2019/20

# Computer Science - NEA

## Obstruction Game

Holly Church (189605)
GODALMING COLLEGE - 64395

# Contents

Research ..... 3
Overview ..... 3
Background ..... 3
How to play ..... 4
Third Party ..... 5
Other versions of the game ..... 6
Algorithms ..... 6
Minimax ..... 6
Combinatorial Game Theory ..... 7
Analysis ..... 7
Data flow diagram ..... 7
IPSO diagrams ..... 8
Flowchart ..... 9
Decision table ..... 10
Objectives ..... 11
Dialogue ..... 11
Requirements ..... 12
Design ..... 13
Files ..... 13
Algorithms ..... 13
Drawing the grid out ..... 13
switching player ..... 13
Click on the grid ..... 14
Game won ..... 14
AI Algorithm ..... 14
Front end design ..... 15
Main menu ..... 15
Board size ..... 17
Loading up a previous game ..... 18
Opening grid ..... 19
Mid Game ..... 20
Game won ..... 21
Testing strategy ..... 22

Evaluation ..... 27
Meeting of requirements ..... 27
Improvements ..... 29
Independent feedback ..... 29
Technical solution ..... 30

Figure 1 http://www.papg.com/images/Obstruction.gif example of gameplay ..... 4
Figure 2 http://www.papg.com/obstruction.html example of a 3x3 grid ..... 4
Figure 3 http://www.lkozma.net/images/p1.PNG example of how to win on an odd x odd grid ..... 4
Figure 4 http://www.papg.com/obstruction.html example of gameplay on a 2x3 grid ..... 5
Figure 5 http://www.papg.com/obstruction.html example of gameplay on a 2x2 grid ..... 5
Figure 6 http://www.papg.com/obstruction.html example of gameplay on a 2x4 grid ..... 5
Figure 7 https://www.math.ucla.edu/~tom/Games/dawson.html this is the game of Dawson's chess which is similar to obstruction ..... 6
Figure 8 http://www.papg.com/obstruction.html This shows how the computer has placed both moves as player X rather than one to X and one to O ..... 6
Figure 9 http://www.papg.com/obstruction.html this is the game play from the pen and paper games website ..... 6
Figure 10 https://www.math.ucla.edu/~tom/Games/dawson.html this shows the wining message appearing before the pieces have been placed ..... 6
Figure 11 Data flow diagram ..... 7
Figure 12 IPSO Diagram for the player ..... 8
Figure 13 IPSO Diagram for the AI ..... 8
Figure 14 IPSO Diagram for the front-end design ..... 9
Figure 15 Flow Chart overview ..... 9
Figure 16 Decision table for game progression ..... 10

# Research

## Overview

Obstruction is a pen and paper game where players take turns in placing a symbol onto a square or rectangular grid. You pick a square to mark and then the surrounding 8 squares are unable to be marked, you win by making your opponent unable to move .It is played on a grid of any size by any size, however board sizes of 6x6 and up are typically used as it leads to longer, more interesting gameplay. The game is traditionally played with two people on a piece of paper but there are a couple of online versions of the game, however there are glitches and problems within them which impacts the gameplay.

I aim to create a program where a user can play a game of obstruction on their desired size of board and can have the option to play against either a real person, they are with or against an AI.

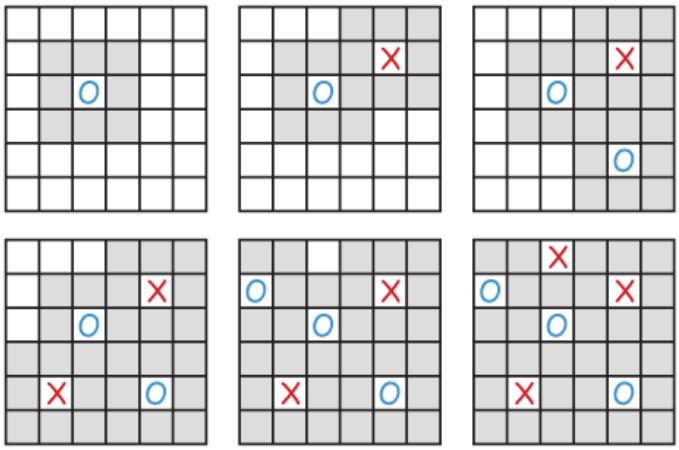
This game is a mathematical game and therefore needs an understanding of the mathematical theorems to find solutions to the game and how to predict what the best move will be.

## Background

This game was originally invented and analysed by the Romanian mathematician and theoretical computer scientist László Kozma who used multiple theorems from different games to thoroughly analyse and understand the winning strategies to the game. The game take influences from games such as Go, Dawson's chess, played on a 1 x n board, and a game called Regio where you take into account the four direct neighbours instead of the surrounding eight. This compilation of multiple other games means that there are many theorems behind this game to try to get the most accurate representation of it.

## How to play

To play the game you take your grid and the first player can mark their chosen space with their symbol (e.g. X or O), the surrounding eight squares (or less if it's on an edge or corner) are then shaded out and nobody can place their symbol there. The next player can then place their symbol in any unshaded square and the respective squares around that are also shaded out. This carries on until all the squares are either shaded or contain a symbol, the last player to successfully place a symbol down wins.



Here X wins as O cannot place their symbol anywhere.

*Figure 1 http://www.papg.com/images/Obstruction.gif example of gameplay*

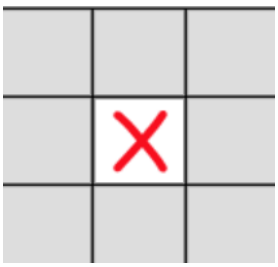


*Figure 2 http://www.papg.com/obstruction.html example of a 3x3 grid*

The board size dramatically changes how difficult the game is to win. For example on a 3x3 grid the first player can always win by placing their symbol in the centre. This strategy can be expanded to any size board if it is odd x odd as the first player can place their symbol in the middle of the board and then just mirror the second players move by putting their symbol in the space opposite with respect to the centre of the board, until the second player runs out of moves. Similar problems occur with other boards of smaller sizes as well, for example on a 2x2 grid the first player will always win as no matter where they put it the entire grid will then be blocked off. On a Board of 2x3 the first player can win by placing their symbol in one of the centre two squares, however on a board of 2x4 the second player will win. With boards of bigger sizes the winning strategies are not as simple and can depend on whether you go first or second and where the other player places their symbols.

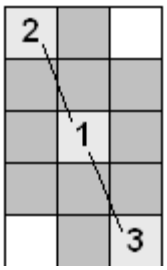



*Figure 3 http://www.lkozma.net/images/p1.PNG example of how to win on an odd x odd grid*

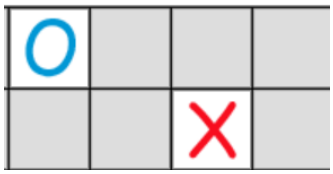


*Figure 6 http://www.papg.com/obstruction.html example of gameplay on a 2x4 grid*

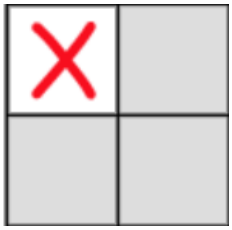


*Figure 5 http://www.papg.com/obstruction.html example of gameplay on a 2x2 grid*

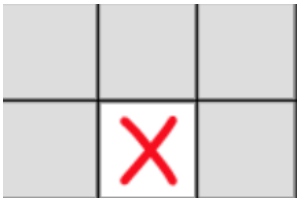


*Figure 4 http://www.papg.com/obstruction.html example of gameplay on a 2x3 grid*

## Third Party

Below is an interview that I held with the third party user, this helps to gain a better understanding of the objectives and requirements that need to be included within the coded program.

**Hello, could you please say who you are and tell us a bit about yourself**

Hello, my name is Sophie Spencer, I am a 6th form student. I am 17 years old and I really enjoy playing board game type games, from games like cluedo and monopoly to noughts and crosses and dots and boxes.

**What is it that you would like me to do for you?**

I would like you to create a game of the pen and paper game called obstruction for me where I can play against an artificial intelligence.

**Why would you like me to create this for you?**

I really enjoy playing pen and paper type games but unfortunately I do not always have someone to play with as I am either alone or the person I'm with would prefer to play a video game.

**How would you like the game to look?**

I would like for the design to be simple to understand and use but I also want it to be graphical so that I can see where the symbols have been placed and where else is available to place them. I want the interface to be very interactive with me as the user.

**How challenging would you like the game to be?**

I would like the AI to be able to determine the best possible move and place it there. I would like to be challenged by it and for it to know all of the winning strategies so it is as if the AI is just another person that I am playing against.

**Is there anything else that you would like to be included within the game?**

I would like for there to be some sort of scoring system so I can keep track of how well I am doing. I would also like to be able to reload a saved game and to continue where I left off.

**Thank you for your time.**

Thank you.

## Other versions of the game

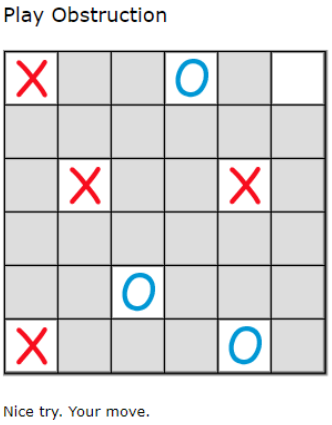



*Figure 9 http://www.papg.com/obstruction.html this is the game play from the pen and paper games website*

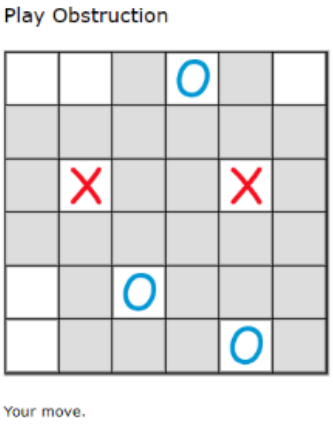



*Figure 8 http://www.papg.com/obstruction.html This shows how the computer has placed both moves as player X rather than one to X and one to O*

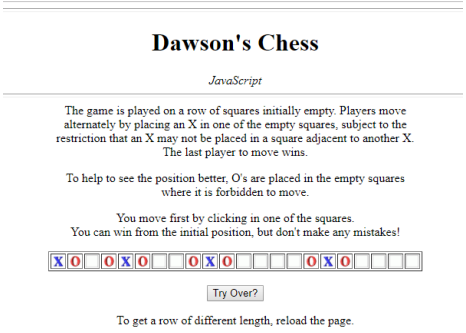



*Figure 7 https://www.math.ucla.edu/~tom/Games/dawson.html this is the game of Dawson's chess which is similar to obstruction*

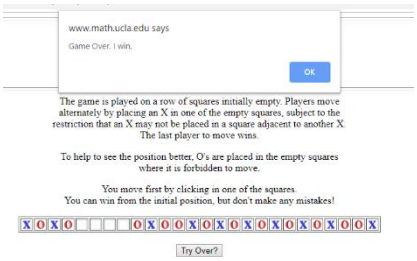



*Figure 10 https://www.math.ucla.edu/~tom/Games/dawson.html this shows the wining message appearing before the pieces have been placed*

Online there is only one full version of the game as described found, this found at http://www.papg.com/show?2XMX however this game is not the best as sometimes when clicking the wrong symbol is placed down which appears to confuse the AI as to who has won and lost the game. The game is also played by the website loading up a new webpage every time you make a move which appears to be inefficient and can impact the gameplay. Another similar version of the game is found at https://www.math.ucla.edu/~tom/Games/dawson.html and is called Dawson's chess, this is similar to obstruction however it only ever has one row. This game doesn't appear to malfunction in any way however it can be confusing as both players are represented by the X and the O is used to show the boxes that have been blocked off. This game however is not as visually attractive as the obstruction game on papg.com. Also the fact that the winners message appears before the final moves have appeared on the board means that the game doesn't run as smooth and simple as it tries to.

## Algorithms

### Minimax

The minimax theorem is an algorithm used in game theory and decision making to find the best move possible. It does this by analysing all of the possible moves and then from each of

them is counts up the number of winning outcomes Vs losing outcomes to select the move that will give the best chance of winning. In Minimax each of the two players is either assigned the maximiser, who tries to get the highest score, or the minimiser who tries to get the lowest score possible. This tries to minimise the loss while maximising gain. It is used to look through the possible moves to work out the move option with the best outcomes with the other player moving optimally.

### Combinatorial Game Theory

Combinatorial game theory is the theory of games where two players take turns to take a position until a winning condition is met. It is used when there is perfect information, the state of the board and all moves are known to both players. It is used to help understand what the optimal gameplay will result in. It uses the known theories form other games to help to optimise the winning theory to other games.

## Analysis

### Data flow diagram

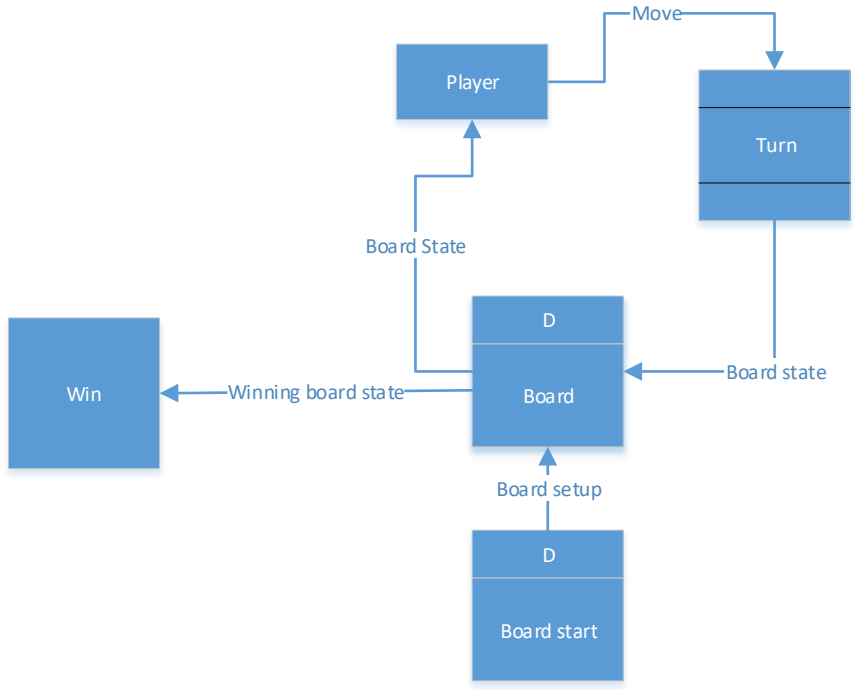



*Figure 11 Data flow diagram*

## IPSO diagrams

For player

| Input | Process |
|---|---|
| select move | Check box is available<br>Make move<br>Make surrounding boxes unavailable<br>Update board |
| Store | Output |
| Board state<br>Player turn | Display board |

*Figure 12 IPSO Diagram for the player*

For AI

| Input | Process |
|---|---|
| Player move | Get board state<br>Figure out best move<br>Place move<br>Make surrounding boxes unavailable<br>Update board |
| store | output |
| Board state<br>Player turn | Move |

*Figure 13 IPSO Diagram for the AI*

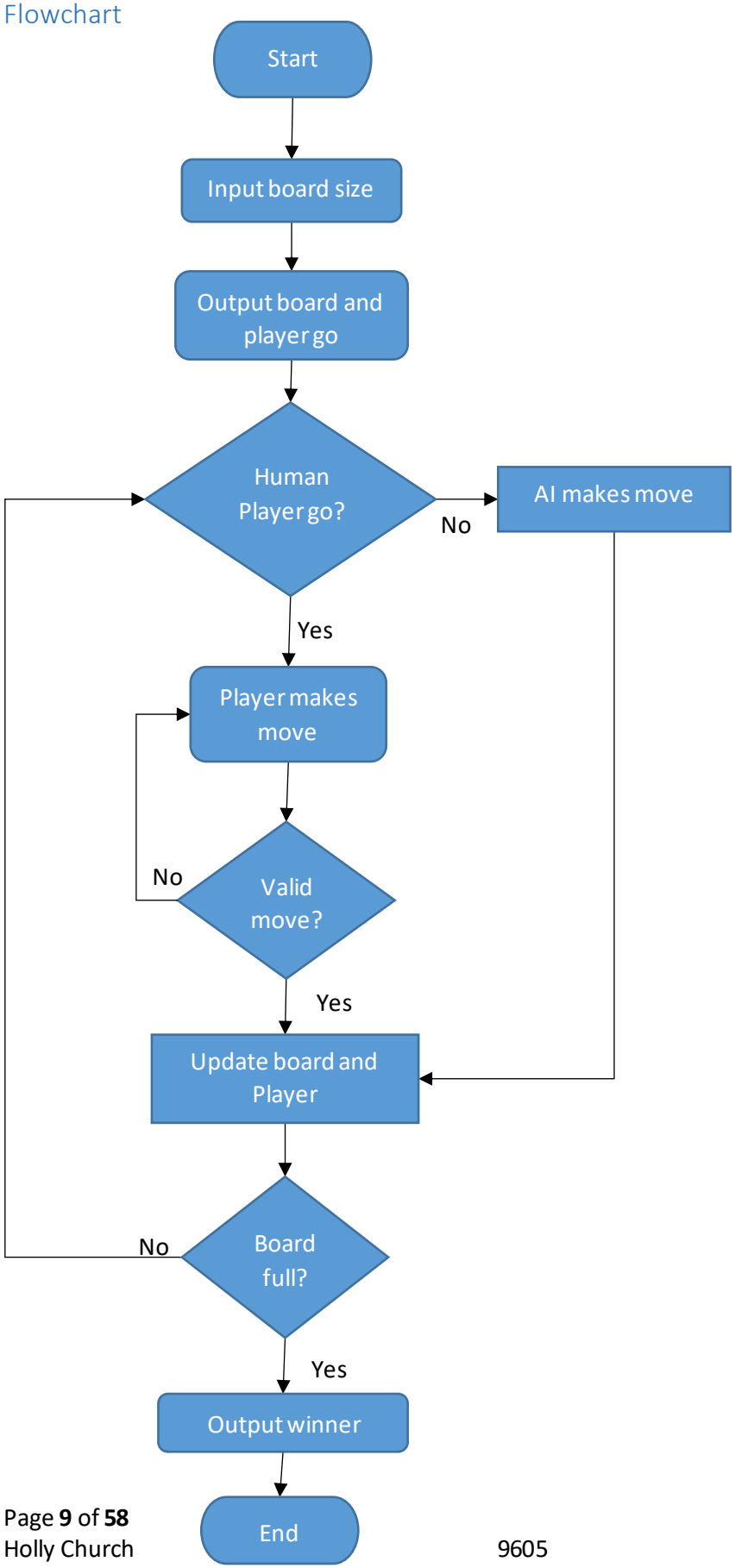
Front end

| Input | Process |
|---|---|
| Box click<br>Board size | Switch between players<br>Change board size<br>Who goes first |
| Store | Output |
| Board size<br>Board state | Board size<br>First player |

| | |
|---|---|
| Player go | |

*Figure 14 IPSO Diagram for the front-end design*

## Flowchart




Overview flowchart

*Figure 15 Flow Chart overview*

## Decision table

| **Conditions** | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 |
|---|---|---|---|---|---|---|---|---|
| Valid move | N | Y | N | Y | N | Y | N | Y |
| Full board | N | N | Y | Y | N | N | Y | Y |
| Player move | N | N | N | N | Y | Y | Y | Y |
| AI turn | Y | Y | Y | Y | N | N | N | N |
| **Actions** | | | | | | | | |
| Player turn | | X | | | | | | |
| AI turn | | | | | | X | | |
| Invalid | X | | | | X | | | |
| Winner player | | | X | X | | | | |
| Winner AI | | | | | | | X | X |

*Figure 16 Decision table for game progression*

# Objectives

## Dialogue

The board has to be created in a clear way so that it is easy for the player to understand and use, so that the player can always clearly understand what is happening in the game and how it is progressing. Therefore a minimalistic design for the grid would be the clearest for them to comprehend. The creation of the grid also need to be simple for the player to use, textboxes that only allow integers between 1 and 10 to be entered would allow for the player to easily enter the grid size that they need, the textboxes should also only allow for one value for the width and the height respectively. However the game should allow for new values to be entered if the player has finished the game and has chosen to play another round of the game. The program should draw a single box for each coordinate represented by the width and the height, this should be drawn clearly as to not confuse the player. As the game progresses on the board should change accordingly, if the player places a valid move then the symbol should appear on the board and the surrounding squares should be shaded out so that it is clear to see that they have been taken up and piece cannot be placed there. The same should occur when the artificial intelligence places their piece, the symbol should appear and the surrounding squares should be shaded out.

The interface should also be simple for the player to understand and to use, the player should be able to simply click where they would like to place their piece. The program should then check whether the move is valid or not, if the move is invalid a clear message should appear to inform the player that their move was invalid they should then be able to try to place their piece again. It should be simple to see for the player when the program is waiting for an input from the player, or when the game is won. When the game is won the program should display the total games won by the player and the total won by the computer, the player should then have the option to either play another game or to exit the program.

The Artificial intelligence should be effective and for each turn it should calculate the best possible move. It should be able to do this using the minimax algorithm to be able to choose the most beneficial place for it to be able to win the game. It should use the algorithm for every go so that each move is the most beneficial in response to the move made by the player to create the most effective gameplay.

The game should also have the option at any point to save the game, the position of each of the pieces and whether pieces can be placed in each of the squares should be recorded, and the total number of games won by both the player and the computer should be stored. The total number of games won should also be automatically stored at the end of each game and if the player exits.

## Requirements

1. A board is created
    1.1. The size is the height and width entered by the user
        1.1.1. Height and width are entered via a textbox
            1.1.1.1. The only values allowed to be entered are integers between 1 and 9
            1.1.1.2. Only allows one value to be entered to create the grid
                1.1.1.2.1. Allowed to enter another set of values if the game has ended and the player selects to play another game round
    1.2. The system draws a box for each coordinate to create the grid
        1.2.1. The grid is clear to see and understand
2. The board changes as the game progresses
    2.1. If the players move is valid the players piece appears on the board
        2.1.1. The piece is placed
        2.1.2. The surrounding squares are shaded out
            2.1.2.1. Clear to see that a piece cannot be placed there
            2.1.2.2. Each player has a different colour to represent their moves
    2.2. The AI's move is placed on the board
        2.2.1. The AI's move is clear
3. A simple interface for the user
    3.1. Click to select where piece is wanted to be placed
    3.2. Use of buttons to navigate through the user interface
    3.3. A player's move is checked to be valid
        3.3.1. Clear message is shown if move is invalid
        3.3.2. The player can choose somewhere else to place their piece
    3.4. It is clear when it is the players turn to play their piece
    3.5. It is clearly shown when a game is won
        3.5.1. Displays the total number of games won by the player and by the computer
        3.5.2. Option to play another game
        3.5.3. Option to exit the game
4. An effective AI
    4.1. Use of the minimax algorithm
        4.1.1. The most beneficial place is chosen for the AI
        4.1.2. This is used for each move so that the algorithm can be used most effectively
    4.2. All AI inputs are valid
5. Game is saved
    5.1. When the game is exited the game is saved as current game
    5.2. The player has the option to save the game under a title at any point
        5.2.1. A clear message is displayed if there is an error with saving
    5.3. The moves are recorded
    5.4. The number of games won by the player and by the computer is recorded and stored

# Design

## Files

The files needed for the project will need to contain all the data to load up a previous game or the current game. The data will be held in a simple text file and separated by commas; the data will be held as follows;

2player game or AI

Player (X or O), the button coordinates (repeated for all moves)

For the total number of games won a different text file will be user. It will be saved as follows;

Games won by player 1 vs player 2, games won by player 2 vs player 1

Games won by player 1 vs AI, games won by AI vs player 1

## Algorithms

Below outline some of the main processes that happen within the program

### Drawing the grid out

```
Ylim ← TEXTBOXY TEXT

Xlim ← TEXTBOXX TEXT

FOR int x = 0 TO Xlim – 1
        FOR int y = 0 TO Ylim – 1
                ADD BUTTON AT (X*50, Y*50)
        END FOR
END FOR
```

### switching player

the process of switching the player from X to O or from O to X

```
IF player = ‘X’ THEN
        Player = ‘O’
ELSE
        Player = ‘X’
END IF
```

### Click on the grid

When the player clicks on a button in the grid

```
BUTTON(X,Y) ← BUTTON CLICKED
BUTTON (X, Y) TEXT = player
FOR I = X-1 TO X +1
    FOR J = Y-1 TO Y+1
        IF X > -1 AND X < TEXTBOXX -1 AND Y> -1 AND Y< TEXTBOXY – 1 THEN
            BUTTON (X, Y) = DISABLED
            Check if won
        END IF
    END FOR
END FOR
```

| | | |
|---|---|---|
| X-1, Y-1 | X, Y-1 | X+1, Y-1 |
| X-1, Y | X, Y | X+1, Y |
| X-1, Y+1 | X, Y+1 | X+1, Y+1 |

### Game won

```
IF numberofgridbuttonsclicked = (TEXTBOXX * TEXTBOXY) THEN
    GAME = WON
ELSE
    Numberofgridbuttonsclicked = numberofgridbuttonsclicked + 1
```

### AI Algorithm

the minimax algorithm

call with Minimax (XPOSITION, YPOSITION, Depth, TRUE)

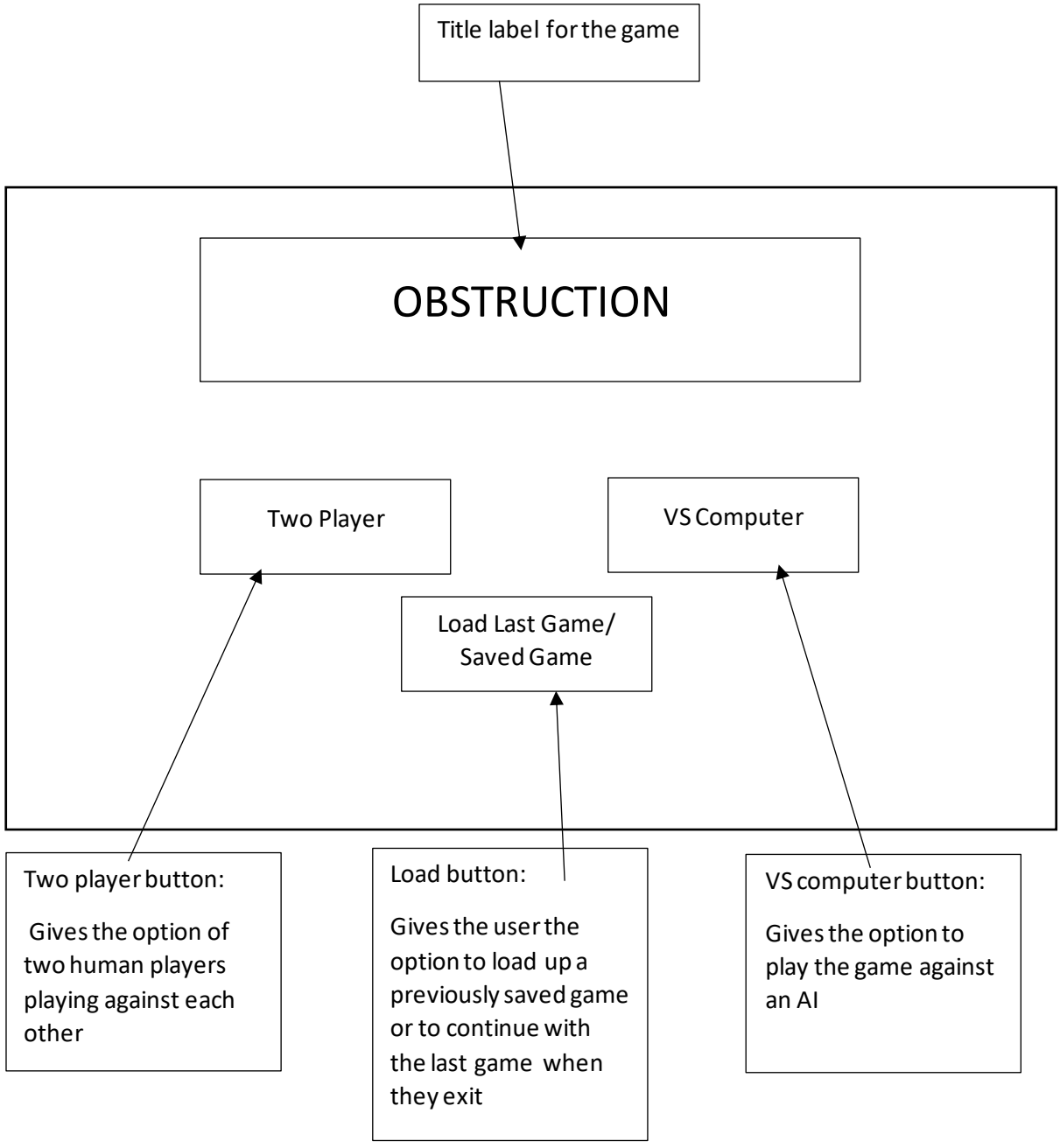
```
FUNCTION Minimax (XPOSITION, YPOSITION, Depth, MaxmisingPlayer)
    IF Depth = 0 or GameIsWon then
        RETURN Heuristic score
    ELSE IF Maximising player THEN
        MaxScore =- ∞
        Score = Heuristic score
        IF Score > MaxScore THEN
            MaxScore = score
```

```
            ELSE
                Minimax (XPOSITION, YPOSITION, Dept – 1, FALSE)
            END IF
        ELSE
            Minscore = + ∞
            Score = Heuristic score
            IF Score > Minscore THEN
                Minscore = score
            ELSE
                Minimax (POSITIONX, POSITIONY, Depth -1, TRUE)
            END IF
        END IF
END FUNCTION
```

## Front end design

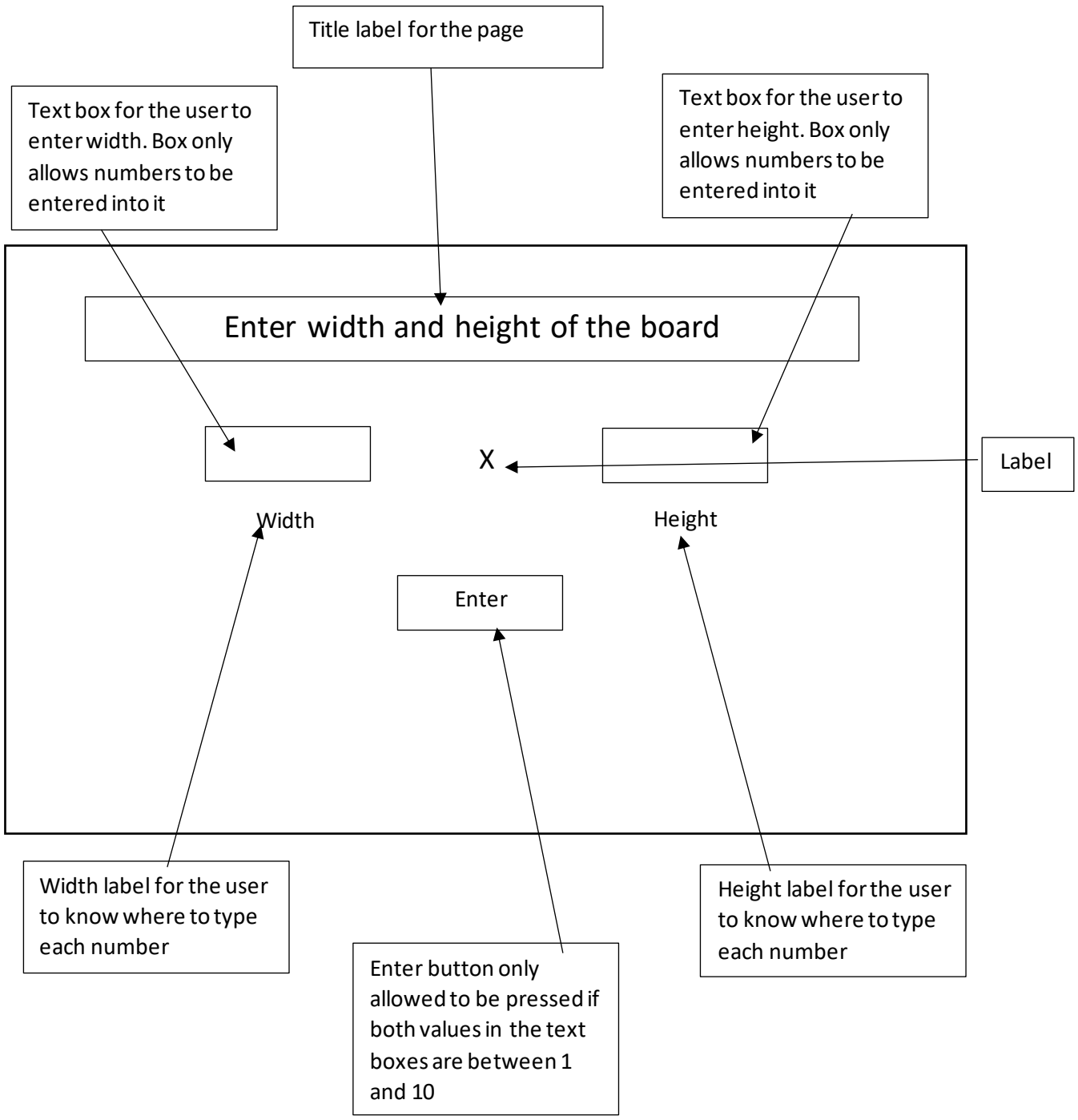
### Main menu

This is what first loads when the game is opened.



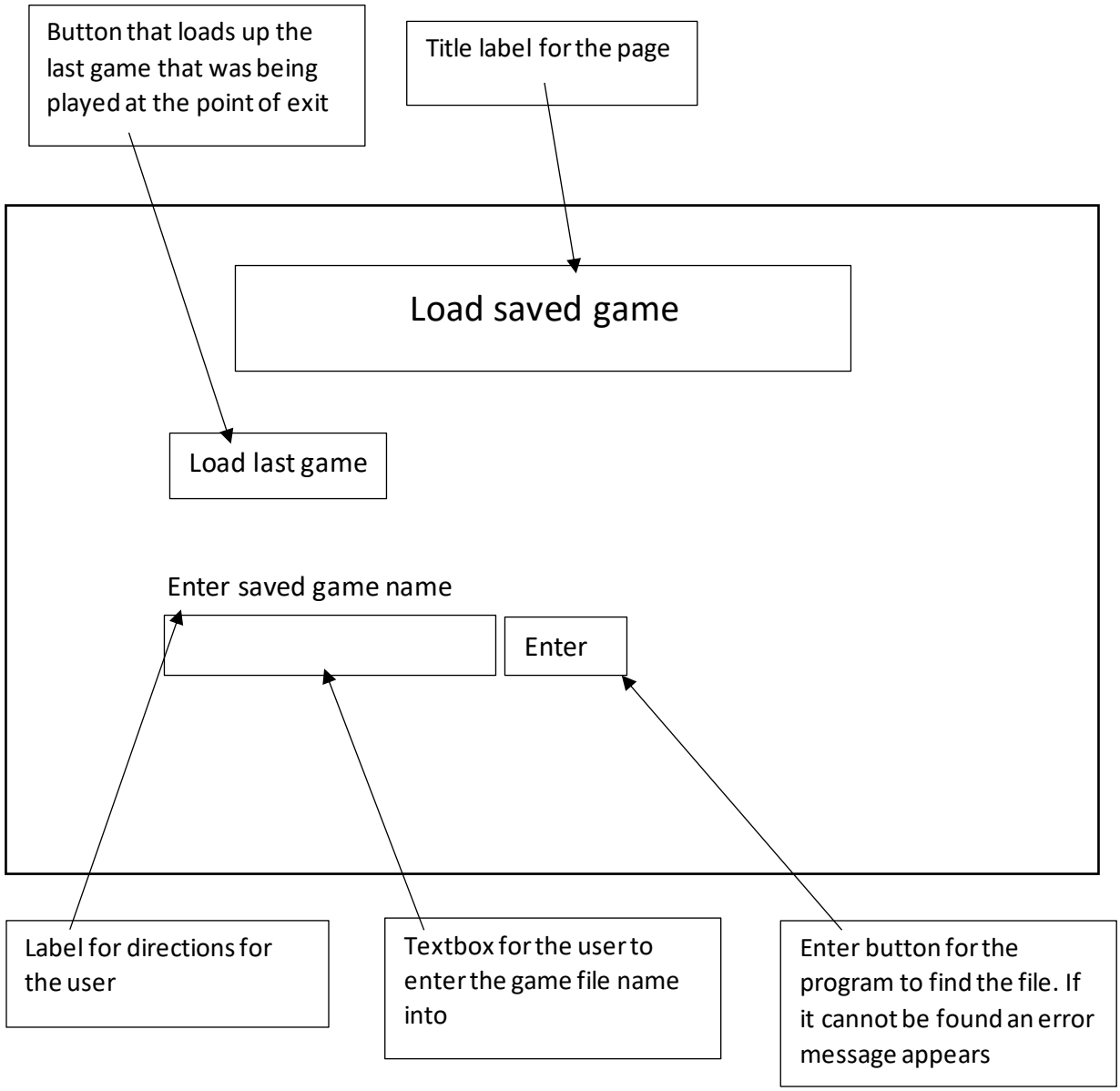
## Board size

This is what loads up after the main menu when either the two player or computer option is picked by the user



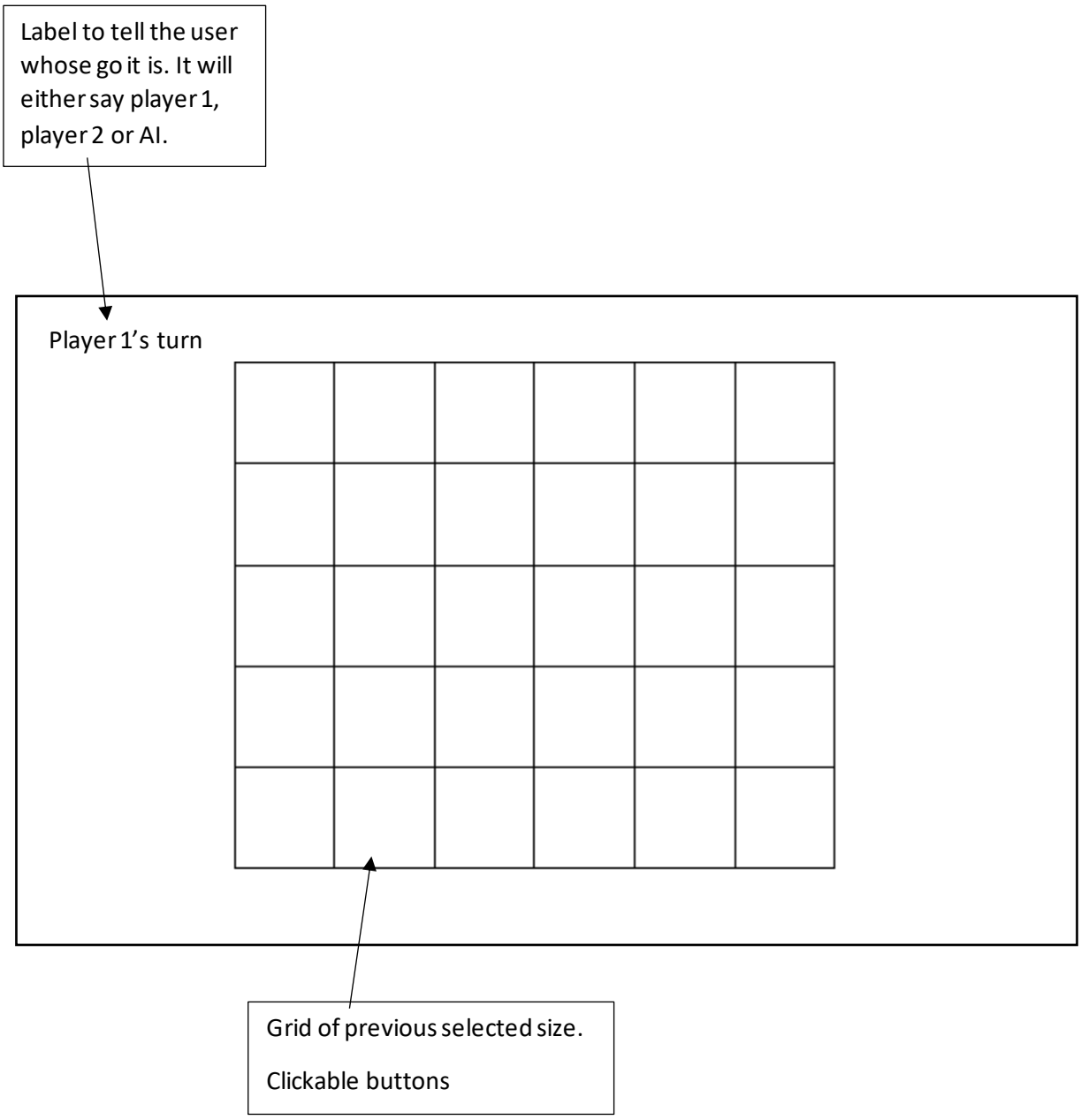
## Loading up a previous game

This is what is loaded up if the load button is pressed at the main menu



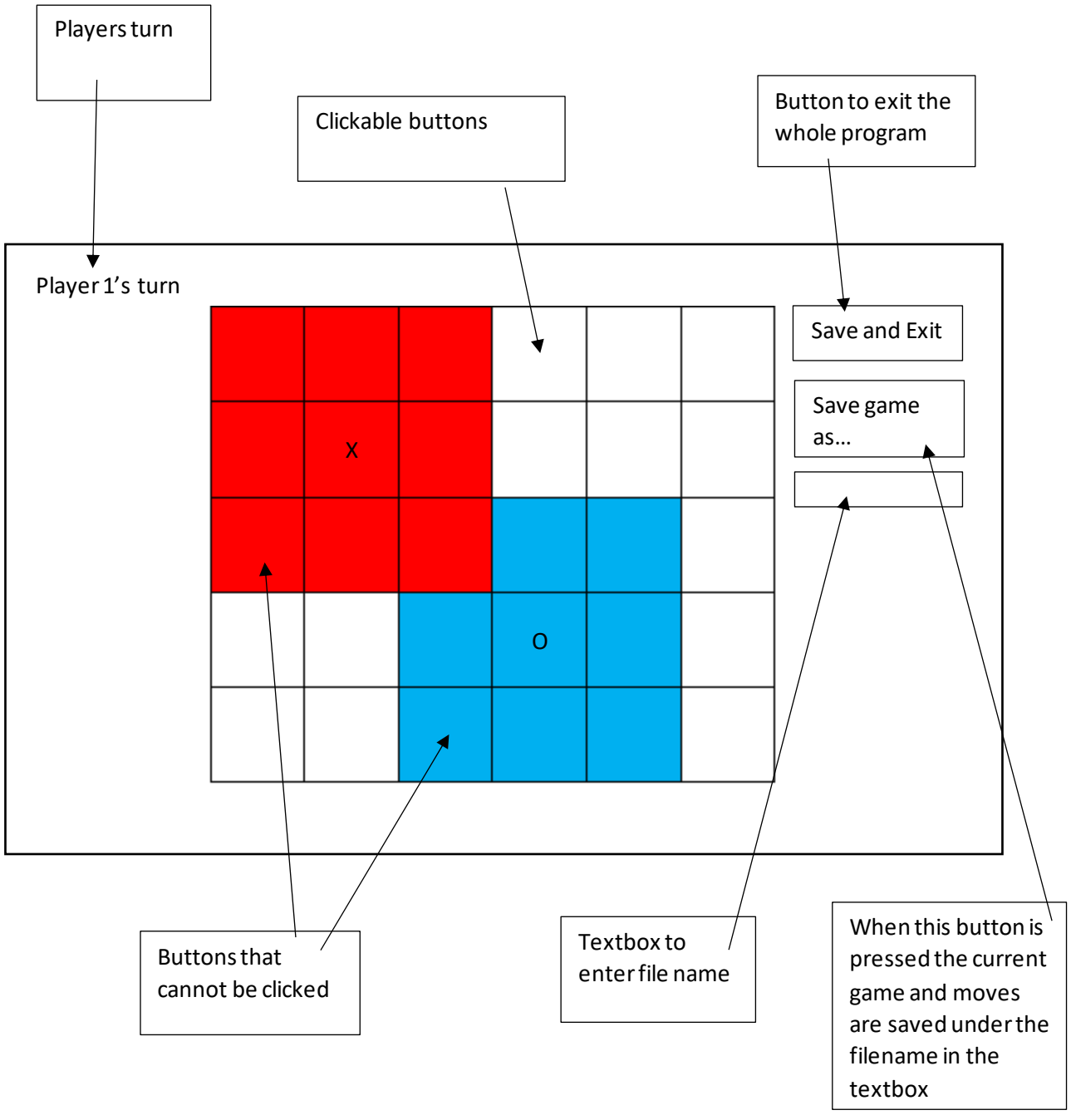
## Opening grid

This is what is loaded up if a blank grid is needed at the beginning of the game



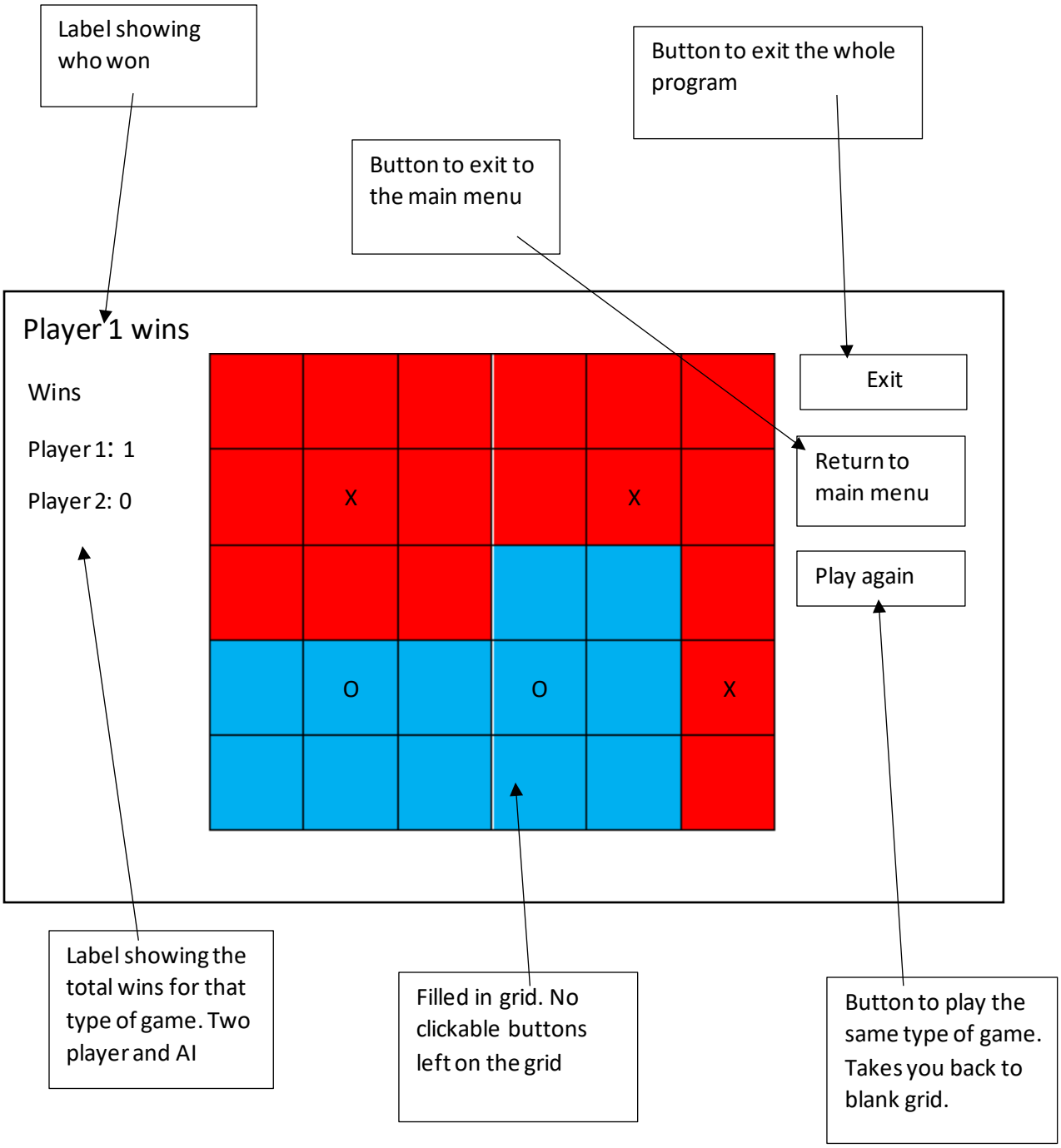
## Mid Game

Either part through game or a loaded game



## Game won

This shows a situation where the game is won by someone



## Testing strategy

Input tests for grid size, button clicks and entering a correctly formatted file name

Processing tests for minimax algorithm

Storage tests for saving a game under a name and for saving the game when exiting and saving the total wins

Output tests for label changing and winning a game

| Test Number | Description | Data type | Input | Expected result | Pass/fail | Cross reference |
|---|---|---|---|---|---|---|
| I 1 | Board grid size test | typical | 4, 5 | Should appear with a grid of 4x5 | Pass | https://youtu.be/Ste8k13yyNk 0:25 |
| I 2 | Board grid size test | typical | 6, 6 | Should appear with a grid of 6x6 | Pass | https://youtu.be/Ste8k13yyNk 0:39 |
| I 3 | Board grid size test | erroneous | A, 5 | Shouldn't allow the letter to be entered at all | Pass | https://youtu.be/Ste8k13yyNk 0:51 |
| I 4 | Board grid size test | erroneous | 5, a | Shouldn't allow the letter to be entered at all | Pass | https://youtu.be/Ste8k13yyNk 0:55 |
| I 5 | Board grid size test | erroneous | 10, 2 | Enter button will not accept the values | Pass | https://youtu.be/Ste8k13yyNk 1:03 |
| I 6 | Board grid size test | erroneous | 6, 0 | Enter button will not accept | Pass | https://youtu.be/Ste8k13yyNk 1:19 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | the values | | |
| I 7 | Board grid size test | Extreme | 1,9 | Should appear with a grid of 1x9 | Pass | https://youtu.be/Ste8k13yyNk 1:30 |
| I 8 | Test for when a button on the grid is clicked | typical | Click enabled button | Should disable the button, put the correct character on it (X or O) and disable the surrounding buttons | Pass | https://youtu.be/Ste8k13yyNk 1:38 |
| I 9 | Test for when a button on the grid is clicked | erroneous | Click disabled button | Nothing should happen | Pass | https://youtu.be/Ste8k13yyNk 2:00 |
| I 10 | Test for clicking of menu buttons | typical | Click vs computer option | Should take you to the enter grid size page. | Pass | https://youtu.be/Ste8k13yyNk 2:12 |
| I 11 | Test for file saving name | Typical | Enter MySavedGame.txt | Should accept it as a file name | Pass | https://youtu.be/Ste8k13yyNk 2:31 |
| I 12 | Test for file saving name | erroneous | Enter currentgame.txt | Shouldn't accept it as that is the file name that the program uses to save the | Pass | https://youtu.be/Ste8k13yyNk 2:55 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | current game | | |
| I 13 | Test for file saving name | errone ous | Enter savegame | Should reject as it is not in the correct format | Pass | https://youtu.be/Ste8k13yyNk 3:05 |
| I 14 | Test for file saving name | Errone ous | Enter .txt | Should reject as it is not a full file name | Pass | https://youtu.be/Ste8k13yyNk 3:21 |
| I 15 | Test for file saving name | Errone ous | Don't enter anything into the textbox | Should reject | Pass | https://youtu.be/Ste8k13yyNk 3:29 |
| I 16 | Test for file saving name | Extrem e | Enter 1.txt | Should accept it as a file name | Pass | https://youtu.be/Ste8k13yyNk 3:36 |
| S 1 | test for saving a game | | Save game under MySavedGame.txt Then reload and open the game | Should reload the board exactly as it was at the time that it was saved. | Pass | https://youtu.be/s8nxOcAn4sU 0:30 – 1:00 |
| S 2 | Test for the saving of the current game | | Exit a game midway through. Then open the game and select the load last game option | Should reload at the same place that the game was exited at. | Pass | https://youtu.be/s8nxOcAn4sU 2:07 – 2:17 |
| S 3 | Test for saving total wins | | Play a full game on two player | The total wins for a two player option should | Pass | https://youtu.be/s8nxOcAn4sU 2:17 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | load, update and display when the game is won | | |
| S 4 | Test for saving total wins | | Play a full game on Vs AI | The total wins for the vs AI option should load, update and display when the game is won | Pass | https://youtu.be/s8nxOcAn4sU 2:46 |
| P 1 | Testing minimax algorithm | | Play on a 3x6 board with player clicking in 2,2 | The AI should click in 2,5 to win the game | Pass | https://youtu.be/T7dHiNxUOKQ 0:20 |
| P 2 | Testing minimax algorithm | | Play on a 6x3 board with player clicking in 2,2 | The AI should click in 5,2 to win the game | Pass | https://youtu.be/T7dHiNxUOKQ 0:30 |
| P 3 | Testing minimax algorithm | | Play on a 4 x 3 board with player clicking in 2,2 | The AI should click in 4,2 to win the game | Pass | https://youtu.be/T7dHiNxUOKQ 0:45 |
| P 4 | Testing minimax algorithm | | play on a 3x 4 board with player clicking in 2,2 | The AI should click in 2,4 to win the game | Pass | https://youtu.be/T7dHiNxUOKQ 0:57 |
| P 5 | Testing minimax | | Play on a 3x 3 board with | The AI should | Pass | https://youtu.be/T7dHiNxUOKQ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | algorithm | | player clicking in 1,1 | click in 3,3 to block the players win | | 1:06 |
| O 1 | Test for the players go label changing | | On two player option 1st player go | Label should change from Player X's go to Player O's go | | https://youtu.be/T5tlfLypDzo 0:18 |
| O 2 | Test for game being won | | Play a game and make sure that all buttons are disabled | The last player to go should be announced as the winner | | https://youtu.be/T5tlfLypDzo 0:30 |

# Evaluation

## Meeting of requirements

I believe that my programme has met the requirements well. The programme creates the board with inputs from the user that only allows integers between 1 and 9 and then creates a clear board. The board is created of buttons that allow the users to easily click only the available buttons, when an available button is clicked the surrounding buttons are coloured with the coordinating colours to the player. There is an option to play either with an AI player or against another human player. The programme stores the total wins for each of the players of the two types of games, the games are saved after each move so that if a game is left it can be returned to by clicking the load last game option button. There is also the option to save any game at any point under a name which can then be reloaded at any other point in time, if there is an issue with saving it the player is notified and the programme does not crash. It handles if a name is entered that is being used for another part of the programme and asks for another name to be entered. There is a label that clearly displays who's go it is and who has won at the end. There is button to return to the main menu to be able to select another option. The AI player's move is always valid and uses the minimax algorithm to select a move.

Requirements

1. A board is created
   - 1.1. The size is the height and width entered by the user
     - 1.1.1. Height and width are entered via a textbox
       - 1.1.1.1. The only values allowed to be entered are integers between 1 and 9
       - 1.1.1.2. Only allows one value to be entered to create the grid
         - 1.1.1.2.1. Allowed to enter another set of values if the game has ended and the player selects to play another game round
   - 1.2. The system draws a box for each coordinate to create the grid
     - 1.2.1. The grid is clear to see and understand
2. The board changes as the game progresses
   - 2.1. If the players move is valid the players piece appears on the board
     - 2.1.1. The piece is placed
     - 2.1.2. The surrounding squares are shaded out
       - 2.1.2.1. Clear to see that a piece cannot be placed there
       - 2.1.2.2. Each player has a different colour to represent their moves
   - 2.2. The AI's move is placed on the board
     - 2.2.1. The AI's move is clear
3. A simple interface for the user
   - 3.1. Click to select where piece is wanted to be placed
   - 3.2. Use of buttons to navigate through the user interface
   - 3.3. A player's move is checked to be valid
     - 3.3.1. Clear message is shown if move is invalid
     - 3.3.2. The player can choose somewhere else to place their piece
   - 3.4. It is clear when it is the players turn to play their piece
   - 3.5. It is clearly shown when a game is won
     - 3.5.1. Displays the total number of games won by the player and by the computer
     - 3.5.2. Option to play another game
     - 3.5.3. Option to exit the game
4. An effective AI
   - 4.1. Use of the minimax algorithm
     - 4.1.1. The most beneficial place is chosen for the AI
     - 4.1.2. This is used for each move so that the algorithm can be used most effectively
   - 4.2. All AI inputs are valid
5. Game is saved
   - 5.1. When the game is exited the game is saved as current game
   - 5.2. The player has the option to save the game under a title at any point
     - 5.2.1. A clear message is displayed if there is an error with saving
   - 5.3. The moves are recorded
   - 5.4. The number of games won by the player and by the computer is recorded and stored

*Figure 17 Requirements*

| Requirement reference | Works (Y/N) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Y | 2 | Y | 3 | Y | 3.5.1 | Y | 5 | Y |
| 1.1. | Y | 2.1 | Y | 3.1 | Y | 3.5.2 | Y | 5.1 | Y |
| 1.1.1 | Y | 2.1.1 | Y | 3.2 | Y | 3.5.3 | Y | 5.2 | Y |
| 1.1.1.1 | Y | 2.1.2 | Y | 3.3 | Y | 4 | Y | 5.2.1 | Y |
| 1.1.1.2 | Y | 2.1.2.1 | Y | 3.3.1 | Y (just doesn't allow it) | 4.1 | Y | 5.3 | Y |
| 1.1.1.2.1 | Y | 2.1.2.2 | Y | 3.3.2 | Y | 4.1.1 | Y | 5.4 | Y |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1.2 | Y (buttons) | 2.2 | Y | 3.4. | Y | 4.1.2 | Y | | |
| 1.2.1 | Y | 2.2.1 | Y | 3.5 | Y | 4.2 | Y | | |

## Improvements

To improve my program further I could have developed an even more complex heuristic to make the game AI even better at defeating a player. This would take into account even more moves on even bigger boards as well to increase the difficulty for both the player and for the AI.

## Independent feedback

Feedback from my third party, Sophie, a student:

"I like the design it is clear and makes the game easy to use, the buttons mean that there is no confusion on how the game is played. I really like that the players have different colours as it means that the moves from each player are super clear and there is no confusion. There is a computer player to play against and it presents a challenge just as much as playing against another human player would. There is a scoring system that records the games won which is useful to keep track of how well I am doing. I also really like the game saving option as it means that I can come back to games, it also means that if I accidentally close it I can still keep on playing when I reload the game.

## Technical solution

```
Public Class Form1
    Dim openmenulabel, boardsizetitlelabel, xlabel, widthlabel, heightlabel,
loadgametitlelabel, savedgamelabel, gameboardplayerlabel, XVsOscoreXlabel,
XVsOscoreOlabel As New Label
    Dim twoplayerbtn, VsAIbtn, Loadgamebtn, widthheightenterbtn, loadlastgamebtn,
savedgameenterbtn, gridbtn, savegameasbtn, returntoMMbtn As New Button
    Dim widthtxtbox, heighttxtbox, savedgametxtbox, savegameastxtbox As New TextBox
    Dim board(,) As Button
    Dim boardwidth, boardheight, totalboardspaces, score As Integer
    Dim totalbtnsclicked As Integer = 0
    Dim currentplayer As Char = "O"
    Dim VsAIflag As Boolean = False
    Dim wonflag As Boolean
    Dim whowon As Char
    Dim currentboard As String

    'This starts off the program it loads the screen size and initiated the main menu
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MyBase.Height = 800
        MyBase.Width = 800
        MyBase.BackColor = Color.FloralWhite
        returntoMMbtn = New Button
        returntoMMbtn.Name = "returntoMMbtn"
        returntoMMbtn.Text = "Return to main menu"
        returntoMMbtn.Height = 50
        returntoMMbtn.Width = 100
        returntoMMbtn.BackColor = Color.Lavender
        returntoMMbtn.Font = New Font("Microsoft sans serif", 10)
```

```
        returntoMMbtn.Location = New Point(680, 25)
        Me.Controls.Add(returntoMMbtn)
        AddHandler returntoMMbtn.Click, AddressOf returntoMMbtn_click
        Dim filewriter As System.IO.StreamWriter
        filewriter = My.Computer.FileSystem.OpenTextFileWriter("XvsOscores.txt", True)
        filewriter.Close()
        openmenu()
    End Sub

    'This creates and loads the labels, buttons etc for the main menu
    Sub openmenu()
        currentboard = "openmenu"

        openmenulabel = New Label
        openmenulabel.Name = "openmenulabel"
        openmenulabel.Text = "Obstruction"
        openmenulabel.Height = 50
        openmenulabel.Width = 190
        openmenulabel.Font = New Font("Microsoft Sans Serif", 25)
        openmenulabel.Location = New Point(305, 50)
        Me.Controls.Add(openmenulabel)

        twoplayerbtn = New Button
        twoplayerbtn.Name = "twoplayerbtn"
        twoplayerbtn.Text = "Two Player"
        twoplayerbtn.Height = 50
        twoplayerbtn.Width = 150
        twoplayerbtn.BackColor = Color.LavenderBlush
        twoplayerbtn.Font = New Font("Microsoft Sans Serif", 15)
```

```
twoplayerbtn.Location = New Point(100, 150)
Me.Controls.Add(twoplayerbtn)
AddHandler twoplayerbtn.Click, AddressOf twoplayerbtn_Click



VsAIbtn = New Button
VsAIbtn.Name = "VsAIbtn"
VsAIbtn.Text = "Vs Computer"
VsAIbtn.Height = 50
VsAIbtn.Width = 150
VsAIbtn.BackColor = Color.LavenderBlush
VsAIbtn.Font = New Font("Microsoft Sans Serif", 15)
VsAIbtn.Location = New Point(550, 150)
Me.Controls.Add(VsAIbtn)
AddHandler VsAIbtn.Click, AddressOf VsAIbtn_Click

Loadgamebtn = New Button
Loadgamebtn.Name = "Loadgamebtn"
Loadgamebtn.Text = "Load last game/saved game"
Loadgamebtn.Height = 75
Loadgamebtn.Width = 200
Loadgamebtn.BackColor = Color.LavenderBlush
Loadgamebtn.Font = New Font("Microsoft sans serif", 15)
Loadgamebtn.Location = New Point(300, 250)
Me.Controls.Add(Loadgamebtn)
AddHandler Loadgamebtn.Click, AddressOf loadgamebtn_Click



End Sub
```

```
'This makes all of the items on the main menu invisible so that the screen doesnt become confusing for the user
Sub closeopenmenu()
    openmenulabel.Visible = False
    twoplayerbtn.Visible = False
    VsAIbtn.Visible = False
    Loadgamebtn.Visible = False
End Sub

'This creates the screen for the user to enter the board sizes
Sub boardsizemenu()
    currentboard = "boardsizemenu"
    boardsizetitlelabel = New Label
    boardsizetitlelabel.Name = "boardsizetitlelabel"
    boardsizetitlelabel.Text = "Enter the width and height of the board"
    boardsizetitlelabel.Height = 50
    boardsizetitlelabel.Width = 600
    boardsizetitlelabel.Font = New Font("Microsoft sans serif", 25)
    boardsizetitlelabel.Location = New Point(100, 100)
    Me.Controls.Add(boardsizetitlelabel)

    widthtxtbox = New TextBox
    widthtxtbox.Name = "widthtxtbox"
    widthtxtbox.Text = ""
    widthtxtbox.Height = 50
    widthtxtbox.Width = 100
    widthtxtbox.BackColor = Color.WhiteSmoke
    widthtxtbox.Font = New Font("Microsoft sans serif", 15)
    widthtxtbox.Location = New Point(200, 200)
```

```
Me.Controls.Add(widthtxtbox)
AddHandler widthtxtbox.KeyPress,  AddressOf checknumber

heighttxtbox = New TextBox
heighttxtbox.Name = "heighttxtbox"
heighttxtbox.Text = ""
heighttxtbox.Height = 50
heighttxtbox.Width  = 100
heighttxtbox.BackColor = Color.WhiteSmoke
heighttxtbox.Font  = New Font("Microsoft sans serif", 15)
heighttxtbox.Location = New Point(500, 200)
Me.Controls.Add(heighttxtbox)
AddHandler heighttxtbox.KeyPress,  AddressOf checknumber

xlabel = New Label
xlabel.Name = "xlabel"
xlabel.Text = "X"
xlabel.Height = 50
xlabel.Width = 20
xlabel.Font = New Font("Microsoft  sans serif", 15)
xlabel.Location = New Point(390, 200)
Me.Controls.Add(xlabel)

widthlabel = New Label
widthlabel.Name = "widthlabel"
widthlabel.Text = "Width"
widthlabel.Height = 20
widthlabel.Width = 50
widthlabel.Font = New Font("Microsoft sans serif", 10)
```

```
        widthlabel.Location = New Point(225, 230)
        Me.Controls.Add(widthlabel)

        heightlabel = New Label
        heightlabel.Name = "heightlabel"
        heightlabel.Text = "Height"
        heightlabel.Height = 20
        heightlabel.Width = 50
        heightlabel.Font = New Font("Microsoft sans serif", 10)
        heightlabel.Location = New Point(525, 230)
        Me.Controls.Add(heightlabel)

        widthheightenterbtn = New Button
        widthheightenterbtn.Name = "widthheightenterbtn"
        widthheightenterbtn.Text = "Enter"
        widthheightenterbtn.Height = 50
        widthheightenterbtn.Width = 100
        widthheightenterbtn.BackColor = Color.WhiteSmoke
        widthheightenterbtn.Font = New Font("Microsoft sans serif", 15)
        widthheightenterbtn.Location = New Point(350, 250)
        Me.Controls.Add(widthheightenterbtn)
        AddHandler widthheightenterbtn.Click, AddressOf widthheightenterbtn_Click
    End Sub

    'This closes the board size menu
    Sub closeboardsizemenu()
        boardsizetitlelabel.Visible = False
        widthtxtbox.Visible = False
        heighttxtbox.Visible = False
```

```
        xlabel.Visible = False
        widthlabel.Visible = False
        heightlabel.Visible = False
        widthheightenterbtn.Visible = False

    End Sub

    'This creates the menu for the user to load up previous games
    Sub Loadgamemenu()
        currentboard = "loadgamemenu"
        loadgametitlelabel = New Label
        loadgametitlelabel.Name = "loadgametitlelabel"
        loadgametitlelabel.Text = "Load saved game"
        loadgametitlelabel.Height = 50
        loadgametitlelabel.Width = 300
        loadgametitlelabel.Font = New Font("Microsoft sans serif", 25)
        loadgametitlelabel.Location = New Point(250, 50)
        Me.Controls.Add(loadgametitlelabel)

        loadlastgamebtn = New Button
        loadlastgamebtn.Name = "loadlastgamebtn"
        loadlastgamebtn.Text = "Load last game"
        loadlastgamebtn.Height = 50
        loadlastgamebtn.Width = 150
        loadlastgamebtn.Font = New Font("Microsoft sans serif", 12.5)
        loadlastgamebtn.Location = New Point(50, 150)
        Me.Controls.Add(loadlastgamebtn)
        AddHandler loadlastgamebtn.Click, AddressOf loadlastgamebtn_Click
```

```
        savedgamelabel = New Label
        savedgamelabel.Name = "savedgamelabel"
        savedgamelabel.Text = "Enter saved game name"
        savedgamelabel.Height = 20
        savedgamelabel.Width = 200
        savedgamelabel.Font = New Font("Microsoft sans serif", 10)
        savedgamelabel.Location = New Point(50, 250)
        Me.Controls.Add(savedgamelabel)

        savedgametxtbox = New TextBox
        savedgametxtbox.Name = "savedgametxtbox"
        savedgametxtbox.Text = ""
        savedgametxtbox.Height = 50
        savedgametxtbox.Width = 100
        savedgametxtbox.Font = New Font("Microsoft sans serif", 10)
        savedgametxtbox.Location = New Point(50, 270)
        Me.Controls.Add(savedgametxtbox)

        savedgameenterbtn = New Button
        savedgameenterbtn.Name = "savedgameenterbtn"
        savedgameenterbtn.Text = "Enter"
        savedgameenterbtn.Height = 25
        savedgameenterbtn.Width = 50
        savedgameenterbtn.Font = New Font("Microsoft sans serif", 10)
        savedgameenterbtn.Location = New Point(150, 270)
        Me.Controls.Add(savedgameenterbtn)
        AddHandler savedgameenterbtn.Click, AddressOf savedgameenterbtn_Click
    End Sub
```

```
    'This closes that menu
    Sub closeloadgamemenu()
        loadgametitlelabel.Visible = False
        loadlastgamebtn.Visible = False
        savedgamelabel.Visible = False
        savedgametxtbox.Visible = False
        savedgameenterbtn.Visible = False
    End Sub

    'This creates the initial board menu, it adds the labels etc, it also creates the buttons for the board
    'This also runs some functions that collect information about the board size and the player
    Sub gameboard()
        currentboard = "gameboard"
        gameboardplayerlabel = New Label
        gameboardplayerlabel.Name = "gameboardplayerlabel"
        gameboardplayerlabel.Text = ""
        gameboardplayerlabel.Height = 20
        gameboardplayerlabel.Width = 100
        gameboardplayerlabel.Font = New Font("Microsoft sans serif", 10)
        gameboardplayerlabel.Location = New Point(50, 50)
        Me.Controls.Add(gameboardplayerlabel)

        XVsOscoreXlabel = New Label
        XVsOscoreXlabel.Name = "XVsOscoreXlabel"
        XVsOscoreXlabel.Text = ""
        XVsOscoreXlabel.Height = 20
        XVsOscoreXlabel.Width = 500
        XVsOscoreXlabel.Font = New Font("Microsoft sans serif", 10)
        XVsOscoreXlabel.Location = New Point(150, 50)
```

```
Me.Controls.Add(XVsOscoreXlabel)

XVsOscoreOlabel = New Label
XVsOscoreOlabel.Name = "XVsOscoreOlabel"
XVsOscoreOlabel.Text = ""
XVsOscoreOlabel.Height = 20
XVsOscoreOlabel.Width = 500
XVsOscoreOlabel.Font = New Font("Microsoft sans serif", 10)
XVsOscoreOlabel.Location = New Point(260, 70)
Me.Controls.Add(XVsOscoreOlabel)

savegameasbtn = New Button
savegameasbtn.Name = "savegameasbtn"
savegameasbtn.Text = "Save game as..."
savegameasbtn.Height = 50
savegameasbtn.Width = 100
savegameasbtn.Font = New Font("Microsoft sans serif", 10)
savegameasbtn.Location = New Point(550, 100)
Me.Controls.Add(savegameasbtn)
AddHandler savegameasbtn.Click, AddressOf savegameasbtn_click

savegameastxtbox = New TextBox
savegameastxtbox.Name = "savegameastxtbox"
savegameastxtbox.Text = ""
savegameastxtbox.Height = 20
savegameastxtbox.Width = 100
savegameastxtbox.Font = New Font("Microsoft sans serif", 10)
savegameastxtbox.Location = New Point(550, 150)
Me.Controls.Add(savegameastxtbox)
```

```
    ReDim board(boardwidth, boardheight)
    wonflag = False
    For x = 0 To (boardwidth - 1)
      For y = 0 To (boardheight - 1)
        gridbtn = New Button
        gridbtn.Name = "gridbtn" & x & y
        gridbtn.Height = 50
        gridbtn.Width = 50
        gridbtn.Location = New Point(50 + (x * 50), 100 + (y * 50))
        gridbtn.Text = " "
        Me.Controls.Add(gridbtn)
        AddHandler gridbtn.Click, AddressOf gridbtnclick
        board(x, y) = gridbtn

      Next
    Next
    boardspaces()
    totalbtnsclicked = 0
    getplayer()
  End Sub

  'This closes the gameboard and removes all of the buttons
  Sub closegameboard()
    gameboardplayerlabel.Visible = False
    savegameasbtn.Visible = False
    savegameastxtbox.Visible = False
    XVsOscoreXlabel.Visible = False
    XVsOscoreOlabel.Visible = False
```

```
        For x = 0 To (boardwidth - 1)
            For y = 0 To (boardheight - 1)
                board(x, y).Visible = False
            Next
        Next

    End Sub

    'This is implemented when the return to main menu button is pressed, it closes the current board and opens the main menu
    Sub returntoMMbtn_click()
        Select Case currentboard
            Case "openmenu"
                MsgBox("You are already at the main menu!")
            Case "boardsizemenu"
                closeboardsizemenu()
                openmenu()
            Case "loadgamemenu"
                closeloadgamemenu()
                openmenu()
            Case "gameboard"
                closegameboard()
                openmenu()
        End Select
    End Sub

    'This runs when the two player option is picked on the main menu it sets the Vs AI flag to false and opens the board size menu
    Public Sub twoplayerbtn_Click(sender As Object, ByVal e As EventArgs)
```

```
        VsAIflag = False
        closeopenmenu()
        boardsizemenu()
    End Sub

    'This runs when the Vs AI option is picked on the main menu it sets the Vs AI flag to true and opens the board size menu
    Public Sub VsAIbtn_Click(sender As Object, ByVal e As EventArgs)
        VsAIflag = True
        closeopenmenu()
        boardsizemenu()
    End Sub

    'This runs when the load last game or saved game option is picked from the main menu it opens the menu to select previous games
    Public Sub loadgamebtn_Click(sender As Object, ByVal e As EventArgs)
        closeopenmenu()
        Loadgamemenu()
    End Sub

    'This runs when the enter button is clicked on the board size menu it checks that the values entered into the textboxes create a valid grid
    Public Sub widthheightenterbtn_Click(sender As Object, ByVal e As EventArgs)
        If widthtxtbox.Text <> "" And heighttxtbox.Text <> "" Then
            setgridsize()
            If boardwidth > 0 And boardheight > 0 And boardwidth < 10 And boardheight < 10 Then
                closeboardsizemenu()
                Dim file As System.IO.StreamWriter
                file = My.Computer.FileSystem.OpenTextFileWriter("currentgame.txt", False)
```

```
            file.WriteLine(boardwidth)
            file.WriteLine(boardheight)
            file.Close()
            gameboard()
        Else
            MsgBox("Please enter a value between 1 and 9")
        End If
    End If
End Sub

'This loads the last game when the button is clicked it opens the corresponding file and recreates the game
Public Sub loadlastgamebtn_Click(sender As Object, ByVal e As EventArgs)
    closeloadgamemenu()
    Dim file As System.IO.StreamReader
    Dim x As Integer = 1
    Dim y As Integer
    Dim z As String

    file = My.Computer.FileSystem.OpenTextFileReader("currentgame.txt")

    boardwidth = file.ReadLine()
    boardheight = file.ReadLine()
    z = file.ReadLine
    gameboard()
    Do While z <> Nothing
        x = Int32.Parse(z)
        y = Int32.Parse(file.ReadLine)
        board(x, y).Text = getplayer()
        makemove(x, y)
```

```
        z = file.ReadLine
      Loop
      file.Close()
  End Sub

  'This loads the save game when the button is clicked it opens the corresponding file and recreates the game if it can be found
  Public Sub savedgameenterbtn_Click(sender As Object, ByVal e As EventArgs)
    If savedgametxtbox.Text <> "" And savegameastxtbox.Text <> "XvsOscores.txt" And savegameastxtbox.Text <> "currentgame.txt" And savegameastxtbox.Text <> "score.txt" Then
      Dim file As System.IO.StreamReader
      Dim x As Integer
      Dim y As Integer

      Try
        file = My.Computer.FileSystem.OpenTextFileReader(savedgametxtbox.Text)
        boardwidth = file.ReadLine
        boardheight = file.ReadLine
        gameboard()
        Do Until file.EndOfStream
          x = file.ReadLine
          y = file.ReadLine
          board(x, y).Text = getplayer()
          makemove(x, y)
        Loop
        file.Close()
        closeloadgamemenu()
      Catch ex As Exception
        closegameboard()
```

```
            MsgBox("Game not found")
            Loadgamemenu()
        End Try
    Else
        MsgBox("Game not found try again")
    End If

End Sub

'This runs when the save game as button is clicked during a game is checks that the filename is valid and saves at that specific point in the game
Public Sub savegameasbtn_click(sender As Object, ByVal e As EventArgs)
    If savegameastxtbox.Text <> "" And Microsoft.VisualBasic.Right(savegameastxtbox.Text, 4) = ".txt" And savegameastxtbox.Text.Length > 4 And savegameastxtbox.Text <> "XvsOscores.txt" And savegameastxtbox.Text <> "currentgame.txt" And savegameastxtbox.Text <> "score.txt" Then
        Dim file As System.IO.StreamWriter
        Dim currentfile As System.IO.StreamReader
        Dim line As String
        Try
            file = My.Computer.FileSystem.OpenTextFileWriter(savegameastxtbox.Text, False)
            currentfile = My.Computer.FileSystem.OpenTextFileReader("currentgame.txt")
            Do
                line = currentfile.ReadLine
                file.WriteLine(line)
            Loop Until currentfile.EndOfStream
            file.Close()
            currentfile.Close()
            MsgBox("Saved")
        Catch ex As Exception
            MsgBox("Could not save")
```

```
            End Try
        ElseIf savegameastxtbox.Text = "" Then
            MsgBox("Please enter a file name")
        ElseIf Microsoft.VisualBasic.Right(savegameastxtbox.Text, 4) <> ".txt" Then
            MsgBox("Couldn't save, make sure it ends in .txt")
        ElseIf savegameastxtbox.Text.Length <= 4 Then
            MsgBox("Please enter a longer file name ")
        ElseIf savegameastxtbox.Text = "XvsOscores.txt" Or savegameastxtbox.Text =
"currentgame.txt" Or savegameastxtbox.Text = "score.txt" Then
            MsgBox("Try another name!")
        Else
            MsgBox("Could not save")
        End If
    End Sub

    'This checks that only integers are entered into the board size text boxes
    Sub checknumber(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs)
        If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) Then
            e.Handled = True
        End If
    End Sub

    'This takes the values from the board size textboxes and assigns them to a variable
    Sub setgridsize()
        boardwidth = Int32.Parse(widthtxtbox.Text)
        boardheight = Int32.Parse(heighttxtbox.Text)
    End Sub
```

```
    'This handles when a button is clicked on the board, it checks the move is valid, runs the function to make the move
    'it checks if the game is being played against a computer and runs the minimax if it is
    'it checks to see if the game has been won
    Sub gridbtnclick(sender As Object, ByVal e As EventArgs)
        If DirectCast(sender, Button).Enabled = False Then
            MsgBox("Invalid move")

        Else
            Dim X As Integer = Int32.Parse(DirectCast(sender, Button).Name(7))
            Dim Y As Integer = Int32.Parse(DirectCast(sender, Button).Name(8))
            Dim file As System.IO.StreamWriter
            Dim currentX As Integer
            Dim currentY As Integer
            Dim currentscore As Integer
            Dim compareX As Integer
            Dim compareY As Integer
            Dim comparescore As Integer
            Dim clearfile As System.IO.StreamWriter
            Dim scorefile As System.IO.StreamReader
            Dim filereader As System.IO.StreamReader
            Dim Xwon As Integer
            Dim Owon As Integer
            Dim depth As Integer = 3
            file = My.Computer.FileSystem.OpenTextFileWriter("currentgame.txt", True)
            file.WriteLine(X)
            file.WriteLine(Y)
            file.Close()
            board(X, Y).Text = getplayer()
            makemove(X, Y)
```

```
If wonflag = False Then
    If VsAIflag = True Then
        clearfile = My.Computer.FileSystem.OpenTextFileWriter("Score.txt", False)
        clearfile.Write("")
        clearfile.Close()
        For positionX = 0 To boardwidth - 1
            For positionY = 0 To boardheight - 1
                If board(positionX, positionY).Enabled = True Then
                    MiniMax(positionX, positionY, depth, True)

                    MiniMax(positionX, positionY, depth, False)
                End If
            Next
        Next

        scorefile = My.Computer.FileSystem.OpenTextFileReader("score.txt")
        currentX = scorefile.ReadLine
        currentY = scorefile.ReadLine
        currentscore = scorefile.ReadLine
        Do Until scorefile.EndOfStream
            compareX = scorefile.ReadLine
            compareY = scorefile.ReadLine
            comparescore = scorefile.ReadLine
            If comparescore > currentscore Then
                currentX = compareX
                currentY = compareY
                currentscore = comparescore
            End If
        Loop
```

```
            scorefile.Close()
            board(currentX, currentY).Text = getplayer()
            Threading.Thread.Sleep(500)
            makemove(currentX, currentY)
            file = My.Computer.FileSystem.OpenTextFileWriter("currentgame.txt", True)
            file.WriteLine(currentX)
            file.WriteLine(currentY)
            file.Close()
          End If
        End If

        If wonflag = True Then
          gameboardplayerlabel.Text = whowon & " won!"
          filereader = My.Computer.FileSystem.OpenTextFileReader("XVsOscores.txt")
          If VsAIflag = True Then
            filereader.ReadLine()
            filereader.ReadLine()
          End If
          Xwon = filereader.ReadLine()
          Owon = filereader.ReadLine()
          XVsOscoreXlabel.Text = ("Total won games: X = " & Xwon)
          XVsOscoreOlabel.Text = (" O = " & Owon)
          filereader.Close()
        End If
      End If
    End Sub

    'This disables the button clicked and the surrounding buttons and runs the function to
check if won

    Sub makemove(X, Y)
```

```
        For a = (X - 1) To (X + 1)
            If a > -1 And a < boardwidth Then
                For b = (Y - 1) To (Y + 1)
                    If b > -1 And b < boardheight Then
                        If board(a, b).Enabled = True Then

                            board(a, b).Enabled = False
                            If currentplayer = "X" Then
                                board(a, b).BackColor = Color.Crimson
                            Else
                                board(a, b).BackColor = Color.CornflowerBlue
                            End If

                            totalbtnsclicked = totalbtnsclicked + 1

                        End If
                    End If
                Next
            End If
        Next
        checkifwon()

    End Sub

    'This function selects and swaps the players
    Function getplayer()
        If currentplayer = "O" Then
            currentplayer = "X"
            gameboardplayerlabel.Text = "Player O's go"
```

```
            gameboardplayerlabel.ForeColor = Color.CornflowerBlue
        Else
            currentplayer = "O"
            gameboardplayerlabel.Text = "Player X's go"
            gameboardplayerlabel.ForeColor = Color.Crimson
        End If
        Return currentplayer
    End Function

    'This checks if the game has been won and if so updates the total games won scores
    Function checkifwon()
        boardspaces()
        Dim filereader As System.IO.StreamReader
        Dim filewriter As System.IO.StreamWriter
        Dim Xwon As Integer
        Dim Owon As Integer
        Dim XwonAI As Integer
        Dim OwonAI As Integer
        If totalbtnsclicked = totalboardspaces Then
            whowon = currentplayer
            currentplayer = "O"
            wonflag = True
            filereader = My.Computer.FileSystem.OpenTextFileReader("XVsOscores.txt")
            Xwon = filereader.ReadLine
            Owon = filereader.ReadLine
            XwonAI = filereader.ReadLine
            OwonAI = filereader.ReadLine
            filereader.Close()
            filewriter = My.Computer.FileSystem.OpenTextFileWriter("XvsOscores.txt", False)
```

```
        If whowon = "X" Then
            gameboardplayerlabel.ForeColor = Color.Crimson
            If VsAIflag = True Then
                filewriter.WriteLine(Xwon)
                filewriter.WriteLine(Owon)
                filewriter.WriteLine(XwonAI + 1)
                filewriter.WriteLine(OwonAI)
            Else
                filewriter.WriteLine(Xwon + 1)
                filewriter.WriteLine(Owon)
                filewriter.WriteLine(XwonAI)
                filewriter.WriteLine(OwonAI)
            End If

        Else
            gameboardplayerlabel.ForeColor = Color.CornflowerBlue
            If VsAIflag = True Then
                filewriter.WriteLine(Xwon)
                filewriter.WriteLine(Owon)
                filewriter.WriteLine(XwonAI)
                filewriter.WriteLine(OwonAI + 1)
            Else
                filewriter.WriteLine(Xwon)
                filewriter.WriteLine(Owon + 1)
                filewriter.WriteLine(XwonAI)
                filewriter.WriteLine(OwonAI)
            End If
        End If
        filewriter.Close()
```

```
    End If
    Return wonflag
  End Function

  'This calculates the total board spaces
  Sub boardspaces()
    totalboardspaces = boardwidth * boardheight
  End Sub

  'This runs the minimax algrithm
  Function MiniMax(positionX, positionY, depth, maximisingPlayer)
    Dim maxscore As Integer
    Dim minscore As Integer
    If wonflag = True Or depth = 0 Then
      Return Heuristic(positionX, positionY)
    End If

    If maximisingPlayer Then
      maxscore = Integer.MinValue
      score = Heuristic(positionX, positionY)
      If score > maxscore Then
        maxscore = score
      Else
        score = MiniMax(positionX, positionY, depth - 1, False)
      End If

    Else
      minscore = Integer.MaxValue
      score = Heuristic(positionX, positionY)
```

```
        If score > minscore Then
            minscore = score
        Else
            score = MiniMax(positionX, positionY, depth - 1, True)
        End If


    End If
End Function

'This runs the heuristic to calculate the score of each of the positions on the board
Function Heuristic(positionX, positionY)
    Dim enabledcount As Integer = 0
    Dim totalenabledcount As Integer = 0
    Dim otherenabled As Integer = 0

    If wonflag = True Then
        If whowon = "O" Then
            score += 10
        Else
            score -= 10
        End If

    ElseIf positionX > 0 And positionX < boardwidth - 1 And positionY > 0 And positionY < boardheight - 1 And board(positionX, positionY).Enabled = True Then


        For i = positionX - 1 To positionX + 1
            For j = positionY - 1 To positionY + 1
                If board(i, j).Enabled = True Then
```

```
                enabledcount += 1
            End If
        Next
    Next
    For x = 0 To boardwidth - 1
        For y = 0 To boardheight - 1
            If board(x, y).Enabled = True Then
                totalenabledcount += 1
            End If
        Next
    Next
    otherenabled = totalenabledcount - enabledcount
    If otherenabled = 0 Then
        score += 10
    ElseIf otherenabled = 1 Then
        score -= 10
    ElseIf (otherenabled Mod 2) = 1 Then
        score += 5
    Else
        score -= 5
    End If
Else
    For x = 0 To boardwidth - 1
        For y = 0 To boardheight - 1
            If board(x, y).Enabled = True Then
                totalenabledcount += 1
            End If
        Next
    Next
```

```
        If ((positionX = 0 And positionY = 0) Or (positionX = 0 And positionY = boardheight - 1)
Or (positionX = boardwidth - 1 And positionY = 0) Or (positionX = boardwidth - 1 And
            positionY = boardheight - 1)) And (boardwidth <> 1 And boardheight <> 1) Then


            Select Case positionX
              Case 0
                If positionY = 0 Then
                  If board(0, 1).Enabled = True And board(1, 0).Enabled = True Then
                    score += 8
                  Else
                    score += 4
                  End If
                Else
                  If board(0, boardheight - 2).Enabled = True And board(1, boardheight -
1).Enabled = True Then
                    score += 8
                  Else
                    score += 4
                  End If
                End If
              Case boardwidth - 1
                If positionY = 0 Then
                  If board(boardwidth - 1, 1).Enabled = True And board(boardwidth - 2,
0).Enabled = True Then
                    score += 8
                  Else
                    score += 4
                  End If
                Else
```

```
                If board(boardwidth - 1, boardheight - 2).Enabled = True And
board(boardwidth - 2, boardheight - 1).Enabled = True Then
                    score += 8
                Else
                    score += 4
                End If
            End If
        End Select


    Else
        If positionX > 0 And positionX < boardwidth - 1 Then
            If board(positionX - 1, positionY).Enabled = True And board(positionX + 1,
positionY).Enabled = True Then
                score += 7
            End If
        End If
        If positionY > 0 And positionY < boardheight - 1 Then
            If board(positionX, positionY - 1).Enabled = True And board(positionX, positionY +
1).Enabled = True Then
                score += 7
            End If
        End If
    End If
    If (totalboardspaces - totalbtnsclicked) Mod 2 = 0 Then
        score += 5
    End If
End If

Dim file As System.IO.StreamWriter
```

```
        file = My.Computer.FileSystem.OpenTextFileWriter("Score.txt", True)
        file.WriteLine(positionX)
        file.WriteLine(positionY)
        file.WriteLine(score)
        file.Close()
        Return score
    End Function
End Class
```