**PROJECT REPORT FOR INTELLIGENT SYSTEMS(ITCS\_6150)**

**Tic-Tac-Toe Game solving using minimax with alpha-beta pruning**

**Project Category:** Game Playing

**Team Members:**

Kshitij Gorde -

Aditi Helekar - 800966667

**Problem Description:**

Implement a general game-playing agent for two-player deterministic games, using either (1) minimax with alpha-beta pruning, or (2) minimax-cutoﬀ (i.e., with cutoﬀ test to replace terminal test and with evaluation function to replace utility function) with alpha-beta pruning. Apply it to the planar 3×3 Tic-Tac-Toe game ﬁrst (see P197, 5.9) and extend it to the planar n×n (n > 3 and n is odd) Tic-Tac-Toe game. You need to have a friendly graphical user interface to allow a human user to play the game with your algorithm. Alternatively, you may choose to apply your game-playing agent to a diﬀerent two-player game or multi-player game (e.g., P198, 5.11), if we can both agree upon it in advance.

**Objective:**

The project is designed to play a 2 player Tic Tac Toe game using minimax with alpha- beta pruning to a planar 3x3 game. We have programmed the whole project using Java and Java Swing Applet for GUI. When the game starts, the user is asked to select amongst the two options for playing:

1. Against Computer
2. Against another player

The minimax algorithm is implemented for player vs computer game wherein the player makes first move and then the algorithm computes the next best move for computer. The grid size can be selected from 3 and 4. Whenever the game starts it askes the Player to select the symbol from ‘X’ and ‘O’. There are three types of result Win, Lose and Draw. These are computed based on the state of the board that is if any of the symbol has occurrence in a line across or diagonal then the player wins. For the moves to be identified when playing we have played an audio clip. The figure 1 below shows the launch view of the program.

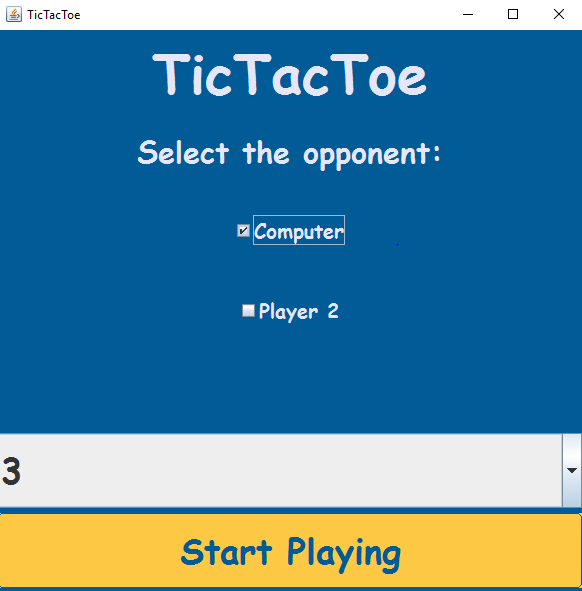


Figure 1

By default, Computer is selected as opponent and Grid size is 3.

The screen shot for playing game with 3x3 grid is shown below:

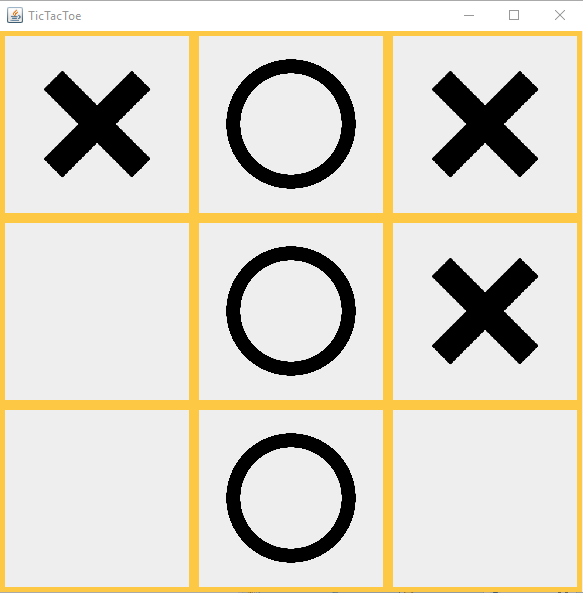


Figure 2

**Project Code Documentation:**

**Class Position:**

This class realizes the board state. The tic tac toe board is represented as a character array. Thus for dimension 3, the board is a character array of 9 characters.

Following are the relevant methods:

public static int getDim() : returns the dimension size selected from the homescreen user interface. This is used further to initialize the board’s character array.

public Position(): A constructor for Position class which takes in the string representation of board and the turn which may be ‘x’ or ‘o’.

public Position move(): Returns position object with specified move on the given index position.

public Position unmove(): Returns position object after unmoving the character on the given index j.

public String toString(): Overridden method for String representation of the character array of board.

public List<Integer> possibleMoves() : An array list of all possible moves. Checks for blank positions and add’s a move there and puts it in a list and thus returns a list of the possible moves.

public boolean isWinFor() : This method checks if it’s a win for the given turn which may by ‘x’ or ‘o’. It in turn uses lineMatch() static method to make the evaluations.

boolean lineMatch(): Method for matching rows, diagonals and columns.

public int code():Utility function for calculating value given the board configuration.

public int miniMax() : Calculating utilities using the popular mini max algorithm. Included hashmap for cutting off portions that need not be evaluated for speedy calculations.

public int blanks(): Returns the blanks given the board configuration.

public int bestMove() : Chooses the best move to be played for the given board configuration. Returns max of list from possible moves if turn is ‘x’ or min if turn is ‘o’.

public boolean isGameEnd() : Checks if the game has ended by evaluating win for ‘x’ or win for ‘y’ or draw.

public void setTurn(char turnp): Setting the turn appropriately.

**Class HomeScreen:**

This class implements the launch screen GUI of the program and interacts with the TictacToeUI class which includes the board arrangement and event handling.

public void createMainScreen(): Creates the launch screen with mouse event handling.

public String getGridSize(): Returns the grid size of the board.

public String getOpponent(): Returns the opponent.

**Class TicTacToeUI:**

This class consists of the board GUI and interacts with class Position to compute the next move using algorithm.

**Class AudioPlayer:**

This class implements the audio interface for the program. It is used to play audio file on moves made by player.

public void playAudio() : Plays the specified audio file for once.

**How to test the program?**

There can be following combinations for testing the program:

1. Select opponent as ‘Computer’, Grid size ‘3’ and Player 1 symbol ‘X’
2. Select opponent as ‘Computer’, Grid size ‘3’ and Player 1 symbol ‘O’
3. Select opponent as ‘Computer’, Grid size ‘4’ and Player 1 symbol ‘X’
4. Select opponent as ‘Computer’, Grid size ‘4’ and Player 1 symbol ‘O’
5. Select opponent as ‘Player 2’, Grid size ‘3’ and Player 1 symbol ‘X’
6. Select opponent as ‘Player 2’, Grid size ‘3’ and Player 1 symbol ‘X’
7. Select opponent as ‘Player 2’, Grid size ‘4’ and Player 1 symbol ‘O’
8. Select opponent as ‘Player 2’, Grid size ‘4’ and Player 1 symbol ‘O’

For all the above cases check if the result if a win for someone or a loss or a draw.

**Execution steps in screenshots:**

1. Launch application and select grid size



Figure 3

1. Click on Start Playing button and UI will pop up to select symbol.

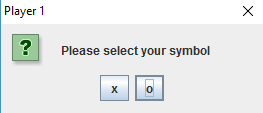


Figure 4

1. Figure shows 4x4 grid with O won.

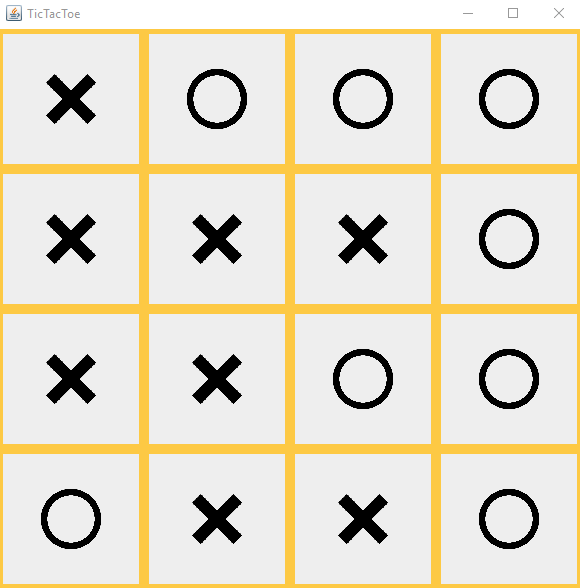


Figure 5

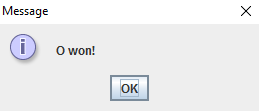


Figure 6

**Future improvements:**

* In future, the grid size can be extended to 5x5 or more with higher code efficiency.
* Networked TicTacToe can be one more advancement wherein two players can play the same game on different machines.

**References:**

1. Artificial Intelligence , A Modern Approach, 3rd Edition, by Stuart J. Russell and Peter Norvig, Prentice Hall, 2010
2. <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>