A Project on crisscross

Submitted by,

Candidate Name: Vakacharla N V S Chaitanya Kumar

Ps no: 99006188

Contents

1. Overview

1.1. Problem statement

- To create a criss-cross game which is to be played by one and the opponent would be the computer/bot.

1.2. Description

Criss-cross is a two-player alternate turn-based game where, first player uses X as marking on a 3-by-3 game board and the second player uses O. The goal is to make three of your marks in a row while blocking and thus preventing your opponent from doing the same. The three in a row marking can be done horizontally, vertically or diagonally

2. Requirements

2.1. Software Requirements

- MinGW compiler
- Visual Studio Code

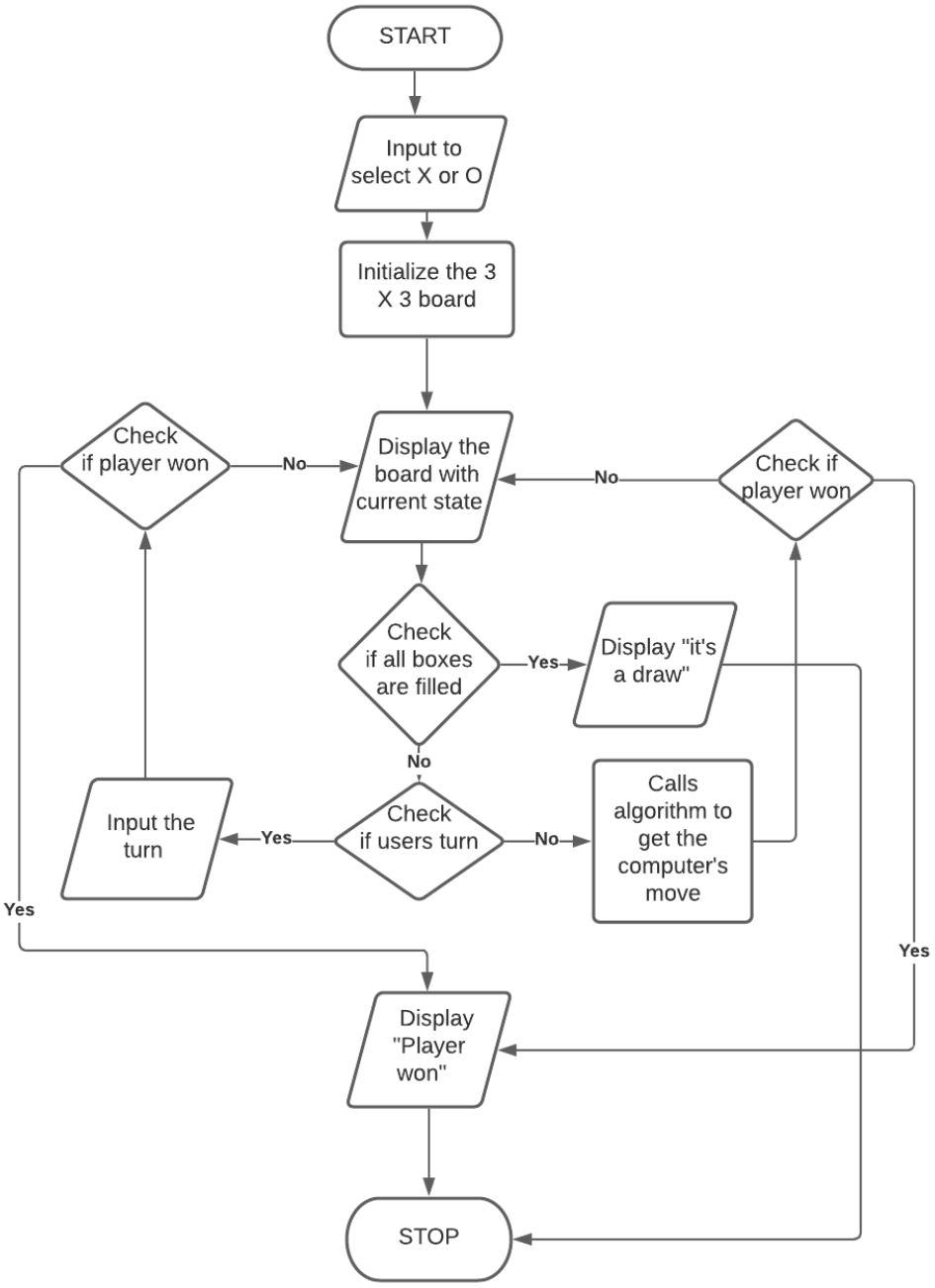
2.2. Functional Requirements

- The program shall be able to display a 3 X 3 matrix to imply the board.
- Display the board prior to every move and at the end of the game.
- The user should be able to choose which marking, X or O, as in the first or second player to start the game. Computer becomes the alternate one.
- The user can input there turns via numbers of array from 0 to 8.
- The computer shall be able to input moves which are counter intuitive to users' inputs rather than random moves and hence increasing its winning chance.
- As per the game's rules whoever is first to input three in a row marking is showed as winner and the game ends.
- If all the boxes are filled and no three in a row marking are there, then the game is draw output should be displayed.

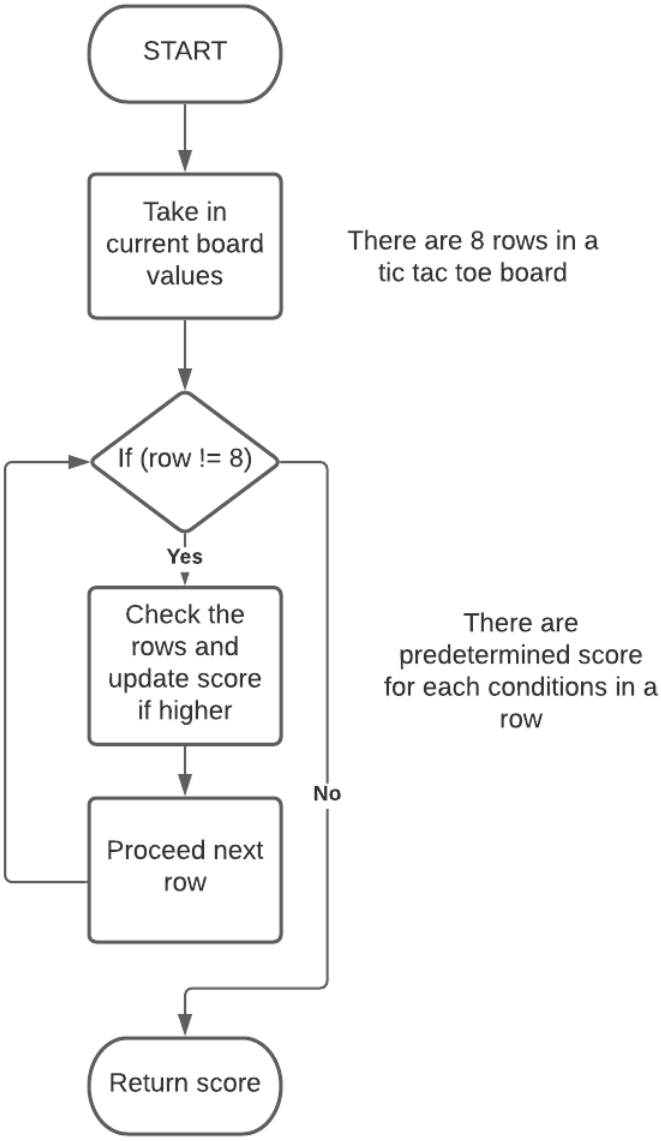
3. Design

3.1. Flow Diagrams

3.1.1. Main Program



3.1.2. Computer/Bot Program



3.2. Explanation

The criss-cross has two players, where the player who starts from 2nd turn will have a disadvantage. So, at first the user gets to choose which player, as in X and O, the user wants to be. Then a 3 by 3 board is virtualized and printed, where from top first cell to last row 3rd cell is counted as 0 to 8 to depict its location. So, the user can give input as numbers to interpret where the user wants their token should be placed for each turn. After each turn from both players, the board within its current state is printed for the user.

For the game to terminate, it has three states for the user, namely- “win”, “lose” or a “draw”. For either of the user or computer/bot to win, they must meet the condition where three tokens of the same player should be in a row. In Tic-Tac-Toe board, there are 8 combinations of rows, any of which can be attained to reach the “win” state. If the computer/bot is able to reach the win state first, then the user will be declared to be in “lose” state. And if both the players can’t reach the “win” before the 9 turns are over, hence populating the whole board without three in a row, they game will be declared as “draw” state. Thus, terminating the game.

Now, after each user’s turn, a programmed bot will be competing against the user. The bot gives the output not as in a random output, but as in the target of achieving the “win” state and therefore also preventing the user to get to “win” state by also producing outputs according to the user’s turns. The bot will be functioning by finding the best move for its progression to the “win” state, and its is calculated via Minimax algorithm. So, its calculated by scanning every row in the current state of the board after each turns. And then for each row a score is given according to these conditions: -

- If there are both X and O in the scanning row, then the score for that row is 0
- If the whole row is empty, then the score is 1
- If there is only one token of the computer/bot on that row, then the score is 10
- If there are two tokens of the computer/bot on that row, then the score is 100
- If there are three tokens of the computer/bot on the row, then the score is 1000, and the winner will be the bot.

After each turn the bot computes the score for each row, and then selects the row with highest score.

4. Test Plan

4.1. Introduction

The test plan is designed to identify the items to be tested, the features to tested, type of testing to be done and to define the test criteria for the criss-cross game.

4.2. Test Strategy

4.2.1. Scope of Testing

4.2.1.1. Features to be tested

Module	Description
view_Board	The program should be able to display out a 3 by 3 matrix at first, and after every turn with current state
X_or_O	At start program should be able to give user a choice of token, X/O, and continue with it for the whole game session
Board[9]	The program should be able to take inputs from user as 0 to 9 interpreting to the locations of the board
Bot	After each turn as user input, an output should be created as a second player/opponent for against the user.
Best_score	The bot should be able to create an output which is increases the chance of its winning and decrease the chance of user to win.
Win_state	If the user reaches a three in a row condition, program should declare it as a win state and terminate the game.
Lose_state	If the bot reaches a three in a row condition before the user, program should declare it as a lose state and terminate the game.
Draw_state	If all the 9 turns are over with neither of the players reaching three in a row condition, the program should declare it as a draw state and terminate the game.
Check_state	The program should be able to verify after every turn that is either of the player has achieved the states, win, lose or draw.

4.2.1.2. Features not to be tested

The features that are not to be tested are:

- User interfaces
- Hardware interfaces

4.2.2. Test Types

The test plan is designed to identify the items to be tested, the features to tested, type of testing to be done and to define the test criteria for the Tic-Tac-Toe game.

Three types of testing are to be done for the Tic-Tac-Toe game:

- **Unit Testing**: In this testing each individual module of the system is tested to ensure that they are fit to use.
- **Integration Testing**: Here the modules are integrated together and tested as a group.
- **System Testing:** Conducted on a complete, integrated system to evaluate systems compliance with its specific requirements

4.3. Test Objective

The test objectives are to verify the functionality of the Tic-Tac-Toe game. The focus should be on testing the functionalities such as Board, userToken_input, bot, Best_score, Win_state,…etc to guarantee that these operations can work normally.

4.4. Test Criteria

4.4.1. Suspension Criteria

If more than 40% of the test cases fail, the test is suspended until all the failed cases are fixed.

4.4.2. Exit Criteria

Specifies the criteria that denotes a successful completion of a test phase

- Run rate is mandatory to be 100% until a clear reason is specified
- Pass rate is 90%

4.5. Test Environment

All the tests are carried out in the CodeBlocks IDE.

5. Test Cases

Test Case ID	Test Scenario	Test Steps	Test Data
T01	Check user input being X token	1. Run program 2. Enter first token option	1) Token = 1
T02	Check user input being O token	1. Run program 2. Enter second token option	1) Token = 2
T03	Check board output for current state	1. Run Program 2. Enter Token 3. Enter a 0 to 8 location in board	1) Token = 1 2) Move = 3
T04	Check for invalid response for giving same move twice.	1. Run Program 2. Enter Token 3. Enter a 0 to 8 location in board 4. Enter the same location	1) Token = 1 2) Move = 3 3) Move = 3
T05	Check for bot response in a three in row chance for user in next turn for diagonal rows	1. Run Program 2. Enter Token 3. Enter a location which is in diagonal row 4. Enter a location from the same row of previous entry	1) Token = 1 2) Move = 4 3) Move = 8
T06	Check for bot response in a three in row chance	1. Run Program 2. Enter Token	1) Token = 1 2) Move = 1

	for user in next turn for Vertical rows	3. Enter a location which is in Vertical row 4. Enter a location from the same row of previous entry	3) Move = 7
T07	Check for bot response in a three in row chance for user in next turn for Horizontal rows	1. Run Program 2. Enter Token 3. Enter a location which is in horizontal row 4. Enter a location from the same row of previous entry	1) Token = 1 2) Move = 8 3) Move = 7
T08	Check for program response for three in a row for user	1. Run Program with bot function not called 2. Enter token 3. Enter Three location of the same row	1) Token = 1 2) Move = 3 3) Move = 4 4) Move = 6

T09	Check for program response for three in a row for the bot	1. Run program 2. Enter token 3. Enter three location from corners of the board	1) Token = 1 2) Move = 0 3) Move = 2 4) Move = 8
T10	Check for program response for all 9 turns being exhausted without a three in a row condition	1. Run program without calling the bot function 2. Enter token 3. Enter all the nine location	1) Token = 1 and 2 2) Move = 0 3) Move = 1 4) Move = 2 5) Move = 3 6) Move = 4 7) Move = 5 8) Move = 6 9) Move = 7 10) Move = 08

6. Expected Result

Test Case ID	Test Scenario	Expected Results
T01	Check user input being X token	User input should be token X for the rest of the game
T02	Check user input being O token	User input should be token O for the rest of the game.
T03	Check board output for current state	Program should output a board of 3 X 3 matrix size.

T04	Check for invalid response for giving same move twice	Program should loop until a different location which isn't used in past turns
T05	Check for bot response in a three in row chance for user in next turn for diagonal rows	Program bot should give a token on the location 0.
T06	Check for bot response in a three in row chance for user in next turn for Vertical rows	Program bot should give a token on the location 4.
T07	Check for bot response in a three in row chance for user in next turn for Horizontal rows	Program bot should give a token on the location 6.
T08	Check for program response for three in a row for user	Program should give an output of "Win" and then terminate the game
T09	Check for program response for three in a row for the bot	Program should give an output of "Lost" and then terminate the game
T10	Check for program response for all 9 turns being exhausted without a three in a row condition	Program should give an output of "Draw" and then terminate the game