Tic Tac Toe

Andreas Nikolaou
40211330@live.napier.ac.uk
Edinburgh Napier University - Algorithms
Data Structures(SET08122)

Figure 1: option for Game mode

```
F:\uni\AlgorithmsAndDataStructures\cw>ticTacToe
Enter 1 for AI opponent or 2 for two human players:
```

1 Introduction

In the following document you will find out how a Tic Tac Toe game is run from terminal, and an adequately description of how it is implemented. Furthermore I'm going to show you how i constructed my code and how i used some of the data structures. There are different approaches in developing a Tic Tac Toe game like versus game mode (Playing with a second Human) or AI mode (Against a computer which chooses a random move and initiates the winning move when there is one available). In my implementation i give the user the option to choose whether to play against a human opponent, or a Computer opponent with user input in the terminal. Then, depending on user input the program responds accordingly.

Figure 2: Starting after selecting game mode

```
F:\uni\AlgorithmsAndDataStructures\cw>ticTacToe
Enter 1 for AI opponent or 2 for two human players:1

Board:

   -   -   -

   -   -   -

   -   -   -
Please enter a move from 1 to 9:
```

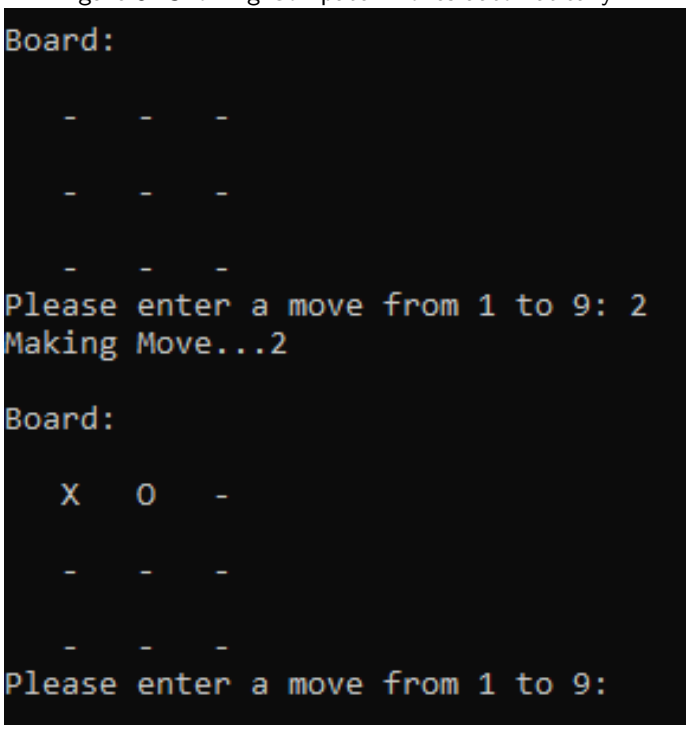
2 Design

During the introduction i talked up to the point where the user picks its option of which mode desires to play. But as most of us know, " it's what's behind the eyes that matters" in this case, behind the code. Firstly, like all the games out there we need our Environment, to build from there. I initialised the board using a "ConvertTo25" int array as declared in my code which isolates the 9 indexed squares that i needed to add functionality later on initialising a nice 9 squared board with boarders and with the appropriated spacing. Also i used enumerator build in function for the declaration of the elements to making up the board to compressed my code a bit. Furthermore, created a void function called RunGame() where basically i entered the iterations of the game rules (e.g.While(!GameOver) ...). Basically in this function i set the ground rules of how the game runs and when it ends (e.g.Won or Draw). Then, after having the main basic error handlers and parameters needed. It is time start getting Player moves. I created an integer function simply returning the specific move of the player and then printed on the board. I have also implemented a HasEmpty() function used in RunGame() function with the functionality to simply check within our board if there are any available positions left, and if there aren't then it returns the result of the game as Draw and Closes. Moreover, i then developed a new function called GetHumanMove() and it returns an integer index number of internal board (users input). Also in this function handles wrong user inputs stating what is the error with specific error messages. Then i created a similar function but this time for the computer moves called GetComputerMove. Now this function is different in respect to the human because a human is unpredictable and he inputs whatever he wants in our program. But the computer is not, because we can control its input option and amend with it. In my case i return an index integer randomly depending on the board positions available and using GetWinningMove function the computer opponent is also able to always finish the game with a win when you give it the chance (e.g. If the computer has two of the same symbols in any of the directions and the human user does not block its winning move, then the computer will finish it every time with the tic tac toe winning move). Before implementing the GetWinningMove though i did implement every possible winning move with the functions GetNumForDist which calculates the distance between the indexes and is used in the function FindThreeInARow, which determines the winner.

3 Enhancements

Furthermore nothing is ever perfect, so there are things that could be added to my TicTacToe Game and make it a lot bigger and better in terms of design, content and functionality. Firstly i will definitely need to add score in the future and option to prompt the user how many winning games should be played on according to the users preference . Additionally, the extra enhancements of replay a game, undo or redo a Player move which i would have include if i only had the time. Also i would like to make the computer player even more clever so it actualy goes for the win and learns as much

Figure 3: Showing Computer moves automatically.



times as the game is played but again too much things for the specific timeframe and workload. Lastly maybe the font to make it more user friendly because is very ugly in my opinion in the terminal at the moment.

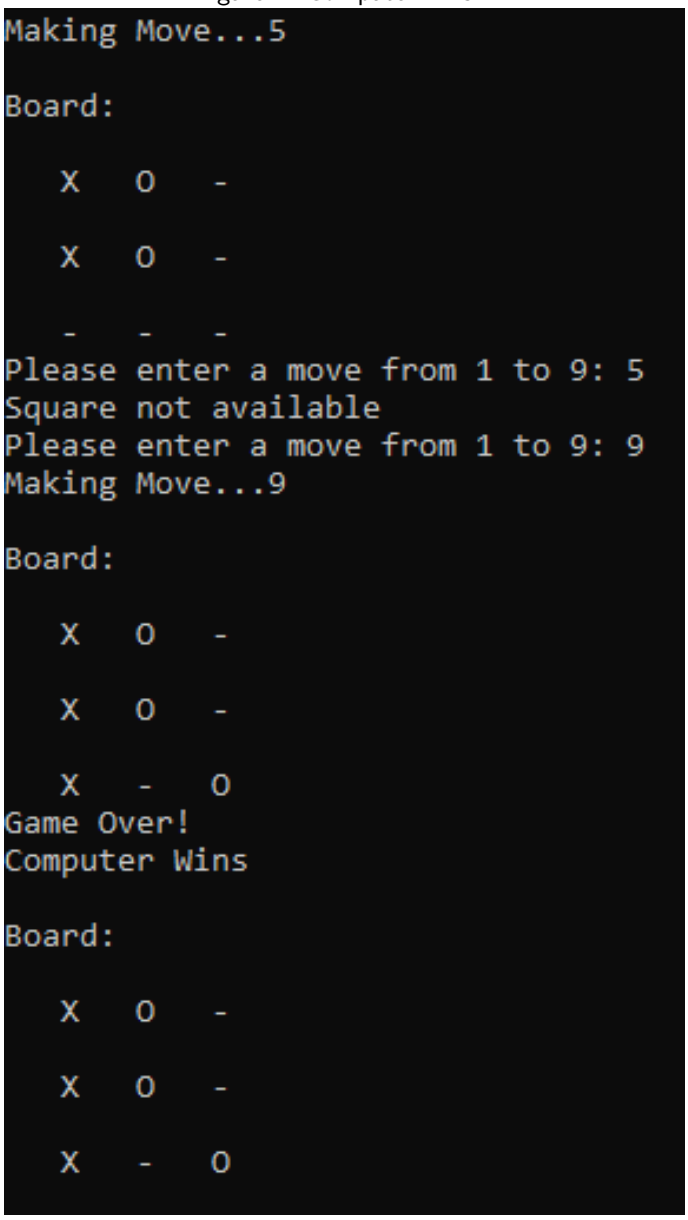
4 Critical Evaluation

In comparison to the original concept i did very well if we consider my misunderstanding in using the wrong programming language . I was asked to implement a simple TicTac-Toe game which i did, and i have researched about. I have to admit that the game is not very nice visually but it has a successful data structure and very good functionality with also very good error/crash handling . Although it's not perfect i believe i learned a lot of things in this coursework related with data structures.

5 Personal Evaluation

In my opinion i learned lots of things from this assessment like creating a game in the terminal. Seeing different cases and different errors that needed handling or different approaches in the coding construction. Familiarizing myself more with C programming language and the terminal. Also through conflicts from the terminal i learnt theory in the language that i haven't seen before learning new stuff. I feel that i could have added some more stuff in my project as detailed in the Enhancements previous section and that i could make it look perfect.

Figure 4: Computer wins.



References

Lab tutorials provided Lectures privided