Flow of Control Document

M Harini Saraswathy
Rishika Varma K

July 2020

1 Introduction

This describes the flow of control in the javascript code while playing against the computer.

2 Selecting Mode and Depth

On selecting 1 or 2 player on the site, the code automatically gets directed to either PvC or PvP mode. In both, odd numbered turns to play are by a person, and in the case of PvC mode alone, even numbered turns are taken by the computer, via the thought process explained in the next section.

On selecting single player, i.e. PvC mode, a dialog box appears requesting you to select a level of intelligence to play against. A higher number indicates a higher level of intelligence.

Selecting level 'x' will call a function which will assign value 'x' to the intelligence level/depth variable, "md", the effect of which is seen in the next section.

3 The Working of the Game

The game itself begins when you mark a square. This calls **onCheckBox**. This updates the box state as marked with the symbol of the current player and then disables it to avoid overwriting, via **checkElement**.

We check if the game has been won using **checkWinner** and hand control to the other player (computer/opponent) via **SwitchPlayer**.

If the mode is PvC and it is the computer's turn to play, we call **ComputerPlays**, which behaves as explained in the following section.

4 Computer Logic : 1 Player Mode

If the computer is in it's first move, **computeFirstMove** is called, which checks if our first move is an edge move, corner move or centre move, depending on which, an appropriate response is executed, via **EdgeMoveResponse** and an amateur level understanding of tic-tac-toe as a game. We have chosen not to implement the minimax algorithm in the first step alone as it takes a lot of time, and hence we hard-coded the results of the minimax algorithm.

If the computer is not on it's first move, we call **ComputeFinishingMove** which is effectively the necessary action for depth 1. In this, we see if, by 1 level of the minimax algorithm, it is possible to win the game, i.e. if there is a direct win.

ComputeSavingMove which is effectively the return value of evaluate function which gives value 0. This step is taken, when **ComputeFinishingMove** does not return any useful (corresponding to a positive evaluation) value.

If neither of the above functions proves useful, i.e. there is neither a direct win nor a direct loss in sight, then we call a function **findmove**, which, as it suggests, finds the best move for the computer to play. It does so by iterating through all the possible moves to be played and calculating the one with highest chance of winning via another function **minimax**.

minimax calls a function **evaluate** which behaves similar to the function **CalculateScore** but returns integral values of 1 (if there is an immediate win for current player), -1 (if there is an immediate win for opponent) or 0 (if neither of the above). If a non-zero value is obtained, then the minimax function terminates and returns said value minus the number of levels of action required to reach this state. This is done to prioritise earlier wins, and hence build a smarter computer. Else it iterates to another level of move along the thought process tree and, in the case of unlimited intelligence level, i.e. 5, recursively does so until the level number causes a non zero evaluation value. Else it only recurses until the intelligence level number of layers has been checked. If no useful result was found until this level, the computer is made to play a random move.

CheckWinner checks if the current player is the winner by identifying all the boxes marked by the current player and passing this list to **CalculateScore**, which in turn will return if the game is over or not, and display this using **ShowWinner** only in the case that isCheckOnly is false.

5 2 Player Mode

Here, there is not much to be implemented, as we merely log the moves of the 2 players and inform them if the game has ended. In the case that the game has ended, a dialog box appears displaying who has won or that the game has ended in a draw.