Edrick Hansel Limantoro, Mikha Kelaya Wy, Roger Cashley

Muhammad Rausyan Fikri, S.Si, M.Eng, MIEEE, MAIAA.

Data Structure and Algorithm (COMP2424 – A)

December 8th, 2021

Final Project Report

Tic Tac Toe A.I. with Minimax Algorithm

**Project Overview**

The tic tac toe game is a famous simple game that is playable by two opposing players, one represents the character ‘X’ while the other ‘O’. With modern computing, players can now play the game against an “artificial intelligence”. The first iteration of the A.I. uses random slot selection and this makes the game “not challenging”, this gave birth to an algorithm called the minimax algorithm.

The minimax algorithm is a backtracking algorithm that optimizes the best next move for a player to take. Minimax only works for games with two players as each player will act as a minimizing player and a maximizing player. With tic tac toe, where there are only two players, minimax can be implemented for the computer to go against the opposing player. During the A.I.’s turn, minimax algorithm will generate and evaluate all possible move up to the depth given or when the algorithm reaches a solution that require less steps. These possible moves generated can be visualized and be travelled in a form of a tree data structure.

**Project Files**

GitHub Repository: <https://github.com/MikhaWy/TicTacToe.git>

Demo and Video Explanation:

<https://drive.google.com/file/d/1JaRZOaVVsdhN8B03KVP_WIO_XSqdvsf8/view?usp=sharing>

**Project Structure**

Folders in this project include:

1. Build

The build folder will contain the make.bat batch file and the compiled output files. The batch file will contain the original command line compile function:

> g++ -o main ../src/main.cpp ../lib/game.cpp ../lib/tttlibrary.cpp

> main

1. Docs

The document folder will hold reports files and used assets.

1. Includes

Contain header files (.h).

1. Lib

Contain library files (.cpp) for the header files.

1. Src

Contain the main.cpp file where the main function is found.

**Sample Input and Output**

**Vs. A.I.**

Text

Description automatically generated

Figure vs Minimax A.I. 1

Background pattern

Description automatically generated

Figure vs Minimax A.I. 2

Text

Description automatically generated with medium confidence

Figure vs Minimax A.I. 3

**Vs Human**

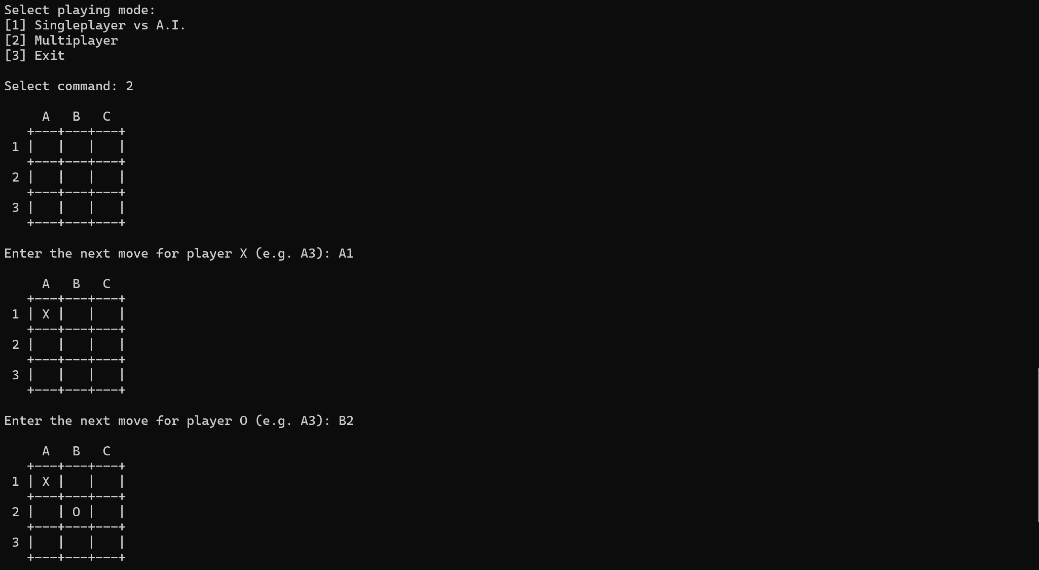
****

Figure vs Human 1

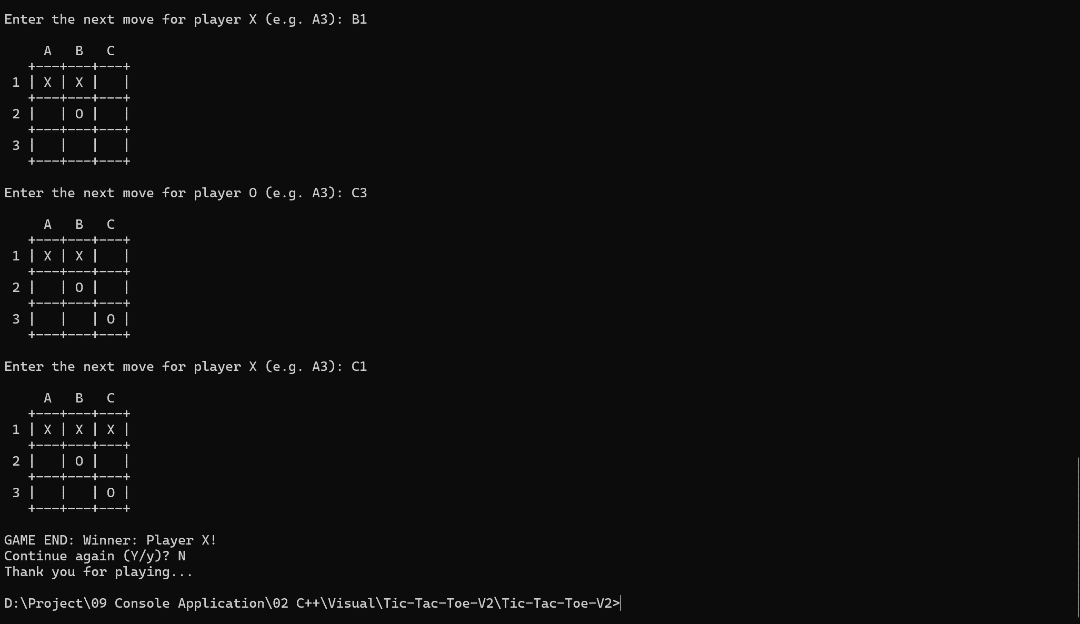
****

Figure vs Human 2

**Future Improvements**

Like every other project, there are always improvements that could be made. In this case, in the early game, the A.I. will always prioritize the left most grid, since the score evaluated on the board will be equal to the rest of the moves. Looking at that, we can create a pseudo random number generator that will make the A.I. randomly choose between the evaluated scores randomly, to make the game more “random”. Features such as:

* Choosing A.I. difficulty by manipulating the search depth
* User ranking
* Gameplay history
* User Interface (UI)

Could be added in future improvements. Time complexity for the minimax algorithm could be improved by introducing the alpha-beta pruning optimization.