AI PROJECT REPORT

A Project

on

“ TIC-TAC-TOE ”

Submitted by

Rahul Vaishnav (18ucs063)

Pinkesh Unadkat (18ucs091)

Deep Mavani (18ucs033)

Siddharth Singhvi (18ucs069)

Submitted to

Prof. Nitin Kumar

Dr. Poulami Dalapati

Dr. Roshni Chakraborty

INTRODUCTION

Problem Statement : Design tic-tac-toe game between two players where the first player will always have the winning strategy irrespective of the opponent's move.

Tic-Tac-Toe : Tic-tac-toe is not a very challenging game for human being. Here, we are using traditional 3x3 tic-tac-toe board game. The game consists of two players. One player has to opt for a nought and other for a cross. The players get alternate turn to place their mark of the board. The first player who succeeds in placing 3 consecutive marks (either vertically or horizontally or diagonally) will win the game.

In this project one player is computer and other is user, where the mark for computer is cross and that for user is nought. First move is always played by the computer.
The computer uses Minimax algorithm with alpha-beta pruning to play its move.

Methods /Algorithms Applied

1.) printboard :
    Parameters – playboard
    Return type – void
    Function – to print current playboard

2.) pos_filled:
    Parameters – playboard, position-pair
    Return type – bool
    Function – checks whether the given position is filled or not.

3.) get_filled_position :
    Parameters – playboard, mark
    Return type – vector pair
    Function – returns the vector of position-pairs which are filled with given mark.

4.) game_result :
    Parameters – filled-positions(vector-pairs)
    Return type – bool
    Function – returns true when any winning state is reached.

5.) gamestatus :

Parameters – playboard, mark
Return type – integer
Function – returns 0 when game is draw, 1000 when player with given mark wins, -1000 when player with given mark loses.

6.) minimax :

Parameters – playboard, mark, depth, alpha, beta
Return type – pair of pairs
Function – Minimax is a recursive algorithm for choosing the next move in this game. A value is associated with each state of the game. The player makes the move that maximizes the minimum value of the position resulting from the opponent's possible following moves.
Here, Alpha–beta pruning is used as it seeks to decrease the number of nodes that are evaluated by the Minimax algorithm in its search tree. This algorithm allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available.

7.) gameover :

Parameters – playboard
Return type – bool
Function – returns true when either a player has won the game or all the positions are filled in the game playboard.

Experiments (results)

Example 1:

```
       -----------
        Tic Tac Toe
       -----------

USER = O         COMPUTER = X

   |   |
-----------
   | X |
-----------
   |   |

Enter Row and Col (use 0 indexing) :
```

I/P - 0 2

```
Enter Row and Col (use 0 indexing) : 0 2
```

O/P :

```
X |   | O
-----------
   | X |
-----------
   |   |

Enter Row and Col (use 0 indexing) :
```

I/P - 2 2

```
Enter Row and Col (use 0 indexing) : 2 2
```

O/P :

```
 X |   | O
-----------
   | X | X
-----------
   |   | O

Enter Row and Col (use 0 indexing) :
```

I/P - 1 0

```
Enter Row and Col (use 0 indexing) : 1 0
```

O/P :

```
 X | X | O
-----------
 O | X | X
-----------
   |   | O

Enter Row and Col (use 0 indexing) :
```

I/P - 1 1

```
Enter Row and Col (use 0 indexing) : 1 1
```

O/P :

```
Invalid Move. Try again..
```

I/P - 2 1

```
Enter Row and Col (use 0 indexing) : 2 1
```

O/P :

```
 X | X | O
-----------
 O | X | X
-----------
 X | O | O

---- GAME OVER ----

It's A DRAW
```

Example 2 :-

```
       -----------
        Tic Tac Toe
       -----------

USER = O         COMPUTER = X

   |   |
-----------
   | X |
-----------
   |   |

Enter Row and Col (use 0 indexing) :
```

I/P - 2 0

```
Enter Row and Col (use 0 indexing) : 2 0
```

O/P :

```
 X |   |
-----------
   | X |
-----------
 O |   |

Enter Row and Col (use 0 indexing) :
```

I/P - 1 2

```
Enter Row and Col (use 0 indexing) : 1 2
```

O/P :

```
 X |   |
-----------
   | X | O
-----------
 O |   | X

---- GAME OVER ----

OOOOPSSS..... USER LOSE
```

Discussion / Insights / Observations

To make Minimax more efficient, alpha-beta pruning is used with it. Minimax with alpha-beta pruning is an adversarial search algorithm. It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

Conclusion

With the basis of minimax algorithm for deciding the move of computer and by using alpha beta pruning concept to speed up the process and optimizing the utility function using heuristic function, the 3x3 tic tac toe game was developed. We explored that a 3x3 tic tac toe, an adversary search technique game in artificial intelligence, can be developed using these techniques. Increasing the size of this game would create a huge time complexity issue with the same algorithm and techniques, for which other logics must be further researched.

Reference / Bibliography

1.) Minimax algorithm with alpha-beta pruning : https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/

2.) PDF file of tic-tac-toe game by Dr. Poulami Dalapati : https://drive.google.com/file/d/18HCcToAO2Tk3n0VGcIfuEUKjA2YBPvOk/view?usp=sharing

Contribution of Every Participant

Pinkesh Unadkat (18UCS091):- Defined Main function, winning states, integrated all methods and partial contribution in report preparation.

Rahul Vaishnav (18UCS063):- Minimax method, printboard method and partial contribution in report preparation.

Deep Mavani (18UCS033):- Gameover, Gamestatus methods and partial contribution in report preparation.

Siddharth Singhvi (18UCS069):- pos_filled , get_filled_positions, game_result methods and partial contribution in report preparation.

Link of the code and demo-video on google drive :

https://drive.google.com/drive/folders/1IWEY-IcWFVMz4kBP1AZNmemTpc_TpB8k?usp=sharing