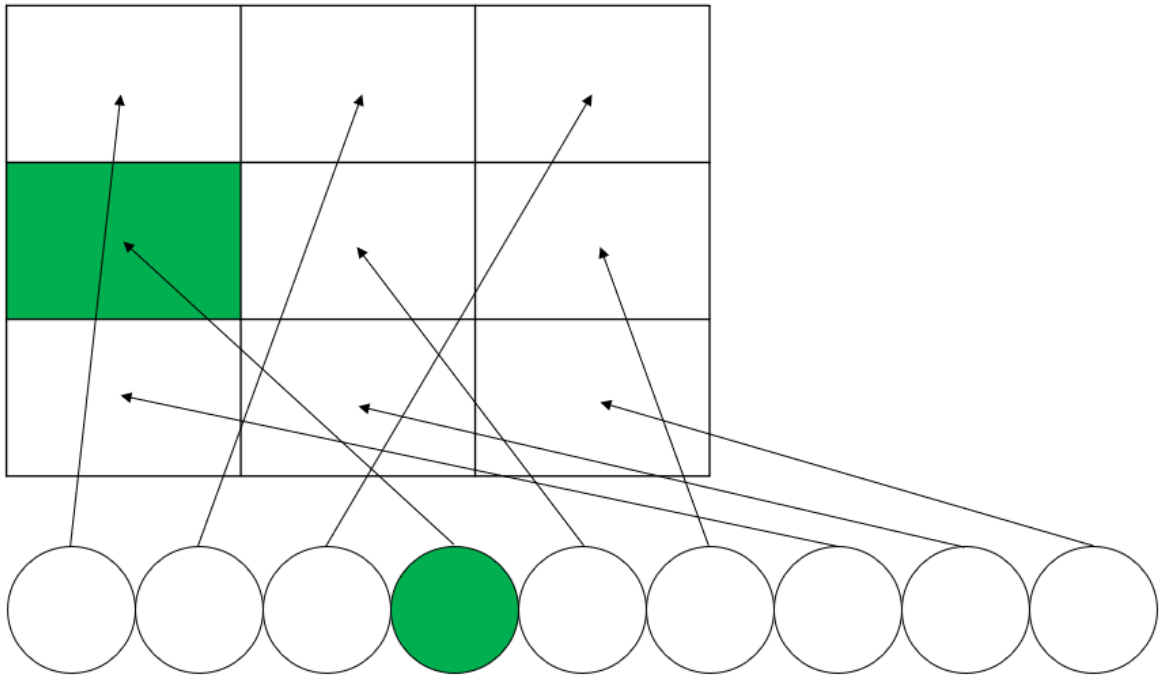
These regions will then be turned into a 1 dimensional array. An example of what the neural network input layer will look like after this process is shown below for a smaller number of regions than will be used.




As the missiles and bullets move at a constant speed it should be possible to predict the path and where to shoot a bullet such that it will reach the missile at the correct time.




The values obtained from the screen will be entered as inputs into the neural network, then the output node with the highest value will be used.



The screen will be split the same way for picking target positions as for the inputs so there will be the same number of output nodes as there were input nodes for the neural network and each output node will be mapped 1 to 1 with a corresponding position on the grid.

## Forward Propagating the neural network

After the screen has been processed to give the input layer of the neural network, the neural network must now forward propagate to generate the output values. The equation for the values of the nodes in the next layer of the neural network can be given by:

$$n_{i,j} = sig\left(\sum_{r=1} n_{i-1,r} \cdot w_{i-1,r,j}\right)$$

where i is the current layer, j is the index of the node we are calculating in the next layer, and r is the current index of the node in the previous layer, $w_{i,r,j}$ is the weight that connects the rth node of the ith layer to the jth node of the (i + 1)th layer, $n_{i,j}$ is the value of the jth node in the ith layer and sig(x) is the sigmoid function. This equation must be used to calculate all the values of the nodes in each layer.

The sigmoid function is defined by.

$$sig(x) = \frac{1}{1 + e^{-x}}$$

I have found that the equation described above can be simplified to the following matrix multiplication.

$$\begin{pmatrix} w_{i-1,1,1} & w_{i-1,1,2} & \cdots & w_{i-1,1,n} \\ w_{i-1,2,1} & w_{i-1,2,2} & \cdots & w_{i-1,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i-1,m,1} & w_{i-1,m,2} & \cdots & w_{i-1,m,n} \end{pmatrix} \begin{pmatrix} n_{i-1,1} \\ n_{i-1,2} \\ \vdots \\ n_{i-1,n} \end{pmatrix} = \begin{pmatrix} n_{i,1} \\ n_{i,2} \\ \vdots \\ n_{i,m} \end{pmatrix}$$

And then applying the sig function to all the nodes in the next layer.

As the project will be python, I can use the numpy library to represent the matrix and to carry out the multiplication which will be much faster than the alternative of just looping each pair nodes as described at the top of this section. As the numpy library is written in C it is very fast which means that using a matrix multiplication will be significantly faster allowing me to create larger neural networks that are ideal for playing the game better.

## Updating the weights of the neural network

The Q function which I will be trying to approximate will be as follows:

$$Q(s, a) = r(s, a) + \gamma \max_{a} Q(s', a)$$

s is the current state, a is an action that can be used in the current state, r(s, a) is the reward of the current state (which I will discuss later) and γ is a constant between 0 and 1. In my project s will be the current

value of the 81 regions that the screen will be divided into as indicated above, a is any one of the 81 regions that the neural network can chose to shoot at. I will choose the value of γ through trial and error. In order to properly approximate this function, I need to be able to train the neural network in some way. When the neural network forward propagates to get the outputs, the value in each of the output nodes is meant to be the predicted value of the Q function evaluated at the current state for each potential action. So, in order to find the actual value of the Q function evaluated at the chosen state, I will use a copy of the neural network and update the game until the next time the AI should perform an action. I will then evaluate the copy of the neural network at the current state and find the max value of an output node. I will then calculate the reward from the current state, and this is all that is necessary to calculate what the actual value of the Q function should have been. Thus, the error of the approximation of Q function will be

$$approximatedvalue - r(s,a) - \max_{a}(s',a)$$

I will then need to update the values of the weights of the neural network to minimize this error so that it can better approximate the Q function for this game. To do this I will backpropagate the error from each weight backwards to try to find how "wrong" the weights are and adjust.

## Backpropagating the neural network

The recurrence relation for the error of the jth node in the ith layer is given by:

$$error_{i,j} = td(n_{i,j}) \sum_{r=1} error_{i+1,r}.w_{i,j,r}$$

Here the td(x) function represents the following function:

$$td(x) = x(1-x)$$

Assuming x is a value evaluated from the sigmoid function then td(x) will be the derivative of the function at that point. I will show why below.

$$\frac{d}{dx}sig(x) = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$sig(x)(1-sig(x)) = \frac{1}{1+e^{-x}}(\frac{e^{-x}}{1+e^{-x}}) = \frac{e^{-x}}{(1+e^{-x})^2}$$

Therefore:

$$\frac{d}{dx}sig(x) = sig(x)(1-sig(x))$$

I multiply by this so that the weighting updates follow stochastic gradient descent.

The recurrence relation shown above is equivalent to the following matrix multiplication: a d

$$\begin{pmatrix} w_{i,1,1} & w_{i,2,1} & \cdots & w_{i,n,1} \\ w_{i,1,2} & w_{i,2,2} & \cdots & w_{i,n,2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i,1,m} & w_{i,2,m} & \cdots & w_{i,n,m} \end{pmatrix} \begin{pmatrix} error_{i+1,1} \\ error_{i+1,2} \\ \vdots \\ error_{i+1,n} \end{pmatrix} = \begin{pmatrix} error_{i,1} \\ error_{i,2} \\ \vdots \\ error_{i,m} \end{pmatrix}$$

And then multiplying each error value by the td function evaluated at the value of the corresponding node.

The new value of each weight is given as follows:

$$w_{i,j,k} = w_{i,j,k} - \alpha . error_{i+1,j} . n_{i,k}$$

Here α is a constant between 0 and 1. I will choose this value through trial and error.

Using a matrix multiplication here is again much faster than using loops to calculate all the errors as I can use the numpy library which C which is much faster than I could possibly do it in python.

## Calculating the reward of a position

As part of deep Q learning I need some way to determine the reward of a position. This is simply one number to evaluate how well the neural network is doing.

I have decided on the formula:

$$reward = ((change\ in\ score) - (number\ of\ lives\ lost) - 0.5) * 10.$$

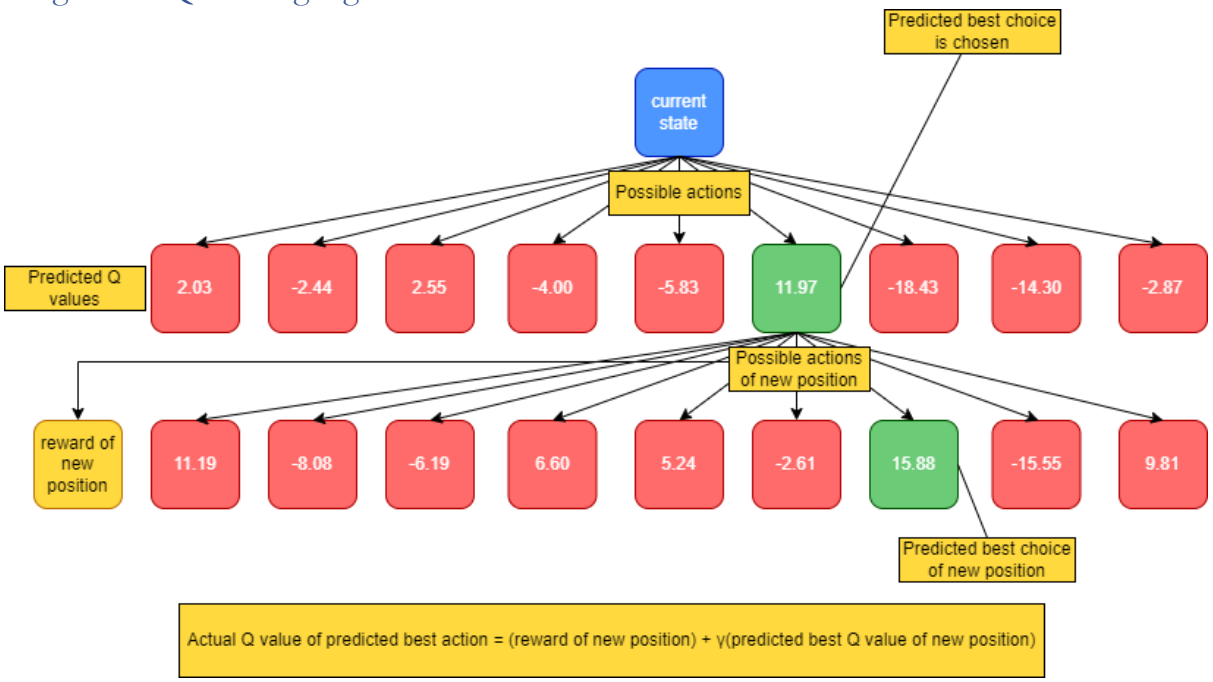
$(change\ in\ score)$ represents the number of missiles that have been hit by bullets shot by the neural network since the last time the neural network was evaluated. $(number\ of\ lives\ lost)$ represents the change in the number of lives since the last evaluation.

I think that this is a good reward function as the $(change\ in\ score)$ will incentivise the AI to destroy as many missiles as possible and the $(number\ of\ lives\ lose)$ will incentivise it to prioritize the missiles that are closest to the ground and subtracting a constant $(0.5)$ will prevent it from doing nothing.

I could also include the change in the players score into the formula, however as of the player bullets and bullets form the neural network will travel at the same speed, there is very little that the neural network can do to prevent the player form getting a score other than destroying the missiles first, which it is incentivised to do anyway by the $(change\ in\ score)$ parameter. So I think that the simpler formula will perform better.

For all the **matrix multiplication** I plan to use numpy arrays from the numpy library. They behave quite similarly to arrays but they also support the option to be treated as a matrix or vector for multiplication. All of these function are written in C which is significantly faster than python so this will allow me to use much larger neural networks.

## Diagram of Q learning logic



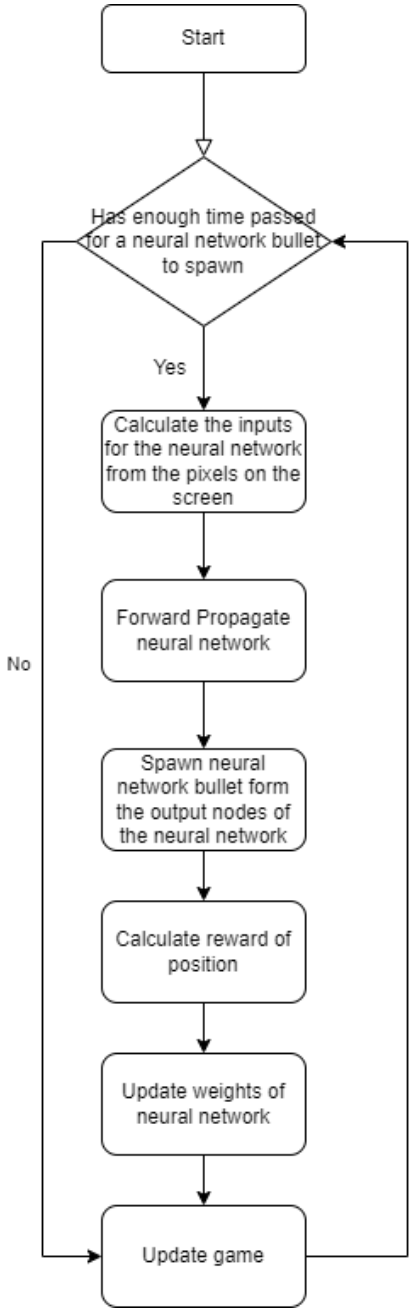

After calculating the actual expected Q value of the action taken in the current position. It is possible to calculate the error between the actual value and expected value (expected – actual) which I can use to update the weights of the neural network accordingly.

## Storing the neural network

It is necessary to be able to store all the weights of the neural network so that an already trained AI can be used, and AI in training can be stored. I think that the simplest way to do this would be to simply store all the weights in a text file. I will simply iterate through every weight of the neural network and write the value of such weight on a new line. When the system needs to read the weights of the neural network it can simply iterate through all the weights in the same way to determine their values.

## Game with neural network flowchart

In the flowchart below the instruction "update game" refers to running the game logic flowchart above.




## Storing high scores

I plan to store the highest scores that have been achieved in a text file. I will simply store the top 100 scores achieved in such a file. I will sort the scores so that it is easiest to find the highest ones to display and so it is easier to remove the lowest score once there are over 100 scores that need to be stored

## Class Definitions

| Class Name | Overview |
|---|---|
| RNG | The random number generator. Takes a seed on initialization which will always give the same outputs, details of calculations is described earlier. |
| Projectile | Something that moves uniformly between two positions and explodes on impact. Missiles and Bullets inherit from this |
| Missile (inherits from Projectile) | Inherits from projectile, are the oncoming missiles from the game |
| Bullet (inherits from Projectile) | Inherits from projectile, are the bullets that the player fires in order to destroy missiles |
| Game | Handles all the logic of the game. It manages spawning missiles and bullets and updating their positions each frame. Removes missiles/bullets that have reached their target or been destroyed from the game. It keeps track of the players lives and score. |
| Game_GUI | Using a Game class it draws everything necessary on the screen, i.e. the missiles, bullets, the ground, and background. |
| Menu_GUI | Manages the Menu that the user navigates. Shows the start screen at the beginning, allows user to navigate to the menu for seed entry and starting the game with or without neural network. |
| Label | For use by the MENU_GUI. It is simply a coloured rectangle with text inside where the text, colour, position and dimensions of the box can be customised. |
| Button (inherits from Label) | Inherits from Label. It is a label that calls an event when clicked. |
| Textbox (inherits from Label) | Inherits from Label. It is a label that the user can write into. |
| Brain | Handles everything to do with neural networks. Also can write all of the weights of the neural network into a text file in a standard format, which can also be read from for use again. |
| Game_Manager | Bring everything together. Handles switch from game screen to menu screen. Handles training the neural network. |

| Class Name | Properties | Description |
|---|---|---|
| RNG | Seed (int) | The seeds that will be used for calculating. |
| | Mod (int) | The prime number used to mod the current value. |

| | | |
|---|---|---|
| | Multiplier (int) | The value that will be used to multiply the current value. |
| | Increment (int) | The amount the current value will be incremented by. |
| | Current (int) | The last value that was returned. |
| Projectile | Position_x (float) | The current x position of the projectile. |
| | Position_y (float) | The current y position of the projectile. |
| | Initial_position_x (float) | The initial x position of the projectile, needed in order to draw the path of the missile. |
| | Initial_position_y (float) | The initial y position of the projectile, needed in order to draw the path of the missile. |
| | Target_position_x (float) | The end x position that the missile is traveling to. |
| | Target_position_y (float) | The end y position that the missile is travelling to. |
| | Speed (float) | The speed at which the bullet should move towards its target position. |
| | Exploded (bool) | Stores whether the bullet has reached its target or not. |
| | Colour (string) | Stores the rgb value of missile trail for use drawing on the canvas. |
| | Dead (bool) | Stores whether the projectile should be removed from the game. |
| Missile (Projectile) | All properties of projectile | |
| Bullet (Projectile) | All properties of projectile | |
| Game | Missiles (List of Missile objects) | Contains a list of all the missiles currently in the game. |
| | Bullets (List of Bullet objects) | Contains a list of all the missiles currently in the game. |
| | Time_elapsed (int) | Stores the number of frames that have passes since the game has started. |
| | Hits (int) | Stores the number of missiles that have reached the ground. |
| | Score (int) | Stores the number of missiles that the player has hit. |
| | Neural_network_score (int) | Stores if necessary the number of missiles that have been hit by the neural network. |
| | Has_game_ended (bool) | Indicates whether the game has ended and thus the score screen should be brought up. |
| | Paused (bool) | Indicates whether the game has been paused. |

| | | |
|---|---|---|
| | RNG (RNG) | The random number generator used to calculate where the next missile will spawn and the time between missiles. |
| | Lives (int) | The number of times the player can be hit before the game ends. |
| Game_GUI | Root (pygame.display) | Contains the canvas in which the user will see and thus where the game will be drawn by this object. |
| | Game (Game) | The game class that it uses to get information about what to draw on the canvas. |
| Menu_GUI | Root (pygame.display) | Contains the canvas in which the user will see and thus where the game will be drawn by this object. |
| | Game_manager (Game_Manager) | Contains the Game_Manager object being used. |
| | Drawables (list) | A list of objects that have a draw function, thus all the objects can by drawn using polymorphism. |
| | Buttons (list) | Contains a list of all the button objects on the screen |
| | Text_box (text_box) | Contains the textbox on the screen if it exists (otherwise it is None). |
| Label | X1 (int) | Contains the x positions of the top left corner of the label. |
| | Y1 (int) | Contains the y position of the top left corner of the label. |
| | X2 (int) | Contains the x position of the bottom right corner of the label. |
| | Y2 (int) | Contains the y position of the bottom right corner of the label |
| | Colour (string) | Contains the rgb value of the colour of the fill of the label. |
| | Text (string) | Contains the text that is contained in the label when drawn. |
| Button (Label) | All properties of Label | |
| | Click_event (function) | Contains the event that is called if the button is clicked |
| | Hover_colour (string) | Contains the rgb value of the colour of the fill if the cursor is hovering over it |
| | Non_hover_colour (string) | Contains the rgb value of the colour of the fill if the cursor is not hovering over the button |
| Textbox (Label) | All properties of Label | |
| Brain | Number_of_input_nodes (int) | Contains the number of input nodes in the neural network |
| | Number_of_hidden_layers (int) | Contains the number of hidden layers in then neural network |

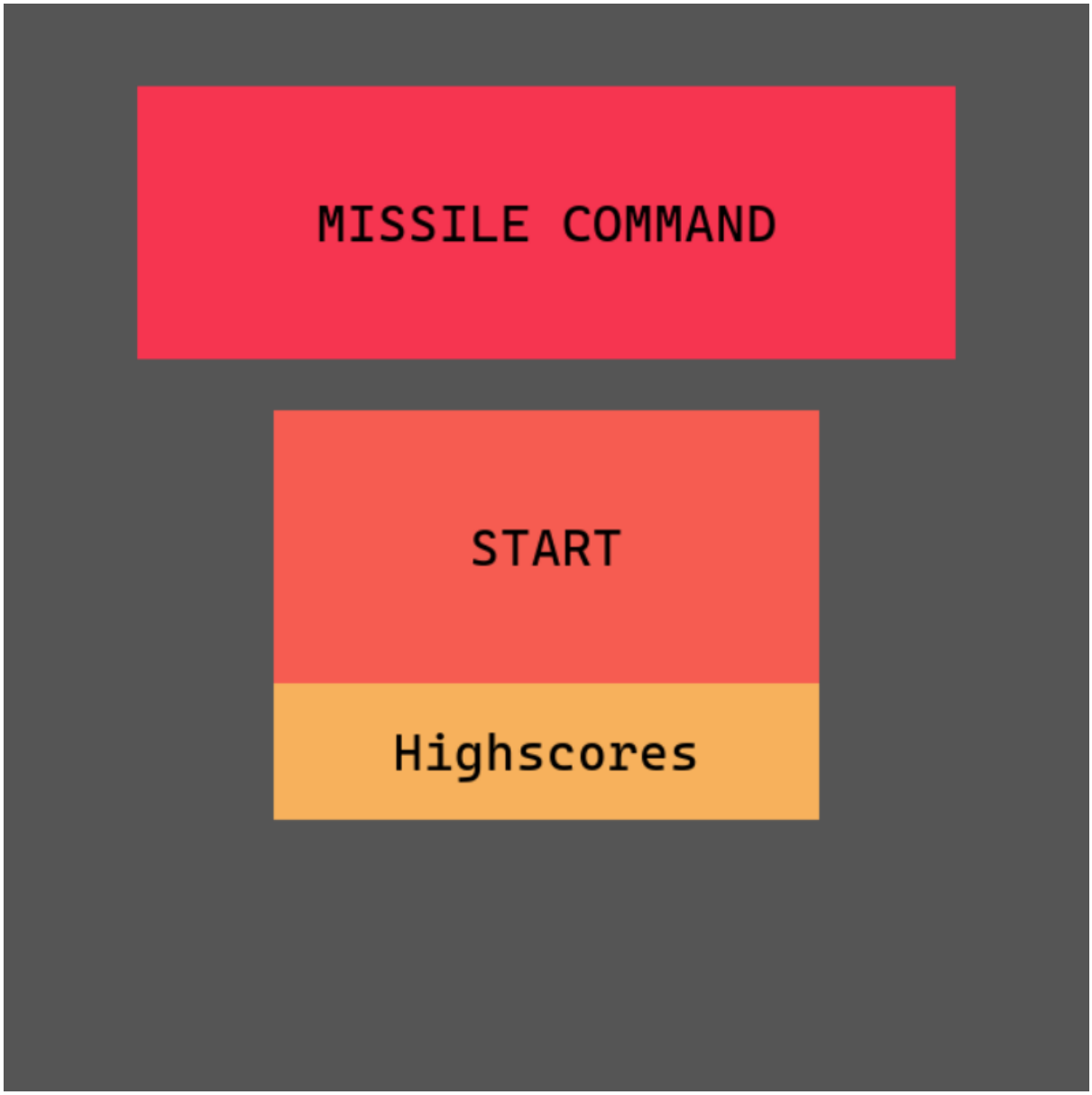
| | | |
|---|---|---|
| | Number_of_hidden_layer_nodes (int) | Contains the number of nodes in each hidden layers |
| | Number_of_output_nodes (int) | Contains the number of output nodes in the neural network. |
| | Size (int) | Contains the number of layers in the neural network. |
| | Layer_sizes (list of int) | Contains the number of nodes in each layer of the neural network. |
| | Nodes (list of (list of float)) | The current value of all the nodes in the neural network. |
| | Weights (list of (list of (list of float))) | Contains all the weights that connect two nodes in the neural network. |

| Class | Method | Description |
|---|---|---|
| RNG | Next | Generate the next number in the sequence of random. |
| Projectile | Draw | Draw the path of the missile on a given canvas |
| | Update | Update the position of the missile by 1 frame. |
| | Has_reached_target | Return whether the missile has reach the target inputted on initialization |
| Missile (Projectile) | All Projectile methods | |
| Bullet (Projectile) | All projectile methods | |
| Game | Motion | Called by pygame module when the mouse is moved, needed to keep track of the mouses position. |
| | On_click | Called by pygame module whenever a mouse button is clicked, needed to be able to shott bullets. |
| | Key_pressed | Called by pygame module whenever a key is pressed. |
| | Create_next_missile | Using the RNG object given it will spawn a random missile on the screen as describe above. |
| | Create_next_bullet | Creates a bullet on the screen given two co-ordinates |
| | Update | Advances every object in the game by one frame, e.g moves the bullets and missiles by 1 frame. |
| Game_GUI | Draw | Draw every object onto the screen from the information of a Game object. |
| Menu_GUI | Update | Called every frame there is a menu active. |

| | | |
|---|---|---|
| | Initialize_start_menu | Initializes the first menu that is shown when the game starts |
| | Initialize_seed_menu | Initializes the menu that is used when the user is choosing a seed. |
| | Initialize_score_menu | Initializes the menu that is shown to show the users score. |
| | Start_game | Calls the start_game function of the game_manager object |
| | Start_game_with_neural_network | Calls the start_game_with_neural_network function in the game_manager object. |
| | On_click | Called by pygamem module whenever the mouse is clicked. |
| | Key_pressed | Called by the pygame module whenever a key is pressed |
| | Draw | Given a canvas it will draw every object in the menu on it. |
| Label | Draw | Draw the label on a given canvas |
| Button (Label) | All Label Functions | |
| | On_click | Called by the menu whenever the mouse is click to check whether the click_event function should be called |
| | Update | Called every frame to check whether the mouse is hovering over it and thus the colour should be changed. |
| Textbox (Label) | All label functions | |
| | Append_letter | Adds the argument onto the end of the label text string |
| | Pop_text | Removes the last letter from the label text string |
| Brain | Set_random_weights | Sets all the weights in the neural network to a random value from -1 to 1. |
| | Set_input | Sets the input layer of the neural network to the values given in the argument |
| | Get_output | Returns the values of all the output nodes int the neural network |
| | Forward_propagate | After the inputs have been entered this runs the forward propagation of the neural network to generate the output layer of the neural network. |

| | | |
|---|---|---|
| | Back_propagate | Given some expected values form for the output layer it will calculate the error from each node and adjust the weights accordingly to train the neural network. |
| | Copy | It will copy the structure and weights of another given brain. |
| | Write_to_text_file | Will write all the weights of the neural network into a text file in a normalized format. |
| | Copy_from_text_file | Given a text file which contains the weights of another neural network in the correct normalized format it will copy all the weights so that they can be re-used. |

# Technical Solution



## Game_GUI
Page: 43

This class heavily relies on using a Game object to obtain all the information that it can to accurately draw the game.

## Label
Page: 45

## Button
Page: 47

## Textbox
Page: 48

## Menu_GUI
Page: 49

The way I decided to go about organising everything to be drawn in the menu was to give every object that will be shown on the menu a draw() function. This means that for whatever is currently being shown on the menu I can simply add them all into a list and then iterate through them calling their draw() functions. This works as python supports the technique polymorphism.

## Brain
Page: 54

I chose the value of the LEARNING_RATE here through trial and error. Using a value significantly less than 0.1 as taking too long to train and using a value more than 0.1 lead to a poor approximation of the Q function. Thus I landed on using a value of 0.1.

## Game_Manager
Page: 59

Here I chose the value of the GAMMA constant through trial and error. It is the constant that is used in the Q function and it indicates how much the neural network should value the past when considering which action to take. Values above and below 0.69 converged onto solution which were worse than what it converged to with 0.69. The constant EPSILON here I also chose through trial and error, it is used to indicate the probability of the neural network choosing a random action in each state, it is useful to make sure that the neural network is "exploring" sufficiently. Here I found that the value of 0.05 had the neural network converge onto the best solution. Anything larger would be exploring too much to exploit the best option and anything smaller would converge too slowly.

```
#RNG

#list of the 256 largest primes less than 100000
PRIMES = [99991, 99989, 99971, 99961, 99929, 99923, 99907, 99901, 99881, 99877,
99871, 99859, 99839, 99833, 99829, 99823, 99817, 99809, 99793, 99787, 99767,
99761, 99733, 99721, 99719, 99713, 99709, 99707, 99689, 99679, 99667, 99661,
99643, 99623, 99611, 99607, 99581, 99577, 99571, 99563, 99559, 99551, 99529,
99527, 99523, 99497, 99487, 99469, 99439, 99431, 99409, 99401, 99397, 99391,
99377, 99371, 99367, 99349, 99347, 99317, 99289, 99277, 99259, 99257, 99251,
99241, 99233, 99223, 99191, 99181, 99173, 99149, 99139, 99137, 99133, 99131,
99119, 99109, 99103, 99089, 99083, 99079, 99053, 99041, 99023, 99017, 99013,
98999, 98993, 98981, 98963, 98953, 98947, 98939, 98929, 98927, 98911, 98909,
98899, 98897, 98893, 98887, 98873, 98869, 98867, 98849, 98837, 98809, 98807,
98801, 98779, 98773, 98737, 98731, 98729, 98717, 98713, 98711, 98689, 98669,
98663, 98641, 98639, 98627, 98621, 98597, 98573, 98563, 98561, 98543, 98533,
98519, 98507, 98491, 98479, 98473, 98467, 98459, 98453, 98443, 98429, 98419,
98411, 98407, 98389, 98387, 98377, 98369, 98347, 98327, 98323, 98321, 98317,
98299, 98297, 98269, 98257, 98251, 98227, 98221, 98213, 98207, 98179, 98143,
98129, 98123, 98101, 98081, 98057, 98047, 98041, 98017, 98011, 98009, 97987,
97973, 97967, 97961, 97943, 97931, 97927, 97919, 97883, 97879, 97871, 97861,
97859, 97849, 97847, 97843, 97841, 97829, 97813, 97789, 97787, 97777, 97771,
97729, 97711, 97687, 97673, 97651, 97649, 97613, 97609, 97607, 97583, 97579,
97577, 97571, 97561, 97553, 97549, 97547, 97523, 97511, 97501, 97499, 97463,
97459, 97453, 97441, 97429, 97423, 97397, 97387, 97381, 97379, 97373, 97369,
97367, 97327, 97303, 97301, 97283, 97259, 97241, 97231, 97213, 97187, 97177,
97171, 97169, 97159, 97157, 97151, 97127, 97117, 97103, 97081, 97073, 97039,
97021, 97007, 97003, 97001]

class RNG():
    @property
    def seed(self):
        return self._seed

    @seed.setter
    def seed(self, v):
        v %= (2**32)
        self._seed = v

        self._mod = PRIMES[v % (2**8)]
        v //= 2**8
        self._multiplier = v % (2**8) + 12345
        v //= 2**8
        self._increment = v % (2**8) + 23457
        v //= 2**8
        self._current = v % (2**8) * 390

        self._uses = 1

    #returns a random number in the range l to r
    def next(self, l, r):
        dif = (r - l) / self._mod
        val = dif * self._current + l

        #performs the operation 7 times to make it more random
        for i in range (7):
            self._current = (self._current * self._multiplier + self._increment)
% self._mod

        #changes the value of the increment based on the number of uses
```

```
36          #this is to make seeds less like each other
37          self._increment *= self._uses
38          self._increment += 1
39          self._increment %= self._mod
40          self._uses += 1
41          return val
```

```
#Projectile

import Game
import math
import Game_Manager
import pygame

EXPLOSION_TIME = 1000

class Projectile():
    @property
    def dead(self):
        return self._dead

    @property
    def time_since_explosion(self):
        return self._time_since_explosion

    @property
    def colour(self):
        return self._colour

    @colour.setter
    def colour(self, v):
        self._colour = v

    @property
    def position_x(self):
        return self._position_x

    @position_x.setter
    def position_x(self, value):
        self._position_x = value
        if (not hasattr(self, "_initial_position_x")):
            self._initial_position_x = value

    @property
    def position_y(self):
        return self._position_y

    @position_y.setter
    def position_y(self, value):
        self._position_y = value
        if (not hasattr(self, "_initial_position_y")):
            self._initial_position_y = value

    @property
    def initial_position_x(self):
        return self._initial_position_x

    @property
    def initial_position_y(self):
        return self._initial_position_y

    @property
    def target_position_x(self):
        return self._target_position_x

    @target_position_x.setter
```

```
60      def target_position_x(self, value):
61          self._target_position_x = value
62
63      @property
64      def target_position_y(self):
65          return self._target_position_y
66
67      @target_position_y.setter
68      def target_position_y(self, value):
69          self._target_position_y = value
70
71      @property
72      def speed(self):
73          return self._speed
74
75      @speed.setter
76      def speed(self, value):
77          self._speed = value
78
79      @property
80      def exploded(self):
81          return self._exploded
82
83      def __init__(self):
84          self._dead = False
85          self._time_since_explosion = 0
86          self._exploded = False
87
88      def draw(self, surface):
89          scale = Game_Manager.SCALE
90
91          #if the projectile has not reached its target then a line will be drawn
    from its initial to end position
92          if (not self.exploded):
93              pygame.draw.line(surface, self._colour,
94                  (self.initial_position_x * scale,
95                  self.initial_position_y * scale),
96                  (self.position_x * scale,
97                  self.position_y * scale),
98                  width = 20
99              )
100
101         else:
102             """
103             if a projectile has reached its target then the following animation
    should play
104             the animation consist of a circle which starts the size of the
    projectile explosion radius and shrinks to the nothing at a constant rate
105             """
106             r = ((Game.EXPLOSION_RADIUS * EXPLOSION_TIME -
    Game.EXPLOSION_RADIUS * self.time_since_explosion) / EXPLOSION_TIME)
107
108             circle = pygame.Rect(((self._position_x - r) * scale,
    (self.position_y - r) * scale),
109                                  ((r * 2 * scale, r * 2 * scale)))
110
111             pygame.draw.ellipse(surface, self._colour, circle)
112
113     """
```

```
114     determines the direction in which the bullet should travel to reach its
    target
115     converts the value into a vector of velocity
116     updates position accordingly
117     """
118     def update(self):
119         #number of ms per frame
120         delta_time = 1 / Game.FRAME_RATE
121 
122         """
123         if the projectile has reached its target it will keep track of how long
    it has been since then so that it can animate the explosion
124         """
125         if (self.has_reached_target()):
126             self._time_since_explosion += delta_time * 1000
127             """
128             if the projectile animation has ended it will be marked as dead as
    to indicate that it should be removed from the game
129             """
130             if (self._time_since_explosion > EXPLOSION_TIME):
131                 self._dead = True
132             return
133 
134         #change in x and y position
135         delta_x = self.target_position_x - self.position_x
136         delta_y = self.target_position_y - self.position_y
137 
138         magnitude = math.sqrt(delta_x * delta_x + delta_y * delta_y)
139 
140         #calculates the x and y components of the velocity
141         velocity_x = (delta_x / magnitude) * self.speed
142         velocity_y = (delta_y / magnitude) * self.speed
143 
144         #updates position by 1 frame
145         self._position_x += velocity_x * delta_time
146         self._position_y += velocity_y * delta_time
147 
148     #this is to check whether the projectile has reached its target
149     def has_reached_target(self):
150         """
151         if initial position y is greater than the target position y then it is
    sufficient to check whether the current position y is less than the target
    position y
152 
153         if the initial position y is less than the target position y then it is
    sufficient to check whether the current position y is greater than the target
    position y
154         """
155         if (self.position_y ≤ self.target_position_y and
    self.initial_position_y ≥ self.target_position_y):
156             self._exploded = True
157             return True
158 
159         if (self.position_y ≥ self.target_position_y and
    self.initial_position_y ≤ self.target_position_y):
160             self._exploded = True
161             return True
162 
163         return False
```

```python
#Missile

import Game
import random
import math

from Projectile import Projectile

#here Missile inherits from Projectile
class Missile(Projectile):
    def __init__(self):
        super().__init__()
        self._colour = "#C62A88"

    """
    this will set the position of the missile to a random legal location
    this has use only in testing as the final product will need to use a seeding system
    """
    def set_random_position(self):
        self._position_x = random.uniform(0, 16)
        self._position_y = 0
        self._target_position_x = random.uniform(0, 16)
        self._target_position_y = Game.GROUND_HEIGHT
        self._initial_position_x = self.position_x
        self._initial_position_y = self.position_y
        self.speed = Game.MISSILE_SPEED
```

```python
#Bullet
import math
import Game
from Projectile import Projectile

#here Bullet inherits from Projectile
class Bullet(Projectile):
    def __init__(self, is_neural_network_bullet=False):
        self._is_neural_network_bullet = is_neural_network_bullet

        super().__init__()
        if (is_neural_network_bullet):
            self._colour = "#03C4A1"
        else:
            self._colour = "#590995"
```

```python
#Game

from Missile import *
from Bullet import *
import Game_Manager
import pygame

GAME_WIDTH = 16
GAME_HEIGHT = 16
MISSILE_SPEED = 5
BULLET_SPEED = 20
FRAME_RATE = 144
EXPLOSION_RADIUS = 2
GROUND_HEIGHT = 15
LIVES = 5

class Game():
    def __init__(self):
        self._missiles = []
        self._bullets = []

        self._explosions = []
        self._time_elapsed = 0
        self._next_missile = 0
        self._hits = 0
        self._score = 0
        self._neural_network_score = 0
        self._has_game_ended = False
        self._paused = False

    @property
    def has_game_ended(self):
        return self._has_game_ended

    @property
    def missiles(self):
        return self._missiles

    @property
    def bullets(self):
        return self._bullets

    @property
    def explosions(self):
        return self._explosions

    @property
    def RNG(self):
        return self._RNG

    @RNG.setter
    def RNG(self, val):
        self._RNG = val

    @property
    def lives(self):
        return LIVES - self._hits

    @property
```

```
60      def score(self):
61          return self._score
62
63          #called whenever mouse is moved
64      def motion(self, event):
65          self.mouse_x = event.x
66          self.mouse_y = event.y
67
68      #called whenever mouse is clicked
69      def on_click(self):
70          if (self._paused):
71              return
72
73          """
74          when somewhere on the screen is clicked the game will create a missile
    at the current x and y coordinates of the mouse
75          pygame has a build in function for retrieving the mouses position
    relative to the top left corner of the window which I use here
76          """
77          x , y = pygame.mouse.get_pos()
78          x //= Game_Manager.SCALE
79          y //= Game_Manager.SCALE
80          self.create_bullet(x, y)
81
82      #called whenever a button on the keyboard is pressed
83      def key_pressed(self, event):
84          if (event == pygame.K_ESCAPE):
85              self._has_game_ended = True
86
87      #spawns a missile on the screen with random starting position and random
    target position
88      def create_random_missile(self):
89          m = Missile()
90          m.set_random_position()
91          self.missiles.append(m)
92
93      #spawns a missile using the random number generator with the current seed
94      def create_next_missile(self):
95          m = Missile()
96          m.position_x = self._RNG.next(0, GAME_WIDTH)
97          m.position_y = 0
98          m.target_position_x = self._RNG.next(0, GAME_WIDTH)
99          m.target_position_y = GROUND_HEIGHT
100         m.speed = MISSILE_SPEED
101         self._missiles.append(m)
102
103     #spawns a bullet using the x and y coordinates given, this method will be
    called on every mouse click
104     def create_bullet(self, target_position_x, target_position_y,
    is_neural_network_bullet=False):
105         b = Bullet(is_neural_network_bullet)
106         b.target_position_x = target_position_x
107         b.target_position_y = target_position_y
108         b.speed = BULLET_SPEED
109         b.position_x = GAME_WIDTH / 2
110         b.position_y = GROUND_HEIGHT
111         self._bullets.append(b)
112
113     #this function will be called every frame to advance the state of the game
    by one frame
```

```
114     def update(self):
115         if (self._paused):
116             return
117 
118         delta_time = int(1000 / FRAME_RATE) #time in ms between frames
119         self._time_elapsed += 1 #stores the number of frames that have passed
120 
121         #spawns a missile if it is time for one
122         #here self._next_missile represents the number of frames that should
    have passed before the next missile is to spawn
123         if (self._time_elapsed > self._next_missile):
124             self.create_next_missile()
125             center = pow(0.99985, self._time_elapsed) * 500
126             self._next_missile += self.RNG.next(center - 25, center + 25) #adds
    a random number which is when then next missile will spawn
127 
128         for m in self._missiles:
129             m.update()
130 
131         #list will contain the indexes of all bullets that have reached their
    target
132         terminated_bullets = []
133         for i in range(len(self.bullets)):
134             #advances the bullets by one frame
135             self.bullets[i].update()
136             if (not self.bullets[i].exploded and
    self.bullets[i].has_reached_target()):
137                 terminated_bullets.append(i)
138 
139         """
140         removes all the bullets that have reached their target
141 
142         every missile that is within the explosion radius of the bullets that
    have exploded are destroyed
143         to do this the program iterates through every bullet that has reached
    its target and for each bullet it iterates through every missile on screen
    checks whether it is in range using pythagoras' theorm
144         if the missile is in range 1 is added to the score
145 
146         it is essential that the missiles are iterated in reverse otherwise
    some missile may not be considered
147         """
148 
149         for i in reversed(terminated_bullets):
150             for j in range(len(self.missiles) - 1, -1, -1):
151                 #x distance between missile and bullet
152                 difx = self.missiles[j].position_x - self.bullets[i].position_x
153                 #y distance between missile and bullet
154                 dify = self.missiles[j].position_y - self.bullets[i].position_y
155                 if (difx * difx + dify * dify ≤ EXPLOSION_RADIUS *
    EXPLOSION_RADIUS):
156                     del self.missiles[j]
157                     if (not self.bullets[i]._is_neural_network_bullet):
158                         self._score += 1
159                     else:
160                         self._neural_network_score += 1
161 
162         """
163         here it will removed all the bullets that have reached their target and
    all the missile that have either
```

```
164         reached their target or been destroyed
165
166         it is essential that the bullets are iterated in reverse otherwise the
    indices of the terminated bullets
167         would change cause the wrong bullets to be considered
168         """
169         for i in range(len(self.bullets) - 1, - 1, - 1):
170             if (self.bullets[i].dead):
171                 del self.bullets[i]
172
173         for i in range(len(self.missiles) - 1, -1, -1):
174             if (not self.missiles[i].exploded and
    self.missiles[i].has_reached_target()):
175                 self._hits += 1
176
177             if (self.missiles[i].dead):
178                 del self.missiles[i]
179
180         #if the player has been hit enough times a flag is set to inform the
    game manager
181         if (self._hits ≥ LIVES):
182             self._has_game_ended = True
```

```
#Game_GUI

from logging import exception
import pygame
import Game
import Game_Manager
import Label

_TEXT_FONT_SIZE = 30
_TEXT_FONT = "Helvetica"
_TEXT_COLOUR = "#177013"

_HITS_POSITION_Y = 32
_SCORE_POSITION_Y = 96

class Game_GUI():
    #game object that this object will use to draw
    @property
    def game(self):
        return self._game

    @game.setter
    def game(self, new_game):
        self._game = new_game

    #canvas on which everything should be drawn
    @property
    def root(self):
        return self._root

    def __init__(self, root):
        self.game = None
        self._root = root

        #this stores the number of milliseconds per frame
        self.delta_time = (1000 / Game.FRAME_RATE)
        #score box to show player score
        self._score_box = Label.Label(5, 5, 100, 100)
        self._score_box._colour = "#590996"

        #score box to show AI score
        self._score_box2 = Label.Label(5, 105, 100, 200)
        self._score_box2._colour = "#03C4A0"

        #text box to show the number of lives the player has remaining
        self._lives_box = Label.Label(535, 5, 635, 100)
        self._lives_box._colour = "#C62A87"

    def draw(self):
        if (self.game == None):
            raise exception("No game class assigned")

        #loads the background image
        #background = pygame.image.load("Assets\\space bg.jpg")
        #draws the background image onto the screen
        #self._root.blit(background, (0, 0))

        self._root.fill("#222222")

```

```
60          #draws missile object onto the screen
61          for m in self.game.missiles:
62              m.draw(self._root)
63
64          #draws bullet objects onto the screen
65          for b in self.game.bullets:
66              b.draw(self._root)
67
68          #draws the ground
69          ground = pygame.Rect((0, Game_Manager.SCALE * Game.GROUND_HEIGHT),
70                               (Game_Manager.RESOLUTION_X + 2, (Game.GAME_HEIGHT -
Game.GROUND_HEIGHT) * Game_Manager.SCALE + 2))
71
72          pygame.draw.rect(self._root, "#AAAAAA", ground)
73
74          #sets the scorebox text to be the current score of the player and then
draws it
75          self._score_box.text = str(self.game._score)
76          self._score_box.draw(self._root)
77
78          #sets the other scorebox text to be the current score of the neural
network and then draws it onto the screen
79          self._score_box2.text = str(self._game._neural_network_score)
80          self._score_box2.draw(self.root)
81
82          self._lives_box.text = str(self._game.lives)
83          self._lives_box.draw(self.root)
```

```
#Label

import pygame

class Label():
    @property
    def text(self):
        return self._text

    @text.setter
    def text(self, v):
        self._text = v

    @property
    def text_start_pos_y(self):
        return self._text_start_pos_y

    @text_start_pos_y.setter
    def text_start_pos_y(self, v):
        self._text_start_pos_y = v

    @property
    def colour(self):
        return self._colour

    @colour.setter
    def colour(self, v):
        self._colour = v

    """
    top left corner of label is (x1, y1) in pixels
    bottom right corner of label is (x2, y2) in pixels
    """
    def __init__(self, x1, y1, x2, y2):
        self._x1 = x1
        self._y1 = y1
        self._x2 = x2
        self._y2 = y2
        self._colour = "#ff0000"

        self._text_start_pos_y = int((self._y1 + self._y2) / 2)

        #this property contains the function that will be called if the button is clicked
        self._click_event = None

    def draw(self, surface):
        #draws the rectangle of the label
        box = pygame.Rect((self._x1, self._y1),
                          (self._x2 - self._x1, self._y2 - self._y1))

        pygame.draw.rect(surface, self._colour, box)

        """
        if the label has a text string
        it will draw the text text by default into the middle of the box
        although the text_start_position_y position can be changed to draw the text somewhere else
        """
```

```
58          if (self._text ≠ None):
59              font = pygame.font.Font("Assets\\Cascadia.ttf", 30)
60
61              words = self._text.split("\n")
62              for i in range(len(words)):
63                  text = font.render(words[i], True, "#000000")
64                  textrect = text.get_rect(center=(int((self._x1 + self._x2) / 2),
self._text_start_pos_y + i * 50))
65
66                  surface.blit(text, textrect)
```

```python
#Button

from Label import Label
import pygame

#here Button inherits from Label
class Button(Label):
    #function to be called when the button is clicked, e.g. start game
    @property
    def click_event(self):
        return self._click_event

    @click_event.setter
    def click_event(self, v):
        self._click_event = v

    #colour to be displayed when the user has their cursor over the button
    @property
    def hover_colour(self):
        return self._hover_colour

    @hover_colour.setter
    def hover_colour(self, v):
        self._hover_colour = v

    #colour to be displayed when the user does not have their cursor over the button
    @property
    def non_hover_colour(self):
        return self._non_hover_colour

    @non_hover_colour.setter
    def non_hover_colour(self, v):
        self._non_hover_colour = v


    #should accept x and y values of mouse
    #checks if the user has click on the button and should thus
    #call the click event
    def on_click(self, x, y):
        if (x > self._x1 and y > self._y1 and x < self._x2 and y < self._y2):
            if (self._click_event == None):
                raise Exception("no on_click event assigned")
            self._click_event()
            return True
        return False

    #if the user currently has their mouse hovering over the button
    #it should display the non hover position
    def update(self):
        x, y = pygame.mouse.get_pos()

        if (x > self._x1 and y > self._y1 and x < self._x2 and y < self._y2):
            self._colour = self._hover_colour
        else:
            self._colour = self._non_hover_colour
```

```python
#Textbox

from Label import Label
import pygame

#here Textbox inherits from Label
class Textbox(Label):
    def __init__(self, x1, y1, x2, y2):
        self._text = ""

        super().__init__(x1, y1, x2, y2)

    """
    the text in the textbox is essentially treated as a stack
    the letters are pushed to the end of the string and if the user presses backspace then it removes the last character
    """
    def key_pressed(self, key):
        if (key == pygame.K_BACKSPACE):
            self.pop_text()
        else:
            self.append_letter(chr(key))

    #pushes character onto the end of text
    def append_letter(self, c):
        self._text += c

    #pops last character from the end of the text
    def pop_text(self):
        self._text = self._text[0:-1]
```

```
#Menu_GUI

from logging import exception
import random
import Game_Manager
import Button
import Textbox
import Label
import pygame

START_BUTTON_SIZE_X = 320
START_BUTTON_SIZE_Y = 160
SEED_TEXTBOX_SIZE_Y = 80
TITLE_SIZE_X = 480
TITLE_SIZE_Y = 160
TITLE_POSITION_Y = 50

class Menu_GUI:
    def __init__(self, game_manager, root):
        """
        this list will contain everything that has a draw function
        thus everything can be drawn onto the screen by iterating through the list and calling draw through polymorphism
        """
        self._drawables = []
        #this list will contain every button currently on the screen
        self._buttons = []

        #this is the canvas on which everything will be drawn
        self._root = root
        self._game_manager = game_manager
        self._textbox = None

    def update(self):
        for b in self._buttons:
            b.update()

    def initialize_start_menu(self):
        self._drawables.clear()
        self._buttons.clear()
        self._textbox = None

        top_left_x = Game_Manager.RESOLUTION_X / 2 - TITLE_SIZE_X / 2
        top_left_y = TITLE_POSITION_Y

        #label that displays the title of the game
        title_label = Label.Label(top_left_x, top_left_y, top_left_x + TITLE_SIZE_X, top_left_y + TITLE_SIZE_Y)
        title_label.text = "MISSILE COMMAND"
        title_label.colour = "#F63550"
        self._drawables.append(title_label)

        top_left_x = Game_Manager.RESOLUTION_X / 2 - START_BUTTON_SIZE_X / 2
        top_left_y = Game_Manager.RESOLUTION_Y / 2 - START_BUTTON_SIZE_Y / 2

        #button that you click to start the game
        start_button = Button.Button(top_left_x, top_left_y, top_left_x + START_BUTTON_SIZE_X, top_left_y + START_BUTTON_SIZE_Y)
        start_button.text = "START"
```

```
        start_button.click_event = self.initialize_seed_menu
        start_button.colour = "#F65C51"
        start_button.non_hover_colour = "#F65C51"
        start_button.hover_colour = "#E54B40"
        self._drawables.append(start_button)
        self._buttons.append(start_button)

        #button that brings up the leaderboard screen
        leaderboard_button = Button.Button(top_left_x, top_left_y + START_BUTTON_SIZE_Y, top_left_x + START_BUTTON_SIZE_X, top_left_y + (int)(1.5 * START_BUTTON_SIZE_Y))
        leaderboard_button.text = "Highscores"
        leaderboard_button.click_event = self.initialize_high_score_screen
        leaderboard_button.colour = "#F7B15C"
        leaderboard_button.non_hover_colour = "#F7B15C"
        leaderboard_button.hover_colour = "#E6A04B"
        self._drawables.append(leaderboard_button)
        self._buttons.append(leaderboard_button)

    def initialize_high_score_screen(self):
        self._drawables.clear()
        self._buttons.clear()
        self._textbox = None

        #button to return to home menu
        back_button = Button.Button(5, 5, 100, 100)
        back_button.text = None
        back_button.click_event = self.initialize_start_menu
        back_button.colour = "#E5243F"
        back_button.non_hover_colour = "#E5243F"
        back_button.hover_colour = "#D4132E"
        self._drawables.append(back_button)
        self._buttons.append(back_button)

        #label that shows the high scores
        high_score_box = Label.Label(105, 5, 635, 635)
        high_score_text = ""
        for score in reversed(self._game_manager.high_scores):
            high_score_text += str(score)
            high_score_text += "\n"
        high_score_box.text = high_score_text
        high_score_box.colour = "#ffffff"
        high_score_box.text_start_pos_y = 50
        self._drawables.append(high_score_box)

    def initialize_seed_menu(self):
        self._buttons.clear()
        self._drawables.clear()
        self._textbox = None

        top_left_x = Game_Manager.RESOLUTION_X / 2 - START_BUTTON_SIZE_X / 2
        top_left_y = Game_Manager.RESOLUTION_Y / 2 - START_BUTTON_SIZE_Y / 2

        #button that starts the game without neural network
        start_button = Button.Button(top_left_x, top_left_y, top_left_x + START_BUTTON_SIZE_X, top_left_y + START_BUTTON_SIZE_Y)
        start_button.text = "START"
        start_button.click_event = self.start_game
        start_button.colour = "#F7B15C"
        start_button.non_hover_colour = "#F7B15C"
```

```
114         start_button.hover_colour = "#E6A04B"
115         self._drawables.append(start_button)
116         self._buttons.append(start_button)
117 
118         #button that starts the game with neural network
119         start_button_2 = Button.Button(top_left_x, top_left_y +
START_BUTTON_SIZE_Y, top_left_x + START_BUTTON_SIZE_X, top_left_y + (int)(1.5 *
START_BUTTON_SIZE_Y))
120         start_button_2.text = "START WITH AI"
121         start_button_2.click_event = self.start_game_with_neural_network
122         start_button_2.colour = "#F65C51"
123         start_button_2.non_hover_colour = "#F65C51"
124         start_button_2.hover_colour = "#E54B40"
125         self._drawables.append(start_button_2)
126         self._buttons.append(start_button_2)
127 
128         #textbox where seed can be entered
129         seed_textbox = Textbox.Textbox(top_left_x, top_left_y -
SEED_TEXTBOX_SIZE_Y, top_left_x + START_BUTTON_SIZE_X, top_left_y)
130         seed_textbox._colour = "#ffffff"
131         self._textbox = seed_textbox
132         self._drawables.append(seed_textbox)
133 
134         #button to return to home menu
135         back_button = Button.Button(5, 5, 100, 100)
136         back_button.text = None
137         back_button.click_event = self.initialize_start_menu
138         back_button.colour = "#E5243F"
139         back_button.non_hover_colour = "#E5243F"
140         back_button.hover_colour = "#D4132E"
141         self._drawables.append(back_button)
142         self._buttons.append(back_button)
143 
144     #here the score and seed arguments contain respectively the text to be
display and seed to be display
145     def initialize_score_screen(self, score, seed):
146         self._drawables.clear()
147         self._buttons.clear()
148         self._textbox = None
149 
150         seed_box_size_x = 480
151         seed_box_size_y = 80
152         seed_box_centre_y = 120
153         seed_box_top_left_x = Game_Manager.RESOLUTION_X / 2 - seed_box_size_x /
2
154         seed_box_top_left_y = seed_box_centre_y - seed_box_size_y / 2
155 
156         #contains the label which will show the seed sued
157         seed_box = Label.Label(seed_box_top_left_x, seed_box_top_left_y,
seed_box_top_left_x + seed_box_size_x, seed_box_top_left_y + seed_box_size_y)
158         seed_box.text = "SEED: " + str(seed)
159         seed_box.colour = "#ffffff"
160         self._drawables.append(seed_box)
161 
162         score_box_size_x = 480
163         score_box_size_y = 80
164         score_box_centre_y = 200
165         score_box_top_left_x = Game_Manager.RESOLUTION_X / 2 - score_box_size_x
/ 2
166         score_box_top_left_y = score_box_centre_y - score_box_size_y / 2
```

```
        #contains the label which will show the score used
        score_box = Label.Label(score_box_top_left_x, score_box_top_left_y, score_box_top_left_x + score_box_size_x, score_box_top_left_y + score_box_size_y)
        score_box.text = score
        score_box.colour = "#ffffff"
        self._drawables.append(score_box)

        menu_button_top_left_x = Game_Manager.RESOLUTION_X / 2 - START_BUTTON_SIZE_X / 2
        menu_button_top_left_y = Game_Manager.RESOLUTION_Y / 2 - START_BUTTON_SIZE_Y / 2

        #contains the button which when pressed returns to the home menu
        menu_button = Button.Button(menu_button_top_left_x, menu_button_top_left_y, menu_button_top_left_x + START_BUTTON_SIZE_X, menu_button_top_left_y + START_BUTTON_SIZE_Y)
        menu_button.text = "MENU"
        menu_button.click_event = self.initialize_start_menu
        menu_button.colour = "#E5243F"
        menu_button.non_hover_colour = "#E5243F"
        menu_button.hover_colour = "#D4132E"
        self._drawables.append(menu_button)
        self._buttons.append(menu_button)

    def start_game_with_neural_network(self):
        if (self._textbox == None):
            raise exception("no textbox object")

        """
        if the user has written something in the textbox then it will use it as the seed
        other wise it will use a random integer from 0 to 2^32
        """
        if (self._textbox.text == ""):
            seed = random.randint(0, 2**32)
        else:
            try:
                seed = int(self._textbox.text)
            except:
                seed = random.randint(0, 2**32)

        self._game_manager.start_game_with_neural_network(seed)

    def start_game(self):
        if (self._textbox == None):
            raise exception("no textbox object")

        """
        if the user has written something in the textbox then it will use it as the seed
        other wise it will use a random integer from 0 to 2^32
        """
        if (self._textbox.text == ""):
            seed = random.randint(0, 2**32)
        else:
            try:
                seed = int(self._textbox.text)
            except:
```

```
                seed = random.randint(0, 2**32)

        self._game_manager.start_game(seed)

    #called whenever mouse is clicked
    def on_click(self):
        for b in self._buttons:
            if b.on_click(pygame.mouse.get_pos()[0], pygame.mouse.get_pos()
[1]):
                break

    #called whenever a button on the keyboard is pressed
    def key_pressed(self, event):
        if (self._textbox ≠ None):
            self._textbox.key_pressed(event)

    def draw(self):
        #background = pygame.image.load("Assets\\space bg.jpg")
        #self._root.blit(background, (0, 0))
        self._root.fill("#555555")

        """
        this uses polymorphism

        everything object that needs to be drawn on the screen will have a draw
method
        thus every object that is currently on the screen will be added to the
list called drawables
        and then the program will iterate through every object in that list and
call the draw function on it drawing it on the canvas
        """
        for d in self._drawables:
            d.draw(self._root)
```

```python
#Brain

from cmath import inf
import numpy as np
import random

#a constant which indicates how fast the neural network will converge on a solution
LEARNING_RATE = 0.1
e = 2.7182818284590452353602874713527

#assuming n is a value calculated by the sigmoid function this function returns the derivative at that point
def transfer_derivative(n):
    return n * (1 - n)

def sigmoid(n):
    return 1 / (1 + e**(-n))

kernel = [[1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1]]

#given set of rgb values from the screen this function will simplify it into a much smaller sized structure to be fed into a neural network
#it will do this by splitting the screen into regions and returning the max value of that region
"""
this function uses recursion to downscale the screen
the argument times is the number of times the function will call itself
on each call it returns a screen scaled down by the size of the kernel
"""
def convolute(screen, times):
        size_x = len(kernel[0])
        size_y = len(kernel)

        res = []
        for i in range(0, len(screen[0]), size_x):
            lst = []
            for j in range(0, len(screen), size_y):
                sum = 0
                mx = -inf
                for x in range(i, i + size_x, 1):
                    if (x ≥ len(screen[0])):
                        break

                    for y in range(j, j + size_y, 1):
                        if (y ≥ len(screen)):
                            break

                        sum += screen[x][y]
                        if (screen[x][y] > mx):
                            mx = screen[x][y]
                lst.append(mx)
            res.append(lst)

        if (times > 0):
```

```
            return convolute(res, times - 1)
        return res

class Brain():
    """this is a ffnn"""

    def __init__(self, number_of_input_nodes, number_of_hidden_layer_nodes, number_of_hidden_layers, number_of_output_nodes):
        self._number_of_input_nodes = number_of_input_nodes
        self._number_hidden_layer_nodes = number_of_hidden_layer_nodes
        self._number_of_hidden_layers = number_of_hidden_layers
        self._number_of_output_nodes = number_of_output_nodes

        self._size = 2 + number_of_hidden_layers

        self._layer_sizes = []
        self._layer_sizes.append(number_of_input_nodes)
        for i in range(number_of_hidden_layers):
            self._layer_sizes.append(number_of_hidden_layer_nodes)
        self._layer_sizes.append(number_of_output_nodes)

        """
        stores all the stats of the nodes of  the neural networks as nx1 matrices
        stores states of all the weights as nxm matrices

        self._nodes[0] is the inputs nodes
        self._nodes[-1] is the output nodes
        the rest are the hidden layers
        """

        self._nodes = []

        """
        this initializes the nodes to be an 2d list of floats where nodes[i] contains the
        current values of the nodes ith layer of the neural network

        the values of the nodes are stored in numpy arrays this is mostly because it allows them to treated as vectors and matrices for forward and backward propagation
        """

        #initializes the input layer
        self._nodes.append(np.array([0] * number_of_input_nodes, dtype = float))

        #initializes the hidden layers
        for i in range(number_of_hidden_layers):
            lst = [0] * number_of_hidden_layer_nodes
            self._nodes.append(np.array(lst, dtype = float))

        #initializes the output layers
        self._nodes.append(np.array([0] * number_of_output_nodes, dtype = float))

        """
        here the weights of the neural network are initialized as a 3d list of floats
```

```
        with weights[i][j][k] indicating the weight that connects the jth nodes in the ith layer
        to the kth node in the (i + 1)th layer

        the values are stored in a numpy array so that they can be treated as matrices for forward and backward propagation
        """

        self.weights = []
        #adds all the weights from the input layer to the 1st hidden layer
        self.weights.append(np.array([[0] * number_of_input_nodes for _ in range(number_of_hidden_layer_nodes)], dtype = float))
        #adds all the weights between hidden layers
        for i in range(number_of_hidden_layers - 1):
            self.weights.append(np.array([[0] * number_of_hidden_layer_nodes for _ in range(number_of_hidden_layer_nodes)], dtype = float))
        #adds all the weights from the last hidden layer to the output layer
        self.weights.append(np.array([[0] * number_of_hidden_layer_nodes for _ in range(number_of_output_nodes)], dtype = float))

    #used to randomize all the weights of the neural network
    def set_random_weights(self):
        #here the program iterates through every weight and gives it a random value from -1 to 1
        for i in range(self._size - 1):
            for j in range(self._layer_sizes[i + 1]):
                for k in range(self._layer_sizes[i]):
                    self.weights[i][j][k] = random.uniform(-1, 1)

    #calculates the output for the input set
    def forward_propagate(self):
        #applies the sigmoid function to all the input layer nodes before forward propagating
        for i in range(self._layer_sizes[0]):
            self._nodes[0][i] = sigmoid(self._nodes[0][i])

        """
        starting with the first layer of the neural network the values of the next layer
        are calculated using a single matrix multiplication
        it then iterates through each layer performing the same matrix multiplication until it has reached the final layer which is the output layer
        """

        for i in range(self._size - 1):
            #calculating the values of the next layer using a matrix multiplication
            #here the @ symbol represents the numpy operator for multiplying arrays as matrices
            self._nodes[i + 1] = self.weights[i] @ self._nodes[i]
            if (i + 1 != self._size - 1):
                #applying the sigmoid function to each node in the current layer before continuing propagation
                for j in range(self._layer_sizes[i + 1]):
                    self._nodes[i + 1][j] = sigmoid(self._nodes[i + 1][j])

    #updates weights due to error in output
    def back_propagate(self, expected_values):
        #this will contain all the errors in each node
        errors = [[] for _ in range(self._size)]
```

```
        #this will contain all the errors in the output layer
        lst = []
        for i in range(self._layer_sizes[-1]):
            #the output layer must have sigmoid applied to it as I did not applying it at the end of forward propagation
            sig = sigmoid(self._nodes[-1][i])
            """
            the errors are calculated as the difference in output value and expected value
            multiplied by the derivative of the sigmoid function at that point
            """
            error = (sig - sigmoid(expected_values[i])) * transfer_derivative(sig)
            lst.append(error)

        errors[-1] = np.array(lst, dtype = float)
        """
        the error is then propagated backwards through the neural network
        calculating the error of the ith node of the previous layer can be done by iterating through each node of the current layer
        and summing the product of the weight that connects the pair of nodes and the error of the node in the current layer
        then multiplying the entire sum by the derivative of the sigmoid function of the value in the ith node

        this again is equivalent to a single matrix multiplication followed by multiplying the errors by the derivative of the sigmoid function at that value
        """
        for i in range(self._size - 2, -1, -1):
            errors[i] = errors[i + 1] @ self.weights[i]

            for j in range(self._layer_sizes[i]):
                errors[i][j] *= transfer_derivative(self._nodes[i][j])

        """
        the weights are then adjusted as follows
        the new weights that connects the kth node of the ith layer and the jth node of the (i + 1)th layer
        should be adjusted by the error of the jth node multiplied by the value of the kth node

        this will be multiplied by some constant between 0 and 1 so that stochastic gradient descent happens
        """
        for i in range(self._size - 2, -1, -1):
            for j in range(self._layer_sizes[i + 1]):
                for k in range(self._layer_sizes[i]):
                    self.weights[i][j][k] -= LEARNING_RATE * errors[i + 1][j] * self._nodes[i][k]

    #sets the input nodes to given values
    def set_input(self, lst):
        for i in range(self._layer_sizes[0]):
            self._nodes[0][i] = lst[i]

    #returns the current values fo all the nodes in the output layer
    def get_output(self):
        lst = []
        for i in range(self._layer_sizes[-1]):
            lst.append(self._nodes[-1][i])
```

```
204         return lst
205
206     #given another neural network in it will copy all the weights of that
    neural network
207     def copy(self, brain):
208         for i in range(self._size - 1):
209             for j in range(self._layer_sizes[i + 1]):
210                 for k in range(self._layer_sizes[i]):
211                     self.weights[i][j][k] = brain.weights[i][j][k]
212
213     #will write all the weights of the neural network to a text file in a
    normalized format
214     def write_to_text_file(self, path):
215         """
216         here to program will iterate through each weights and write the value
    of that weight to a given text file
217         """
218         with open(path, 'w') as f:
219             for i in range(self._size - 1):
220                 for j in range(self._layer_sizes[i +  1]):
221                     for k in range(self._layer_sizes[i]):
222                         f.write(str(self.weights[i][j][k]) + '\n')
223
224     #copies the values of weights from a text file that stores the weights of
    another neural network in the same normalized format
225     def copy_from_text_file(self, path):
226         """
227         assuming the file that is to be read from was written using the
    write_to_text_file function above then this function will read it correctly
228         it does this by iterating through all the weights in the same way that
    they are written the to file so that it reads the weights into the same place
229         """
230         with open(path, 'r') as f:
231             cur = 0
232             lines = f.read().split('\n')
233             for i in range(self._size - 1):
234                 for j in range(self._layer_sizes[i + 1]):
235                     for k in range(self._layer_sizes[i]):
236                         self.weights[i][j][k] = float(lines[cur])
237                         cur += 1
```

```
#Game_Manager

from cmath import inf
import random
import Game_GUI
import Game
import RNG
import pygame
import Menu_GUI
import Brain
import os

RESOLUTION_X = 640
RESOLUTION_Y = 640
SCALE = 40
GAMMA = 0.69
EPSILON = 0.05
FRAMES_BETWEEN_NEURAL_NETWORK_BULLETS = 110
FRAMES_BETWEEN_COPYING_NEURAL_NETWORK = 1440
FRAMES_BETWEEN_SAVING_NEURAL_NETWORK = 14400
dir = os.getcwd()

class Game_Manager:
    def __init__(self):
        """
        this is to initialize the game window
        here the root is the canvas to which all the graphics will be drawn
        """
        self._root = pygame.display.set_mode((RESOLUTION_X, RESOLUTION_Y))
        #self._root.title = ("Missile Command")

        self._game_with_neural_network = False

        self.mouse_x = 0
        self.mouse_y = 0

        self.high_scores = []
        self.delta_time = int(1000 / Game.FRAME_RATE)
        self._prev = 0
        self._prev_lives = Game.LIVES

        self.read_high_scores(dir + "\\Assets\\Highscores.txt")

    #reads the highest scores which are stored as a text file in a normalized form
    def read_high_scores(self, path):
        self.high_scores.clear()

        """
        the normalized form of the text file is simply a line by line sorted list of the highest scores
        these are split at new lines into a list of strings
        then the int function is mapped onto each string
        this returns a map which is then converted to a list
        """
        with open(path, 'r') as f:
            self.high_scores = list(map(int, f.read().split("\n")))

    #writes the highest scores in a normalized form in a text file
```

```
58      def write_high_scores(self, path):
59          #all the highest scores are written to a file line by line
60          with open(path, 'w') as f:
61              for i in range(len(self.high_scores)):
62                  if (i != len(self.high_scores) - 1):
63                      f.write(str(self.high_scores[i]) + "\n")
64                  else:
65                      f.write(str(self.high_scores[i]))
66
67      def start_game(self, seed):
68          self._game_GUI = Game_GUI.Game_GUI(self._root)
69          self._game = Game.Game()
70          self._game_GUI.game = self._game
71          self._RNG = RNG.RNG()
72          self._RNG.seed = seed
73          self._game.RNG = self._RNG
74
75          #sets it so that whenever the screen is click or a key is pressed the
    method is called in the game class
76          self._on_click = self._game.on_click
77          self._key_pressed = self._game.key_pressed
78
79          #sets it so that game_update is called every frame
80          self._update = self.game_update
81
82      def start_game_with_neural_network(self, seed):
83          self._game_with_neural_network = True
84          self._game_GUI = Game_GUI.Game_GUI(self._root)
85          self._game = Game.Game()
86          self._game_GUI.game = self._game
87          self._RNG = RNG.RNG()
88          self._RNG.seed = seed
89          self._game.RNG = self._RNG
90
91          self._on_click = self._game.on_click
92          self._key_pressed = self._game.key_pressed
93
94          #creates a neural network of size [81, 81, 81, 81]
95          self._brain = Brain.Brain(81, 81, 2, 81)
96          #self._brain.set_random_weights()
97          #copies the weights of the neural network from the model that was
    previously being trained
98          self._brain.copy_from_text_file(dir + "brains//brain.txt")
99
100         #creates another neural network of size [81, 81, 81, 81] which creates
    the expect output values
101         self._target_brain = Brain.Brain(81, 81, 2, 81)
102         #it must be the same as the current brain
103         self._target_brain.copy(self._brain)
104
105         self._update = self.game_with_neural_network_update
106         self._prev_score = 0
107         self._frames_passed = 0
108
109     def game_with_neural_network_update(self):
110         self._frames_passed += 1
111
112         if (self._frames_passed % FRAMES_BETWEEN_NEURAL_NETWORK_BULLETS == 20):
113
114             """
```

```
            turns all the pixels on the screen into a 2d list
            if a pixel has rgb value of (255, 0, 0) then it is given value 10
            otherwise it is given a value of -10
            """
            pixels = []
            for i in range(0, RESOLUTION_Y, 3):
                lst = []
                for j in range(0, RESOLUTION_X, 3):
                    colour = pygame.Surface.get_at(self._root, (i, j))
                    if (colour == (198, 42, 136 , 255)):
                        lst.append(10)
                    elif (colour == (89, 9, 149, 255) or colour == (3, 196, 161
, 255)):
                        lst.append(0)
                    else:
                        lst.append(-10)
                pixels.append(lst)

            inp = Brain.convolute(pixels, 1)

            #converts the 2d list of pixels into a 1d list
            flat = []
            for i in range(len(inp[0])):
                for j in range(len(inp)):
                    flat.append(inp[i][j])

            #getting the neural network to calculate the output
            reward = ((self._game._neural_network_score - self._prev_score -
0.5) + (self._game.lives - self._prev_lives)) * 10
            self._target_brain.set_input(flat)
            self._target_brain.forward_propagate()

            #gets the output value from the last iteration of the neural
network
            expected_output = self._brain.get_output()
            mx = -inf

            #finds the maximum Q value from the current iteration of the neural
network
            t_output = self._target_brain.get_output()
            for i in range(len(t_output)):
                if (t_output[i] > mx):
                    mx = t_output[i]

            #sets the expected output to contain the new Q value found this
iteration
            expected_output[self._prev] = mx * GAMMA + reward

            #updates the weights of the neural network accordingly
            self._brain.back_propagate(expected_output)

            #the rest of this functions is to calculate the next move of the
neural network
            self._brain.set_input(flat)
            self._brain.forward_propagate()

            out = self._brain.get_output()

            #finds the best move
            ind = 0
```

```
169             mx = out[0]
170             for i in range(len(out)):
171                 if (out[i] > mx):
172                     mx = out[i]
173                     ind = i
174 
175             #converts the output in to a coordinate
176             x = ind % 9
177             y = ind // 9
178 
179             #random chance to do something completely random
180             if (random.uniform(0, 1) < EPSILON):
181                 x = random.randrange(0, 8)
182                 y = random.randrange(0, 8)
183 
184             self._prev = y * 9 + x
185 
186             self._game.create_bullet((x + 0.5) * Game.GAME_WIDTH / 9, (y + 0.5)
    * Game.GAME_HEIGHT / 9, True)
187 
188             self._prev_score = self._game._neural_network_score
189             self._prev_lives = self._game.lives
190 
191         #if sufficient time has passed it copies the neural network that
    calculate
192         #the moves to the one that calculates the expected outputs
193         if (self._frames_passed % FRAMES_BETWEEN_COPYING_NEURAL_NETWORK == 0):
194             self._target_brain.copy(self._brain)
195 
196         #if sufficient time has passed the current
197         #weights of the brain are written to a text file
198         if (self._frames_passed % FRAMES_BETWEEN_SAVING_NEURAL_NETWORK == 0):
199             self._brain.write_to_text_file(dir + "brains//brain.txt")
200 
201         if (self._game.lives ≤ 0):
202             self._prev_lives = Game.LIVES + 1
203             self._game = Game.Game()
204             self._game_GUI.game = self._game
205             self._game.RNG = self._RNG
206             self._prev_score = 0
207 
208         self.game_update()
209 
210     def start_menu(self):
211         self._menu_GUI = Menu_GUI.Menu_GUI(self, self._root)
212         self._menu_GUI.initialize_start_menu()
213 
214         self._on_click = self._menu_GUI.on_click
215         self._key_pressed = self._menu_GUI.key_pressed
216 
217         self._update = self.menu_update
218 
219     def show_score_screen(self, score, seed):
220         self._menu_GUI = Menu_GUI.Menu_GUI(self, self._root)
221         self._menu_GUI.initialize_score_screen(score, seed)
222 
223         self._on_click = self._menu_GUI.on_click
224         self._key_pressed = self._menu_GUI.key_pressed
225 
226         self._update = self.menu_update
```

```
227
228     def game_update(self):
229         self._game.update()
230
231         self._root.fill("#000000")
232         self._game_GUI.draw()
233         """
234         if the game has ended it will show the score screen of the flag set
235
236         if it is a game against neural network then it will show the winner of
    the human and the neural network
237         if it is a solo game it will show the score that the player got before
    loosing
238         """
239         if (self._game.has_game_ended):
240             if (self._game_with_neural_network):
241                 if (self._game._neural_network_score > self._game.score):
242                     self.show_score_screen ("AI wins", self._RNG.seed)
243                 else:
244                     self.show_score_screen("Human wins", self._RNG.seed)
245                 self._game_with_neural_network = False
246             else:
247                 self.show_score_screen("Score: " + str(self._game.score),
    self._RNG.seed)
248                 self.high_scores.append(self._game._score)
249                 self.high_scores.sort()
250                 if (len(self.high_scores) > 100):
251                     self.high_scores.pop(0)
252                 self.write_high_scores(dir + "Assets\\Highscores.txt")
253
254             #self._game_with_neural_network = False
255             #self.start_game(random.randint(0, 2**32))
256             #self._update = self.game_with_neural_network_update
257             #self._prev_score = 0
258             #self._prev_lives = 0
259
260     def menu_update(self):
261         self._menu_GUI.update()
262         self._menu_GUI.draw()
263
264     """
265     this function will always be running throughout the execution of the
    program
266     whenever a new process must be started the update property of the game
    manager will be set to update the correct process
267     the on_click and key_pressed properties will also be set to be called on
    the correct object which will have a on_click and key_pressed method
268     """
269     def mainloop(self):
270         clock = pygame.time.Clock()
271
272         while (True):
273             for event in pygame.event.get():
274                 if event.type == pygame.MOUSEBUTTONDOWN:
275                     self._on_click()
276
277                 if event.type == pygame.KEYDOWN:
278                     self._key_pressed(event.key)
279
280                 if event.type == pygame.QUIT:
```

```
281                     quit()
282
283             """
284             the current process that needs to be updated is set to be
    self.update which is called every frame
285             self.update will be set to be a function so it can be called in
    mainloop as follows
286             """
287             self._update()
288
289             pygame.display.update()
290             clock.tick(Game.FRAME_RATE) #pygame function which will wait for (1
    / (frame rate)) seconds
```

# Testing

In this section I will be running some tests on my code to make sure the it is working as intended and to see if I have met the objectives that I set when I first began working on this project.

## Testing Video

https://youtu.be/oirn_KARMBk

## Menu tests

| Test Number | Test Description | Test Data | Expected Output | Actual Output | Test results |
|---|---|---|---|---|---|
| 1 | When the program is run does the home menu show? | Run the program | Home menu appears on screen | Home screen appears on screen | Test Pass |
| 2 | When the home screen is displayed is the user presented with two button, one to view their highest scores, one to play the game? | Run the program | The buttons appear reading "START" and "Highscores" | The buttons appear reading "START" and "Highscores" | Test Pass |
| 3 | When the user clicks on the "Highscores" button in the home menu does it take you to the high scores screen? | Click "Highscores" button | The Highscores screen appears | The Highscores screen appears | Test Pass |
| 4 | When the user clicks the back button in Highscores screen does it return to the Home menu? | Click back button | The Home menu appears | The home manu appears | Test Pass |
| 5 | When the user clicks the "Start" button in home screen does the game options menu appear? | Click "START" button | The game options menu appears | The game options menu appears | Test Pass |
| 6 | In the game options screen does a textbox appears and two buttons reading "START" and "START WITH AI"? | Click "START"" button in home screen | The game options screen opens and shows two Buttons reading "START" and "START WITH AI" and a textbox | The game options screen opens and shows two Buttons reading "START" and "START WITH AI" and a textbox | Test Pass |
| 7 | When the user types in the game options menu does what they type appear in the seed? | Type "1234abcd" on the game options screen | The text "1234abcd" appears in the textbox | The text "1234abcd" appears in the textbox | Test Pass |
| 8 | When the user clicks on the "START" button in the game | Click the "START" button in the | The game screen shows | The game screen shows | Test Pass |

| | | | | | |
|---|---|---|---|---|---|
| | options screen does the game screen show? | game options screen | | | |
| 9 | When the user clicks on the "START WITH AI" button in the game options screen does the game with AI screen show? | Click on the "START WITH AI" button in the game options screen | The game with AI screen shows | The game with AI screen shows | Test Pass **Objective 1 met** |

Game Tests

| | | | | | |
|---|---|---|---|---|---|
| 10 | When the game screen has first loaded do missiles start to come down from the top of the screen? | Click on the "START" button in the game options screen | The game screen shows and missiles start to move to the bottom of the screen | The game screen shows and missiles start to move to the bottom of the screen | Test Pass |
| 11 | After the game screen has been running for a while does the rate of missiles spawns increase? | Click on the "START" button in the game options menu and destroy missiles for 30 seconds | The time in between missile spawns is much lower | The time in between missile spawns is much lower | Test Pass **Objective 3 met** |
| 12 | On the game screen when the user clicks on the screen does a bullet move towards that position of the screen? | On the game screen click somewhere on the screen | A bullet should travel to that position | A bullet travels to the position the user clicked | Test Pass |
| 13 | On the game screen when bullet reaches its target does it explode? | On the games screen click somewhere and wait until the bullet reaches that position | The bullet should enter its explosion animation | The bullet enters its explosion animation | Test Pass |
| 14 | On the game screen if a bullet explodes next to a missile does that missile get destroyed? | Click in the path of the missile in the game screen and wait for it to reach that position | The bullet should enter its explosion animation and the missile should be destroyed | The bullet enters its explosion animation and the missile is removed from the game | Test Pass |
| 15 | On the game screen can the score box be seen? | On game screen look in the top left for score box | Score box is there | Score box is there | Test Pass |

| | | | | | |
|---|---|---|---|---|---|
| 16 | Each time the player destroys a missile does the score in the score box increment by 1? | Shoot a bullet to destroy one of the oncoming missiles | The number in the score box should increase by one | The number in the score box increases by one | Test Pass **Objective 4 met** |
| 17 | On the game screen can the lives remaining box be seen? | On the game screen look at the top left for lives remaining box | Lives remaining box is there displaying "5" | Lives remaining box is there displaying "5" | Test Pass |
| 18 | When a missile reaches the ground does the number in the lives remaining box decrease by 1? | In the game screen wait until a missile has reached the ground | The number of lives remaining should decrease by 1 | The number of lives remaining decreases by 1 | Test Pass |
| 19 | When the number of lives remaining reaches 0 does the score screen show? | In the game screen wait until the lives remaining box reaches 0 | The score screen should show | The score screen shows | Test Pass |
| 20 | On the score screen does the seed and score show of the previous game ? | Play the game until the losing it. | The seed and score can be seen | The seed and score can be seen in a box that reads "SEED: (the seed)" "Score: (score)" | Test Pass |
| 21 | On the score screen does there exist a button that says menu? | Play the game until losing it | There should be a button on the screen that reads "MENU" | There is a button on the screen that reads "MENU" | Test Pass |
| 22 | When the button the score screen that reads "MENU" does this then take the user to the menu screen? | Click the menu button on the score screen | The Home menu should appear | The Home menu appears | Test Pass **Objective 5 met** |

Seeding tests

| | | | | | |
|---|---|---|---|---|---|
| 23 | If a user enters the same number in a seed box multiple times does the same game play out? | Using a random number generator, generate and random number from 1 to 2^32 and use that seed multiple times taking note of the | Each time the same seed is entered missile that spawn from the sky should spawn in the exact same places going in the exact same direction at the same times. | Every time the same seed is entered the missiles spawn at the same place in the same direction at the same times | Test Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | path and spawn time of each missile | | | |
| 24 | If the user does not enter anything into the seed box does a random game begin? | In the game options menu press "START" and die, taking note of the paths and timing of the missiles several times | Each time you repeat the process the missile should spawn in completely different locations at completely different times | Every iteration of the process the missiles always spawn in different places at different times | Test Pass |
| 25 | If the user enters something that is not a number in the seed box does a random game begin? | In the game options menu enter the string "//z" and press "START" and die | Each time you repeat the process the seed in the SEED box of the score screen should be random | Every time the seed displayed is completely random. | Test Pass **Objective 2 met** |

Highscores tests

| | | | | | |
|---|---|---|---|---|---|
| 26 | When the user gets a new high score does it appear on the high score screen? | Play the game and get a sufficiently large score that places within the top 12 highest scores | When viewing the Highscores screen the new score should also be in the list of high scores | On the Highscores screen the new score is in the list of high scores | Test Pass |
| 27 | When the game is closed and re-run do the highest scores remain? | Play the game and get a sufficiently large score that places within the top 12 highest scores and then close and re-run the program | When viewing the Highscores screen the new score should be in the list of high scores | On the Highscores screen the new score is in the list of scores | Test Pass |
| 28 | When the user does not get a new high scores does the new score appear on the high score screen? | Play the game and get a score that does not place within the top 12 highest scores | When viewing the Highscores screen the new score should not be in the list of high scores | When viewing the Highscores screen the new score cannot be seen in the list of high scores | Test Pass **Objective 6 met** |

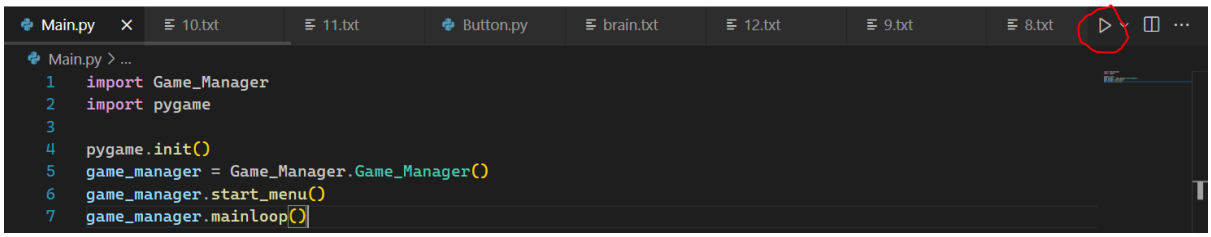
## Neural network tests

| | | | | | |
|---|---|---|---|---|---|
| 29 | When the user clicks on the "START WITH AI" does the game begin with a neural network | On the game options menu click on the button that reads: "START WITH AI" and wait | The game should begin and the blue bullets should appear from the bottom of the screen | The game begins and blue bullets appear on the bottom of the screen. | Test Pass |
| 30 | Does the neural network shoot bullets on a fixed time interval | Click "START WITH AI" button on game options menu and observe | The game should begin and the blue bullets from the neural network should come out in fixed time intervals | The game begins and the blue bullets spawn in fixed time intervals | Test Pass **Objective 8 met** |
| 31 | When the game is closed and re-opened are the weights of the neural network stored | Close the game and re-run it and then navigate to play a game with neural network. | The neural network should get a much higher score that a neural network that uses random weightings | The neural network gets a score of 52, which is much better than random which gets a score of 1. | Test Pass **Objective 9 met** |
| 32 | Can the score of the neural network and player be seen during a game with neural network | Click on the "START WITH AI" button in the game options menu and look in the top left of the screen | There should be two boxes coloured blue and purple which should contain the number 0 | There are two boxes in the top left of the screen which are blue and purple which both read 0 initially | Test Pass |
| 33 | When a neural network bullets destroys a missile does the score increment by 1 | Click the "START WITH AI" button and wait for a neural network bullet to destroy a missile | The score counter to increment by one for each destroyed bullet by the neural network | Each time a missile is destroyed by neural network bullets the score increments by 1 | Test Pass |
| 34 | When a player bullet destroys a missile does the score increment by 1 | Click the "START WITH AI" button and click in the path of a bullet to destroy it | The green box in the top left should increment its number by 1 | Each time a player bullet destroys a missile the player score in the green box increase by 1 | Test Pass **Objective 10 met** |

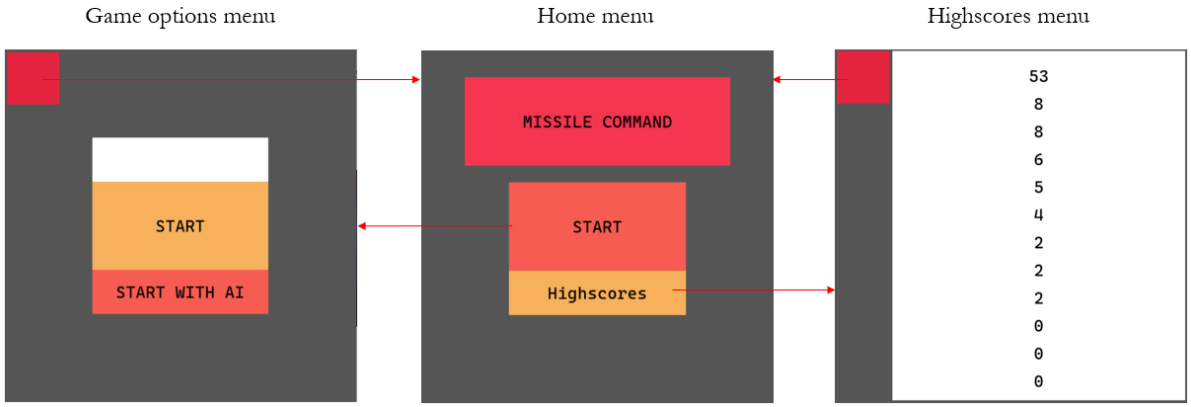
| | | | | | |
|---|---|---|---|---|---|
| | | with a player bullet | | | |
| 35 | When a game with neural network ends does the person who won display? | Click on the "START WITH AI" button and do nothing and wait until the neural network wins | The score screen should show showing the seed and that the AI won. | The score screen shows up and reads "SEED: (the seed)", "AI wins" and a button that reads "MENU" | Test Pass |
| | | Click on the "START WITH AI" button and play get a higher score than the AI. | The score screen should show showing the seed and that the human won. | The score screen shows up and reads "SEED: (the seed)", "Human wins" and a button that reads "MENU" | Test Pass |
| 36 | Does the neural network get better after training for a long time? | Reset the weights of the neural network to be random and then run it training for a few hours. | The scores it should be getting should increase a lot from when the weights were random to after it was trained for a long time. | The neural network gets a lot better | Test Pass |

## Test evidence

### Tests 1-6

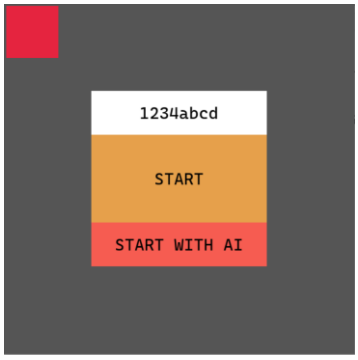



When I run the program the following window shows which is the home menu. The following logic happens when I press the buttons. Below if there is an arrow from a button to a screen it means that when I pressed that button it took me to the screen the arrow points to.



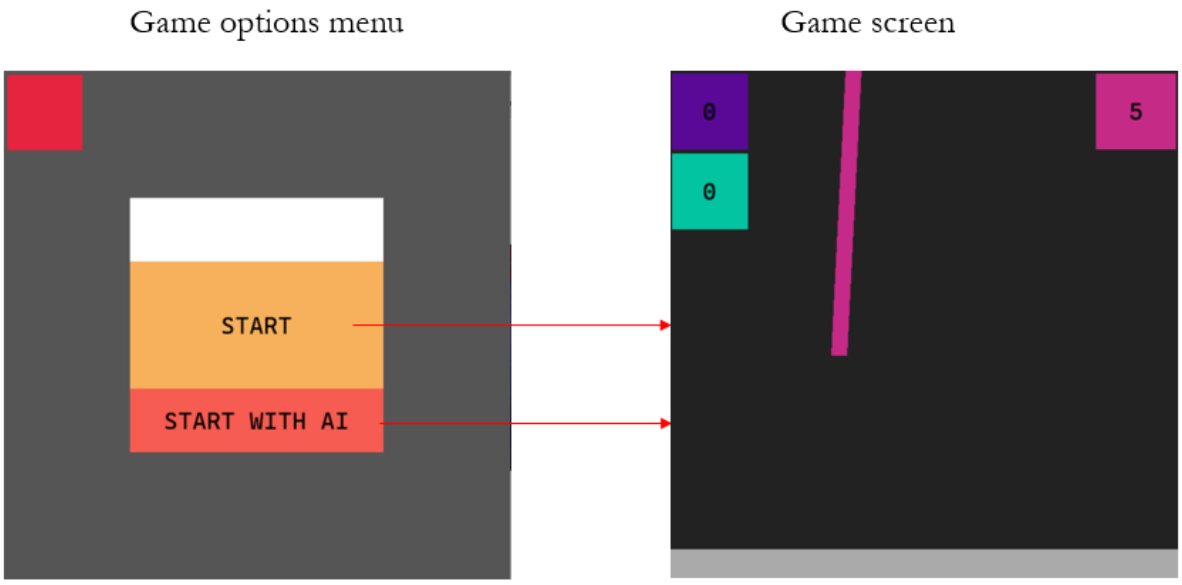

Test 7

After typing "1234abcd" into the game option menu the window looks as follows which is as expected



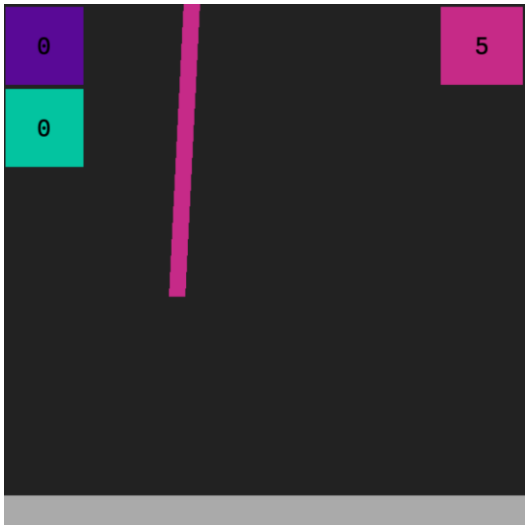

Test 8-10

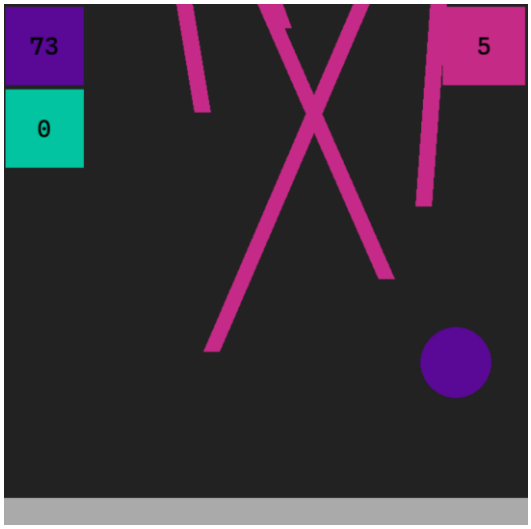
the following relationship happens between the two screen where the arrows indicate the same as above.




Test 11

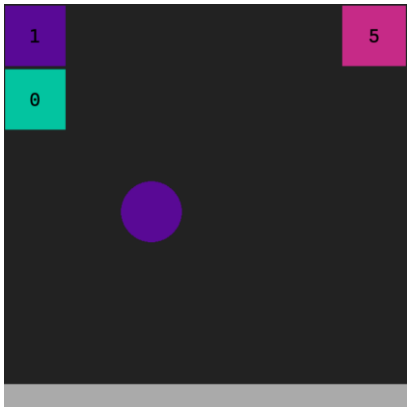
I played the game until I got a score of 70 and the missiles and the density of missile change as shows from the difference in the first image and second image



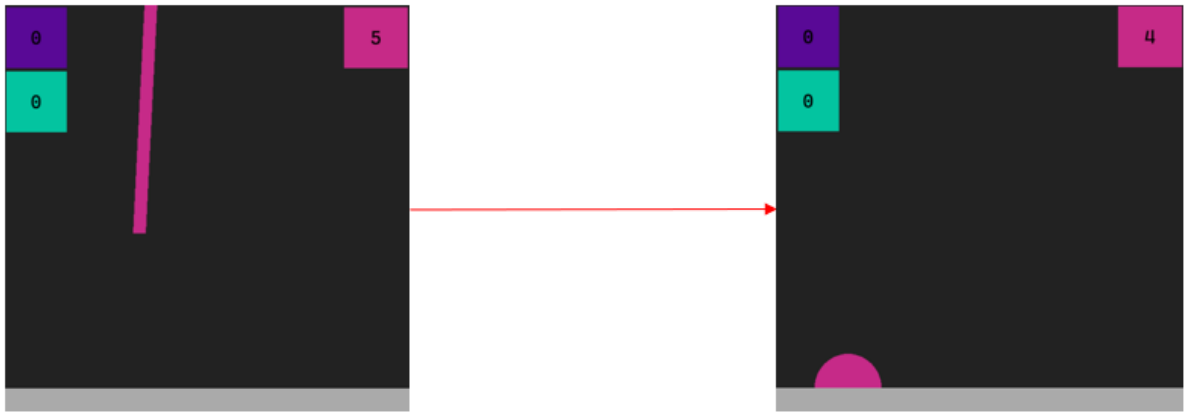

Test 12 – 16

I clicked in the middle of the screen. A bullet then travelled there and exploded next to a missile as shown. The missile was then destroyed. The score box can be seen and was incremented by 1 after the missile was destroyed.



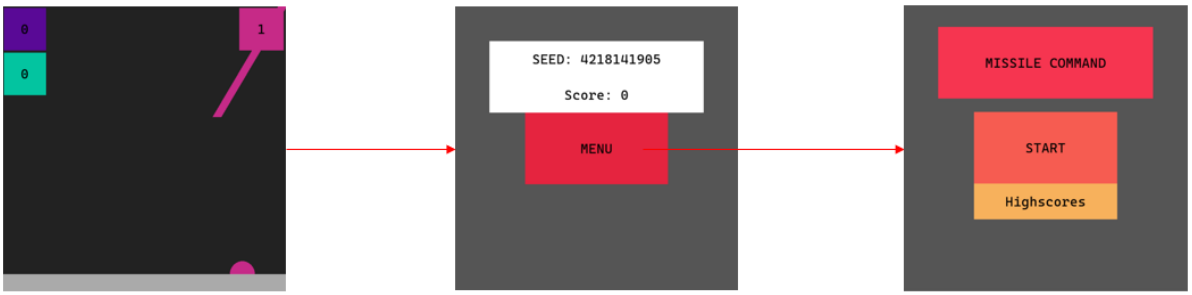

Test 17-18

in the game screen you can clearly see that the red box in the top right contains the number of lives, which is 5 initially. After the player is hit by an oncoming missile the score decrements by 1 and for each missile after it decrement by 1.




Test 19-22

When the user has 1 life left and a missile hits the ground the player is take to the score screen. Here the seed and score can be seen. There is also a button that say menu on it which when clicked takes the user to the home screen.

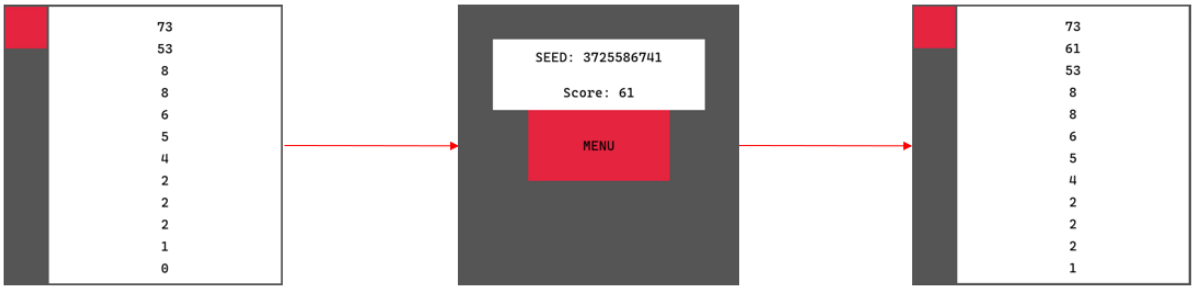



Test 23-25

Video timestamp: 4:59

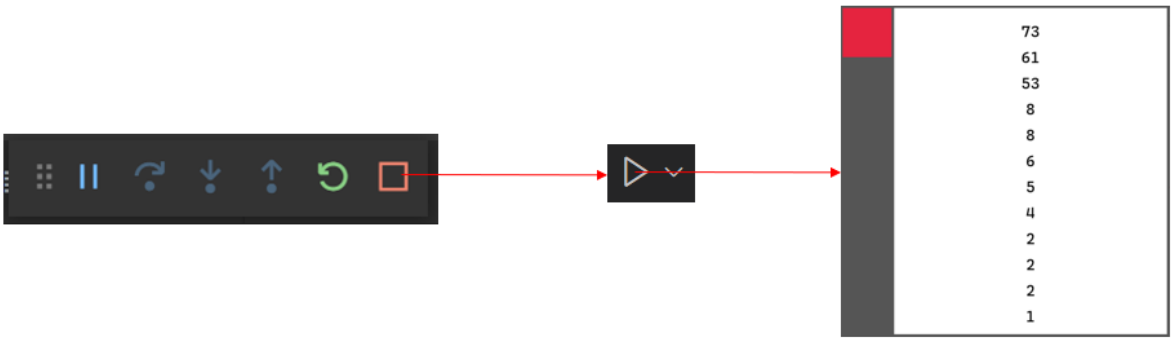
Test 26

When clicking on the high score screen on the home menu it look like as shown below. Then after playing the game and getting a score of 61 and returning to score screen the score of 61 can be seen in the high scores screen.




Test 27

After closing the game in vs code and re-running it. After navigating to the high scores screen the screen looks as shown below which is the same as it was before. Thus is passes the test.

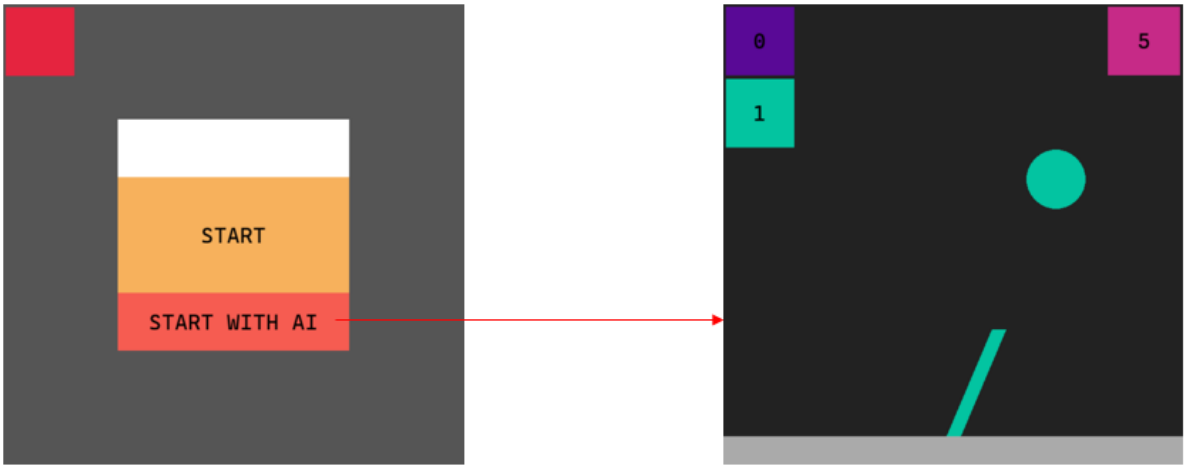



Test 28

Before playing the game the high scores screen looks as follows. After playing the game and getting a score of 0 and navigating back to the high scores menu the screen looks as shown which is identical to as before.

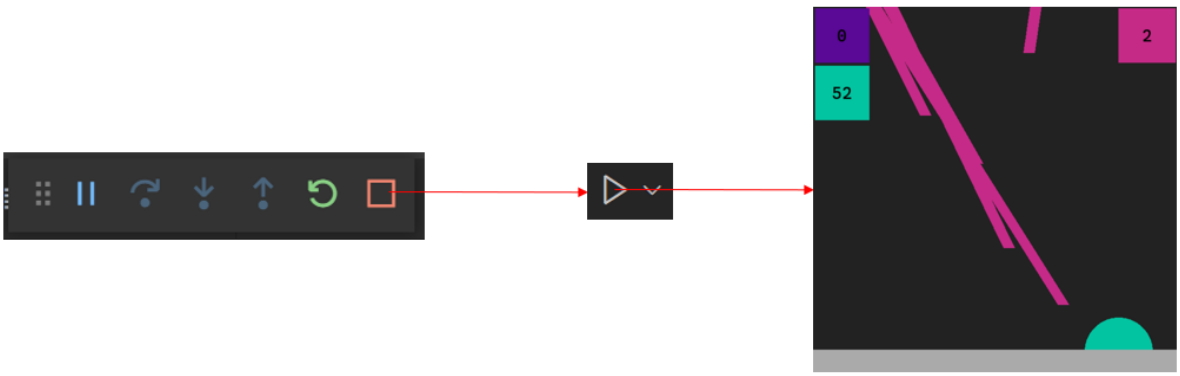

Test 29-30

After clicking on the "START WITH AI" button in the game options menu I was taken to the game screen and some blue bullets appeared from the bottom in a fixed interval.



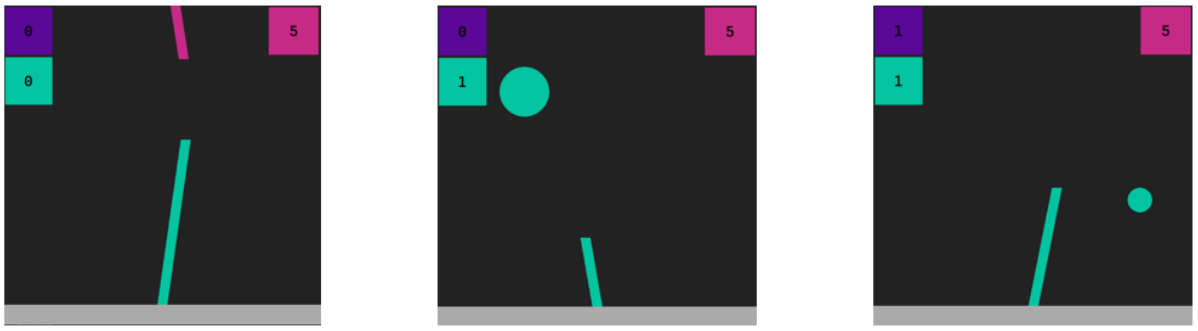

Test 31

After closing the game and re-running. After navigating to a game with neural network and letting it play for a while it manages to get a score of 52 before running out of the 5 lives. After changing the neural network to have random weightings it gets a score of 1. Showing that the neural network is indeed being stored.



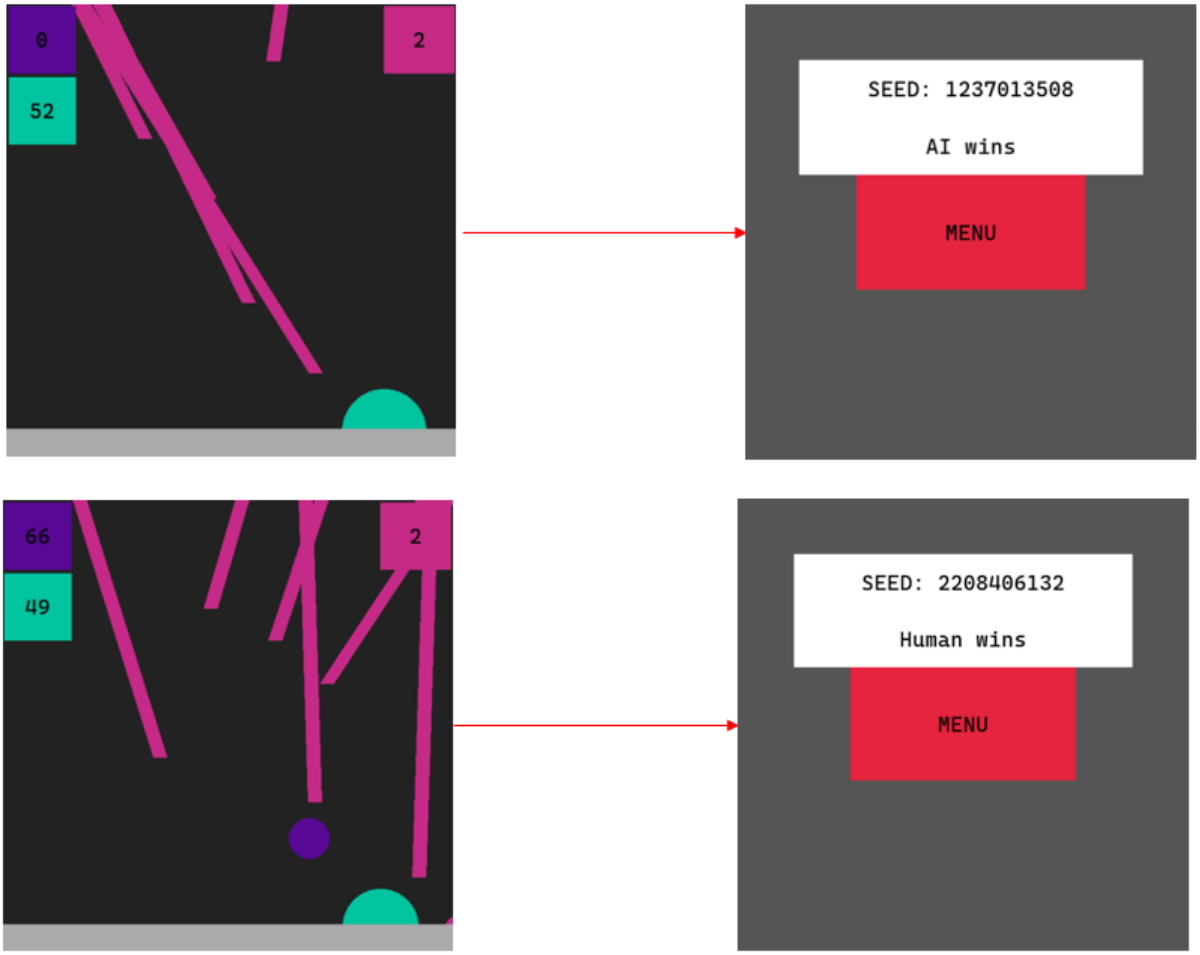

Test 32-34

After clicking the "START WITH AI" button in the game options menu. There clearly is are two boxes in the top left containing the players score and the neural network score. After a neural network bullet destroyed a missile the score in the blue box incremented by 1. After I destroyed a missile with a green bullet the number in the green box incremented by 1




Test 35

After starting a game with neural network and then doing nothing the neural network got a score of 41 and the player 0. After the last life of the game was lost I was taken to a screen which read "AI wins" as expected. After playing a game against the AI and getting a score of 58-40 I was taken to a screen reading "Human wins" as expected.




Test 36
Video timestamp: 10:53

# Evaluation

Overall I feel that my project has been a success. Throughout the project I feel that my programming skills have improved drastically and I think my understanding for neural networks has deepened quite a lot. The final system has met every requirement that my client gave to me.

## Meeting objectives

| No. | Objective | How was it met? | To what extent was it met | In what ways could it have been improved |
|---|---|---|---|---|
| 1 | When the game is started the user should be presented with the options of viewing their high scores, playing the game solo or playing the game against a neural network. | Using the pygame module I created a window on which I could draw boxes onto. Using the built in click events and mouse position detection in pygame I could create a separate button class which when clicked brings up the respective screen. | Entirely | No improvements |
| 2 | The game must have a seeding system that uses a single integer to determine the path of missiles and the time they will spawn so that the game appears random but using the same seed will always give the same time and location of missile spawns. | By always calculating the time of spawning and path of missiles with the same random number generator. Using a linear congruential generator to calculate pseudorandom numbers, given the same starting inputs the outcome will always be the same. | Entirely | No improvements |
| 3 | The game must spawn missiles for the user to shoot and as time passes during a game the number of missiles that spawn per second should increase. | In the update function I made a variable which stores the number of frames which must have passed before the next missile should spawn. After a missile spawns I added a value to that variable. The value which is added decreases with time thus making the number of missile that spawn per second increase. | Entirely | Could have perhaps used a more complicated function to make the difficulty increase less sharp. |
| 4 | The user must be able to spawn bullets by clicking on the screen and have a bullet travel to the location clicked. When a bullet explodes near a missile the missile should be destroyed and the players score should be incremented by one. | By using pygame click detection I set it so that whenever the user clicked somewhere on the screen pygame module would call a method in the game class. This method would then add a new bullet, targeted at the cursor position to the list of bullets. Each frame the program checks if each bullet has reached the position the | Entirely | No improvements |

| | | | | |
|---|---|---|---|---|
| | | user clicked when it was spawned to check if the bullet should explode. After explosion the program iterates through the list of missile to check if any of them are in range. Then for each missile that should be destroyed it is removed and 1 is added to the score. | | |
| 5 | The player must have a given number of lives. Every time a missile reaches the ground they should lose one life, if they run out of live the player should lose the game and show a screen containing the score they achieved, the seed used and a button to return to the starting screen. | Each frame the game class will check if the missile has reach the ground and if so it will be start its explosion animation. The number of lives that the player has remaining is stored as an integer and every time the ground is hit by a missile the number of lives remaining will be decremented by 1. Once the number of lives has reached 0 the score screen will show. Which shows the score and seed using labels. | Entirely | No improvements |
| 6 | The program must be able to store the highest scores achieved by the player when playing solo. The player should also be able to view them after navigating to it from the start menu. | The highest scores achieved by the player are stored in text file. The scores are stored in order of highest to lowest. Each time the player sets a new score the value is added to the list and the list is then sorted and the lowest value removed. | Entirely | Could have used custom file format so that the storage file is lower. |
| 7 | The program must also support the option to play against a neural network. | By dividing the screen up into different regions and using the rgb values of those regions as inputs to the neural network. The outputs of the neural network are the same regions that are used as inputs, which correspond to places the neural network can shoot bullets at. Then using the neural network to approximate a Q function for the game it learnt to play the quite well. | Entirely | The neural network could have been optimised a bit. |
| 8 | The program must have the neural network shoot bullets at a point on the screen in fixed time intervals. | By storing the number of frames that have passed in an integer and then each time 110 divides the number of frames that have passed the neural network is called to fire a bullet. | Entirely | No improvements |

| | | | | |
|---|---|---|---|---|
| 9 | The neural network weights must be stored between uses of the program so that they can be used the next time the program is run. | By iterating through each weight in the neural network and writing them to a text file line by line. By reading the file in the same order that it writes to a file it is possible to decipher which weights correspond to which number in the text file thus allowing the weights of the neural network to be stored and read. | Entirely | Could have used custom file type so that the storage size is lower |
| 10 | The game must show the current score of the player and of the neural network during the games execution. | By storing the number of missiles that the player and neural network has destroyed in an integer and increasing it by one every time someone destroys a missile. Then the scores are shown by using a label and setting the text inside the label to be the current scores of the player and neural network and drawing it onto the screen. | Entirely | No improvements |

## End-User Feedback

After I gave the program to my client and letting him using it for 2 weeks. I sent him the following feedback form to get his opinion on the final result of the project.

**Question**: Do you believe that the final product meets all the requirements set out?

**Response**: Definitely! the product has a functional version of Missile Command and the AI you can play against is quite difficult to beat but it does not feel like it is unfair. All the other features that I asked for have been implemented.

**Question**: Is there anything that you think could be improved upon?

**Response**: The graphics could perhaps be more detailed, at the moment it looks a bit too simple with not much detail. The missile spawns could be sophisticated, at the moment they spawn at somewhat predictable times in no groupings. Also I think that the game would benefit from having the option to start playing at a higher difficulty. At the moment having to start playing from the easiest difficulty is quite boring for people who have some level of experience.

**Question**: Is there anything that you may have wanted added to this project?

**Response**: Maybe a way to show a replay of my previous games. This would be helpful for showing my best games to my friends. I also think that the AI should have selectable different difficulties so that players of different skill levels can either compete with or be able to get more of a challenge against it.

**Question**: Is there anything else you want to mention about this project?

**Response**: I have enjoyed the whole experience of helping to develop this project.

## Analysis of feedback

My client overall seemed very happy with the final product, saying that I have met all the requirements that he set out when we first began this project. He stated that the Neural network that you can play against was quite a challenge to beat but it still felt fair. I think that this is because I made it so that the neural network shoots at the missiles in relatively slow fixed time intervals which makes it behave more similarly to a human as opposed to trying to get it to be as good as possible which would mean letting it shoot every frame.

He said that the graphics looked a bit too simple which I agree with. The entire game was constructed using boxes filled with single colours and text. While this does make it more similar to the original game in the modern day there is much better hardware so it is now possible to use better graphics. Unfortunately under the time limits I do not think that it would have really been feasible to me to make the graphics very sophisticated, if I had spent more time trying to make the game look much better, then I could not have spent as much time creating some of the more important parts of the project. But I do think it would be a nice addition to give the game some better looking graphics.

Craig Chittick's idea about potentially adding a replay system to watch some of his previous games is an interesting idea and definitely feasible to implement. If I were to spend some more time working on the project I would spend some time implementing this feature.

## Suggested improvements

If I were to keep working on this project then here are some of the main things I would change:

Make a more sophisticated way to spawn the missiles. My client mentioned this and I agree, at the moment the way that the missiles spawn is very predictable. The missiles spawn in very close to fixed time intervals, which is quite predictable and boring. In the original version of the game the missile spawn in groups which gives the game a bit more character and I think that the version of the game that I have made would benefit from a system like this.

My client also mentioned that the game should have the option to start at a more difficult point. I think that this is a good idea as it would be ideal for the more skilled players so that they can start their games on games on higher difficulty to give themselves more of a challenge straight away. Similarly he mentioned that the game should feature neural networks players of different difficulties, which again I think is a good idea for similar reasons.