Tic-Tac-Toe

By Pronoma Banerjee

May 19, 2020

1 Objectives

The project describes the game of Tic-Tac-Toe, between the computer and a user. In our game we have used a 3 x 3 square grid. The game is over when the same symbol is obtained in all cells- either horizontally, vertically or diagonally.

The goal of the project is to design a Tic-Tac-Toe game using C language, in a manner that the user never wins, irrespective of who starts the game.

2 Theory of the game

The game is played in a 3x3 grid as shown.

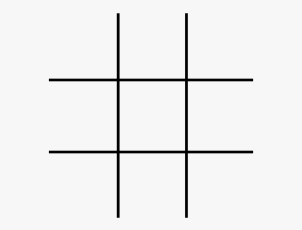

Figure 1: 3x3 grid for the game

The computer uses the symbol "O" and the user uses the symbol "X". The user marks any of the 3x3 squares with his symbol. The computer marks a square either at the first position available row-wise or based on the previous move of the user.

The goal of the computer is:

1. To insert it's symbol in all squares in a row, column or diagonal, before the user does so, to win.

2. To prevent the user from filling up all squares in a row, column or diagonal.

Hence, in the game designed, there are only 2 possibilities - the computer wins or the game ends in a tie.

1 1	1 2	1 3
2 1	2 2	2 3
3 1	3 2	3 3

Table 1: Grid with row and column numbers of cells

If the computer is able to draw three Os in the following combinations, then it wins. Else the game ends in a draw. The combinations are (row column):

a. 1 1, 1 2, 1 3

b. 2 1, 2 2, 2 3

c. 3 1, 3 2, 3 3

d. 1 1, 2 1, 3 1

e. 1 2, 2 2, 3 2

f. 1 3, 2 3, 3 3

g. 1 1, 2 2, 3 3

h. 1 3, 2 2, 3 1

3 Moves

The game has 2 different perspectives- when computer starts the game, and when the user starts the game. The moves have been designed in a way such that the computer has a high chance of winning, and in case of smart moves from the user, can always force the game to a draw.

3.1 Central Logic

The program uses a common logic for both perspectives. At each round of the computer:

1. It places itself in the empty cells, one by one, and checks whether being in that cell makes it win in that round.

2. If such a cell is found, the cell's position is marked as the best position and the computer places its move there.

3. If such a cell is not found, it places the user's move in the empty cells, again one by one, and checks if that makes the user win.

4. If such a cell is found, the cell's position is marked as the best position (since the move should be placed there to block the user from winning).

5. If the computer does find any cell (best position) that makes it win or blocks the user from winning in that round,

a. it either uses a strategy based on who started the game, or

b. places its next move in the first available cell, moving row-wise.

3.2 Computer starts first

3.2.1 First move

Computer plays its first move in the left-top corner (row 1, column 1). From observed strategies of the game, the computer can almost ensure winning the game by this move, if the user places its first move anywhere other than the centre.

3.2.2 Second move

1. If the user places its first move in the centre, the computer plays its next move in the corner opposite to its first move (row 3, column 3). This is a winning strategy, as will be evident in the third move.
2. If the user places its first move anywhere other than the centre, the computer can place its first move in the next available position, and use the central logic throughout the rest of the game.

3.2.3 Third move

1. The computer first uses the central logic to find a cell that makes it win or blocks the user from winning.
2. If no such cell is found:

a. If the computer places its first move in 2 opposite corners (as when the first move of the user is in the centre, it places its next move in one other corner, thus ensuring that it wins in the next round.

b. If this is not the case, the computer simply places its next move in the first available cell.

If the computer wins in this round, game ends.

3.2.4 Following moves

Following moves are based on the central logic of winning or blocking the user from winning. Game ends whenever the computer wins. If all cells are occupied before the computer wins, the game ends in a draw.

3.3 User starts first

3.3.1 First move

1. If the user starts in the centre, computer simply uses the central logic throughout the game starting with the first corner.

2. If the user starts anywhere other than the centre, the first move is at the centre. This is the only way the computer has a chance to win when the user starts the

game (only when user's first move is at an edge), unless the user makes a mistake.

3.3.2 Following moves

The following moves are based on the central logic used in the game. Game ends when the computer wins. If there is no chance of the computer winning, the logic used forces the game to end in a draw.

4 The C program

4.1 Methods

The methods used in the c program are as follows:

1. **void showboard(char board[][SIDE])**: to show the status of the board after each move, to see the positions of each symbol already placed.

2. **void showInstructions()**: to show the instructions of the game each time before the game starts.

3. **void declareWinner(int whoseTurn)**: to declare the winner of the game. Unless there is a logical error in the code written, the method should always declare

the computer as winner, whenever it is called.

4. **int rowCrossed(char board[][SIDE])**: to check if the same symbol is present in all squares across a row.

5. **int columnCrossed(char board[][SIDE])**: to check if the same symbol is present in all squares down a column.

6. **int diagonalCrossed(char board[][SIDE])**: to check if the same symbol is present in all squares along a diagonal.

7. **int gameOver(char board[][SIDE])**: to check if game is Over before all cells are filled. The computer wins if the method returns 0. Iff the method always returns 1, then the game ends in a draw.

8. **void initialise(char board[][SIDE])**: to set all cells to blank space at the start of each game.

9. **void winMove(char board[][SIDE])**: to implement the winning strategy described in the 'Moves' section, when the Computer starts the game and when the user's first move is at the centre.

10. **void nextMove(int moveIndex, char board[][SIDE])**: to decide and implement each move of the computer.

11. **void usersMove(char board[][SIDE])**: to take the user's move in each round.

12. **void playTicTacToe(int whoseTurn)**: to implement the main execution the game of Tic-Tac-Toe, by calling all the other functions whenever required in the game.

13. **int main()**: the driver function which asks the user if he would start the game, and calls the playTicTacToe function to start the game.

5 Limitations

1. The code written is restricted to the 3 x 3 grid and cannot be applied to a general n x n system.

2. The user cannot enter its symbol directly in a cell. Instead he has to mention the row and column numbers.

3. Due to the limitations of C language, any new strategy used has to be explicitly written down in the code, due to which only few strategies could be implemented in the game.

6 References

1. Akshay L Aradhya- "Minimax Algorithm in Game theory (Tic-Tac-Toe AI-Finding optimal move)", Geeks for Geeks, https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/

2. Rachit Belwariar- "Implementation of Tic-Tac-Toe game", Geeks for Geeks, https://www.geeksforgeeks.org/implementation-of-tic-tac-toe-game/

3. "How to Win at Tic Tac Toe", wikiHow, (updated) 9 May 2020, (128 co-authors) https://www.wikihow.com/Win-at-Tic-Tac-Toe