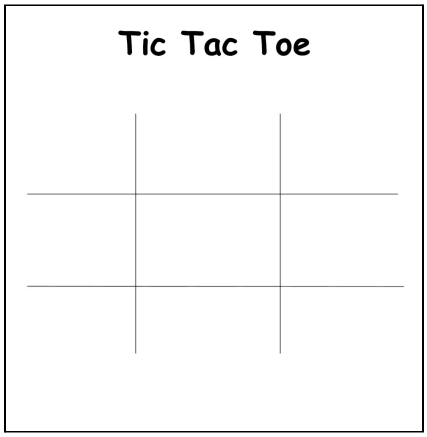
Assignment 0
Design of Solution for Tic Tac Toe

VJS Pranavasri
2020121001

I. Game Information

Tic-Tac-Toe is a 2 player game that can be simply played on any surface with a writable instrument. This game is usually played on a 3 x 3 Board, where players take turns and place their respective symbol (O or X) on one of the boxes. After certain conditions are met (as described in rules below), one of the players is declared a winner. Our aim is to create a code that plays against the player and always wins. This game can similarly be scaled to a N x N Board, but we will focus the design on a 3 x 3 Board.

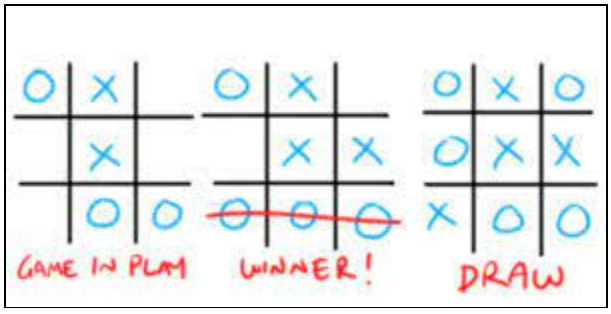


II. Rules

At the start of the game each player selects their respective symbols and one of them goes first, and each player takes alternative turns and places their respective symbol on the board in one of the empty boxes.

There can be 3 states in which a Board can exists

1. Game in Play
2. Game Won
3. Game Draw



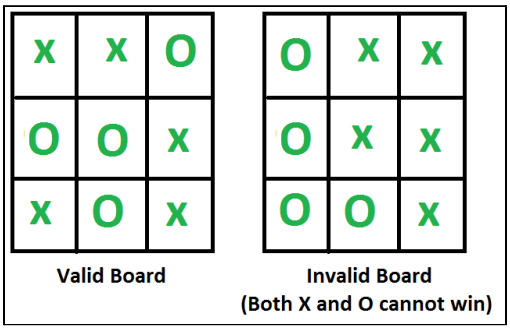
There are 3 cases in which a player is declared winner

1. If there are 3 (Xs/Os) in a row
2. If there are 3 (Xs/Os) in a column
3. If there are 3 (Xs/Os) in a diagonal

The game is declared over when:

1. one of the above mentioned criteria is satisfied, in that case the player whose symbol satisfies the criteria is the winner.
2. There are no more empty boxes on the board.

Note: Both X and Y cannot win at the same time.



III. Requirement of Product

We want to create a software(bot) that will play this game against a human. And we want our bot to always win the game no matter what. Logistically (at least for a 3 x 3 board), The possibilities are trivial, hence we will majorly focus on making sure that the human player doesn't win.

The workflow described below has to be followed to implement the same.

IV. States

There are multiple states that are possible

Board	X/Y	Game State
Empty Board	Playing	X Won
Playing Board	Waiting	Y Won
Completed Board		Draw

V. WorkFlow

Below is step by step information on implementing the given game.

1. Initially Start by taking user input on selecting the shape, and await the user's turn.
2. After Every of users turn as a bot you check the Board Status at that moment and check for two things:
    a. Check if the opponent has 2 in a row or column or diagonal, If Yes Block the third in the row, column or diagonal else do **b**.
    b.
        i. If you have two in a row, column or diagonal, and third is empty place your symbol there
        ii. If you have two in a row, column or diagonal, and third is non-empty, find if one of the adjacent of the two already placed have two empty spaces column wise, row wise or diagonal wise. If more than one are found place at a point which has the most feasibility*
        iii. If you have no coin on the board or the third spot of your previous two in a row is blocked, find a place that has three empty spaces column wise, row wise or diagonal wise. If more than one are found place at a point which has the most feasibility*

Feasibility: Being the most open i.e. If it has open spaces all sides (row column and diagonal), we can call it degrees of freedom, which defines how many movements are subsequently possible.

VI. Design Flow

This is a very stripped simple representation of the above Mentioned workflow.