Stephen Green

CS 396

**Tic-Tac-Toe Game Report**

*Project Requirements:*

The main requirements for the games was to have a working tic-tac-toe game produced from the language of our choice. The game needed to be able to recognize and display the winner as well as having an option to go first or second. Lastly, the most difficult requirement was building a CPU that was unable to be beaten. The CPU had to be able to recognize when to make winning moves or block the opposing player from winning at every turn.

*Project Design and Algorithm:*

Language

I decided to design the game using HTML and JavaScript as I had a touch of previous experience with it prior and felt that it would be the easiest language to implement the board layout as well as designing click-based events.

Layout

To set up the game board and get the design right I decided to create a 3x3 table in HTML and with each section or tile I place an object from the class I created called “box”. By simply adding borders to all the boxes the HTML portion was pretty much complete. From there, I implemented Javascript to make the page interactive with the user. I created global variables of the number of moves, each box on the board, and the game status message to help all the methods coordinate with each other. Upon each user click the designated box will check the turn number and depending on odd or even it will pick X or O respectively to fill in the box. Following the fill, the turn count will be increased, and the game will check if there’s any winners and then lastly call the CPU to make its move.

CPU Algorithm:

In order to make the CPU unbeatable I incorporated a flowchart to cover just about every game state it could come across. To make the AI strategic I pre-prioritized its functions starting with “check for win,” “check for block,” play center, check for trap setups, play open corner, and lastly play open edge. With that flowchart it insured that the computer would always have a move to make and would never get stuck or frozen because it ran out of open tiles. To prioritize finding the win I created two separate methods (one for X’s and one for O’s) that checked for matching pairs to complete the three in a row. In my initial design I had one generic method that checked for any matching pairs but upon testing I discovered that the computer had no way prioritize ending the game by winning over just making another block. Checking for a win or block is the majority of anyone’s moves in Tic Tac Toe but the trickiest part about setting up the AI is telling it what to do during the first 2-3 turns where they usually isn’t any blocks or wins to make. The standard rule for all Tic Tac Toe players is if the middle is open, take it so that’s where I started with making the computer. The computer will always choose the middle if it’s open because in every situation it’s the best move. From there it got a little more complicated as the corners the next best choice but players can easily manipulate the computer by playing opposite corners to set up a double win scenario trap. The double corner trap is just one of the many traps I found through testing when the AI had the basic flowchart so that’s why I took the liberty to make a completely separate function called “avoidTrap” to give the computer coverage for more specific scenarios. Lastly if the Computer finds no moves to either win, block, go for center, or find a trap, it defaults to playing open corners and then lastly filling the open edges.

Other Methods:

Check Win – Checks for all 8 possible win conditions generically for both X and O combinations. Upon finding a winning condition the game will highlight the three winning boxes and then will change the game status message to the respective winner.

ClearBoard – Clears all the boxes and sets their text back to empty string, resets moves to one, and lastly highlights the winning boxes if any

Player First Button – Simply calls for the clear board method

CPU First Button – Calls for clear board method but also calls for computerMove right after

*Known Problems:*

Very rarely the CPU misses a game saving block when the board is full. It never has a problem when the board is just starting to fill up, but I have a suspicion that it gets confused whenever there a lot of tiles to check. I could never figure out exactly why it would do that, but for the most part it almost never comes up.

*References:*

W3Schools.com was very helpful resource for learning Javascript and CSS styling

Also looked up how to do board design by examining many other previously made tic tac toe games online

Lastly, fellow classmate Jeremy Wells was incredibly helpful in debugging my code upon finding errors or glitches