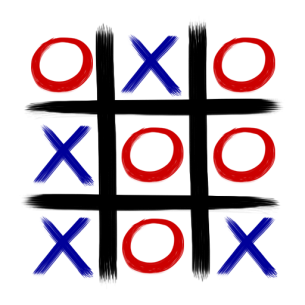
**Project Proposal**

**Project Title: TIC TAC TOE GAME**

**Course Name: ARTIFICIAL INTELLIGENCE**

**Project Description:**

**Introduction:**

Tic-Tac-Toe is a very simple two player game. So only two players can play at a time. This game is also known as Noughts and Crosses or Xs and Os game. One player plays with X and the other player plays with O. In this game we have a board consisting of a 3X3 grid. The number of grids may be increased.  
  
The Tic-Tac-Toe board looks like:

**Game Rules**

1. Traditionally the first player plays with "X". So you can decide who wants to go "X" and who wants go with "O".
2. Only one player can play at a time.
3. If any of the players have filled a square then the other player and the same player cannot override that square.
4. There are only two conditions that may be match will be draw or may be win.
5. The player that succeeds in placing three respective mark (X or O) in a horizontal, vertical or diagonal row wins the game.

**Winning condition**  
Whoever places three respective marks (X or O) horizontally vertically or diagonally will be the winner.

**Why Tic Tac Toe???**

Rules for tic tac toe game is simple enough that we don't need an elaborate analysis of game configurations. Despite being a simple game, the basic AI principles shown here can be applied to more complicated games such as checkers, Go and even chess.

**Functionality**

**•** How to play Tic-Tac-Toe game?

• Some guiding steps to play the game are: •

The game is played between two players

• Both players choose their respective symbols to mark

• Player 1 starts the turn by placing his/her symbol on any of the nine squares

• Then player 2 marks the turn by placing his/her symbol on the empty squares

• Both players make their turns alternately.

• If any player gets the three respective symbols in a horizontal, vertical or diagonal row wins the game

**Game Strategies:**

• Win: If you have two in a row, play the third to get three in a row.

• Block: If the opponent has two in a row, play the third to block them.

• Fork: Create an opportunity where you can win in two ways.

• Block Opponent's Fork:

• Option 1: Create two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork or winning. For example, if "X" has a corner, "O" has the centre, and "X" has the opposite corner as well, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)

• Option 2: If there is a configuration where the opponent can fork, block that fork.

• Centre: Play the centre.

• Opposite Corner: If the opponent is in the corner, play the opposite corner.

• Empty Corner: Play an empty corner.

• Empty Side: Play an empty side.

**USE OF ARTIFICIAL INTELLIGENCE:**

**Minimax algorithm**

MINIMAX is very clearly an application of depth-first search. It has a running time of O(b^n) where b is the maximum number of valid moves in a given turn and n is the maximum depth of the tree.

This all assumes that we are playing against a rational opponent. What if the opponent played sub-optimally? It isn’t too hard to see that MINIMAX cannot do worse in that case since the opponent will have made a move that does not bring us to a state where we have the minimum possible utility.

The astute reader will realize that this algorithm does not create an AI that always wins. It does however create an unbeatable tic-tac-toe computer player. That is, at best you will be able to tie. If like me, you are making your moves too quickly, you may even lose.

**Languages to be used:**

* Language Python version 3.0
* **Hardware requirements:**
* **Processor:** Intel dual core or above
* **Processor Speed:**1.0GHZ or above
  + **RAM:** 1 GB RAM or above
  + **Hard Disk:** 20 GB hard disk or above

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Student Name** | **Registration Code** |
| **1** | **M Sohail Zakir** | **02-131172-028** |
| **2** | **Ammara Riaz** | **02-131172-015** |