TECHNICAL DOCUMENT

THE MARS COLONIZATION PROGRAM

Project 2: ENTERTAIN THE CREW

TIC TAC TOE A Web-Application based game

DEVELOPMENT & IMPLEMENTATION COLLABORATION BETWEEN:
Microsoft Corporation
MnMCoders

Developed by:
Maitrayee Nakade
Nikita Sadhnani
Manasi Gude

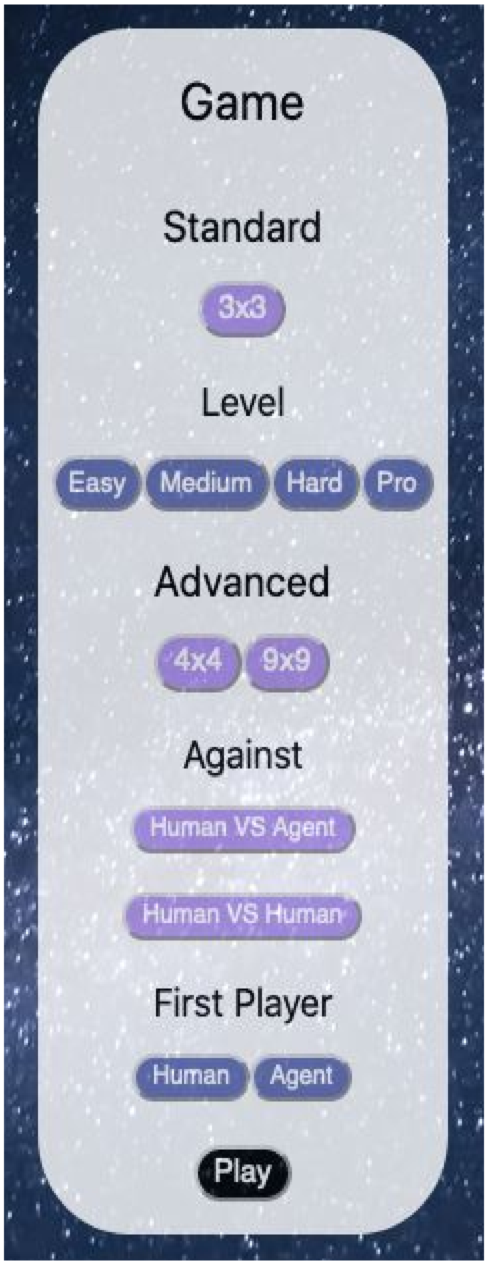
July 23, 2020

Overview

Engage 2020 is a Virtual Mentorship Program organised by Microsoft to establish a permanent human settlement on Mars. As a part of the program, Team - MnMcoders built an unbeatable Tic-Tac-Toe game powered by Artificial Intelligence to entertain the crew.

INDEX

USER GUIDE



Follow these 4 Steps to Play the Game

Step - 1: Select a Game

(i) To play Tic Tac Toe on a 3 X 3 board , click on 3x3

and select one of the levels - Easy Medium Hard Pro

(ii) To play Tic Tac Toe on 4 x 4 board, click on 4x4

(iii) To play Tic Tac Toe on 4 x 4 board, click on 9x9

Step - 2: Select your opponent

(i) If you want to *play against Agent* - Click on Human VS Agent

(ii) If you want to *play against Human* - Click on Human VS Human

Step - 3: Select starting player

-> Applicable when the opponent is an agent. Human Agent

Step - 4: Start the game

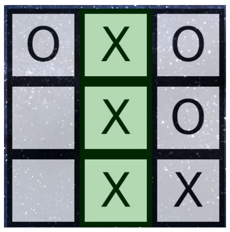
-> Click on - Play

Rules of the game -

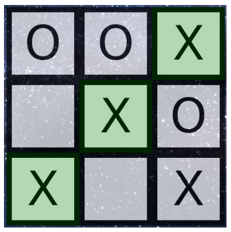
Starting Player's Move is always 'X'.

1. Rules for 3 x 3 and 4 x 4

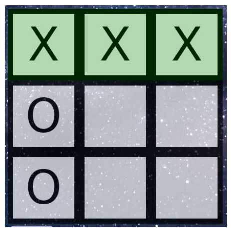
- Click on any cell to make your move.
- Incase you want to change your move, click on Undo
- If you are stuck, click on Hint to get a hint for the best next move. (The recommended square is highlighted in yellow)
- Both the players will alternately make their moves until one of the terminal states is reached .
- **Terminal States -**
    - For **3 x 3:**
        - If the board is full and none of the players have won then the game is a draw
        - To win, a player should have made moves in 3 consecutive cells either vertically,horizontally or diagonally.



VERTICAL

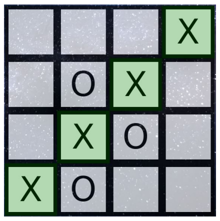


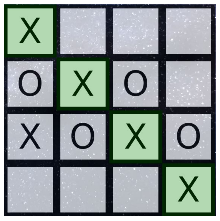
DIAGONAL

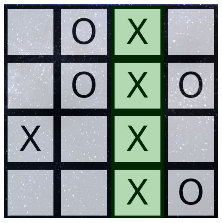


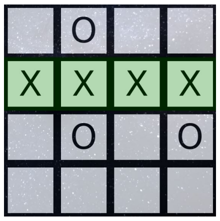
HORIZONTAL

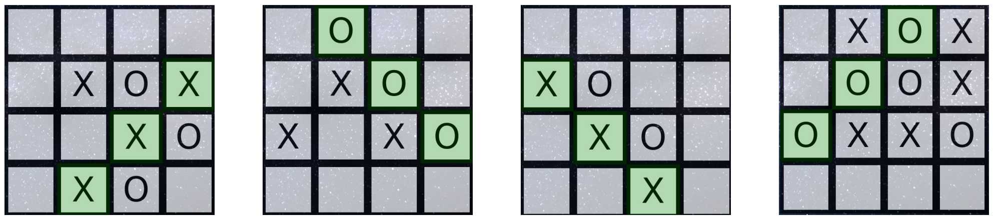
    - For **4 x 4**:
        - If the board is full and none of the players have won then the game is a draw.
        - To win a player should have his/her move in -
            - 4 consecutive cells either vertically, horizontally or diagonally.
            - 3 consecutive cells along the sub diagonals.


Right Diagonal


Left-Diagonal

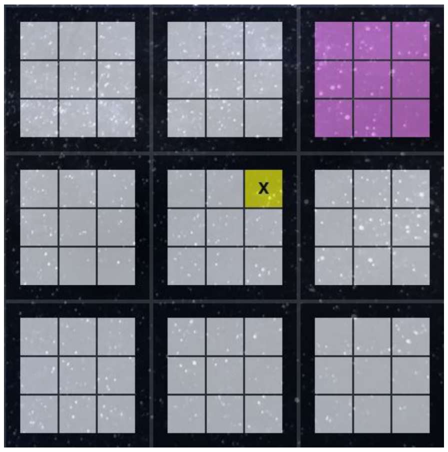

Vertical


Horizontal

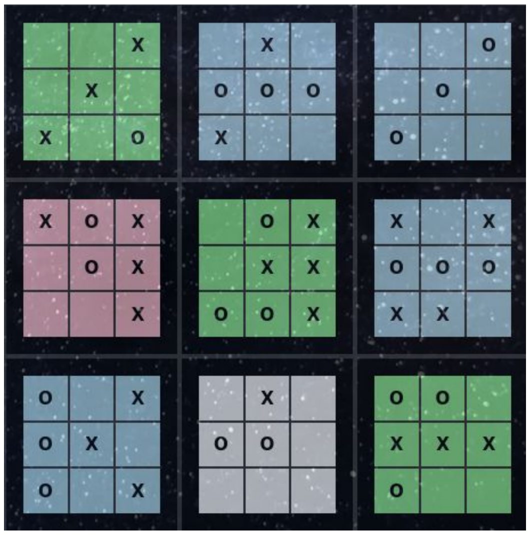

Sub-Diagonals

2.Rules for 9 x 9:

- There is a global 3 x 3 board where each cell comprises a local 3x3 board.
- Click on one of the small squares to make your move.
    - The first player can play on any arbitrary small board.
    - Following this, which smaller board a player can make move in is determined by his opponents move.
    - Whichever small square the opponent plays in ,the position of the next small board relative to the big board is similar to the position of the small square relative to the small board in which it is present.
    - If the small board that a player is directed to is in the terminal state then the player can make a move in any available small board.
    - The next small board where the player has to make a move will be highlighted in **purple** *(As shown in figure below)*. If no small board is highlighted then the player can choose any non-terminal arbitrary small board.



- Rules for the small board are similar to that of the 3 x 3 game mentioned above.
- Both the players will alternately make their moves util one of the terminal states is reached
- Terminal State :
    - Win small boards according to the rules in the 3 x 3 game.
    - To win , a player needs to win 3 consecutive small games in a row, column or along a diagonal.
    - If **'X'** wins a small board, it will be colored in **Pink**.
    - If **'O'** wins a small board, it will be colored in **Blue**.
    - If any player **wins** the entire game, the winning boards will be highlighted in **Green**.



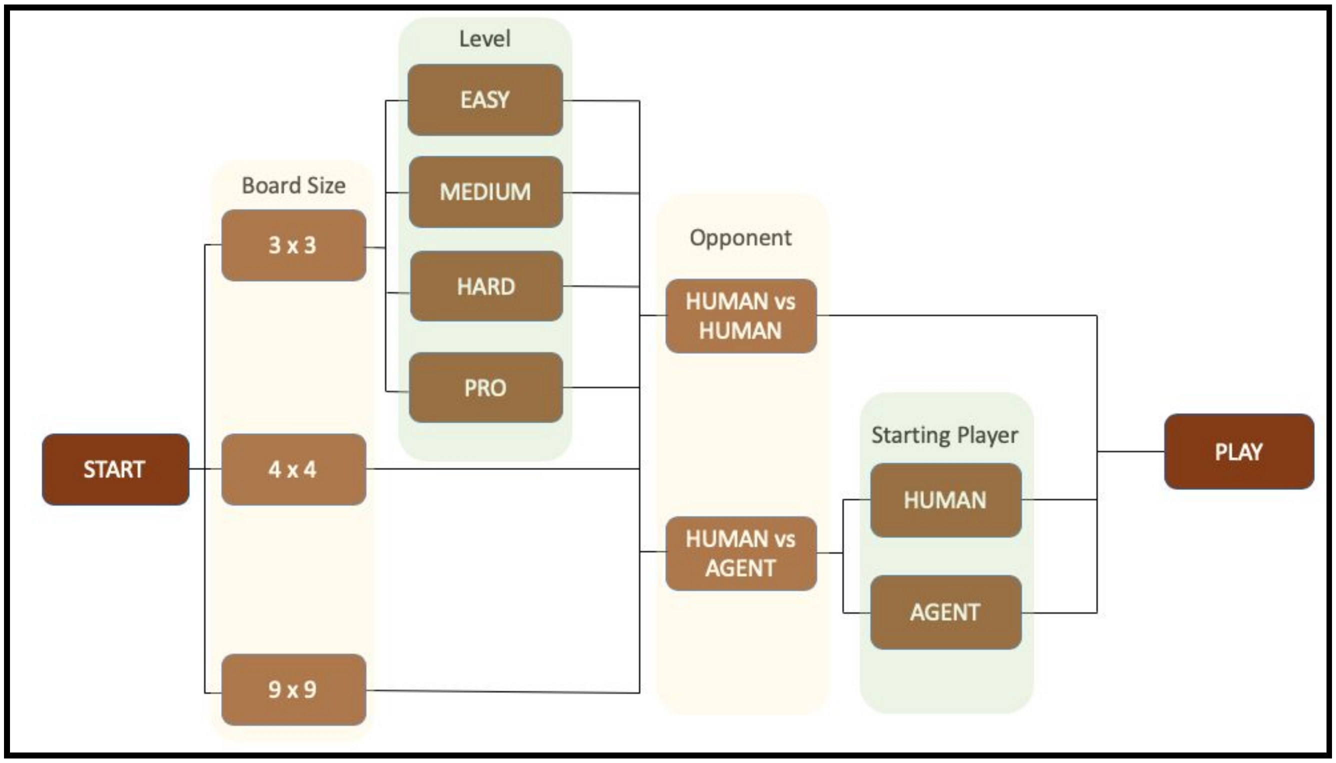
Why Angular?

Specification: Angular 9.1.9 [also used Bootstrap 4.5.0]

Angular is a Client-side framework that allows developers to develop advanced *Single-Page web applications*. Angular provides a lot of built-in features like state management, components, services, Bi-directional data binding and routing. *Component*s, modules and services make it easier to maintain code for larger applications as it provides a better structure.

Angular framework compatible with running on all browser environments regardless of the OS and architecture. It is written using *TypeScrip*t language (superset for JavaScript). It fully compiles to JavaScript, but helps find and eliminate common mistakes while typing the code. Further, Angular has assurity of long term support from Google.

Flowchart for this Angular based single page web application game



Component Interaction

Introduction

Components are the basic blocks of Angular. They control the views of the app screen. Individual Components can be altered to change specific parts of the screen without altering the whole code.

Components in our Project

In our project, we have used 7 components apart from the parent app-component. The app-component can communicate all its children components and can regulate the flow and ordering of them. The 7 components are -

a) Cell Component - Used for creating a view of the 3 x 3 and 4 x 4 boards. It helps in reducing the repetition of code in both the components.

b) Four-XFour Component - Used for carrying out the whole 4 x 4 game.

c) LeftBar Component - Used as a menu for users to select the game play options. Resides in the left side of the screen.

d) Nine-XNine Component - Used for carrying the code of the whole 9 x 9 game.

e) Rules Component - Used for displaying the game rules in the start and in between if no game is started. Located in the center of the screen.

f) Three-Easy Component - Used for handling all the levels of the 3 x 3 game.

g) Title Component - Located at the top for displaying the title of the game.

Components Data Flow

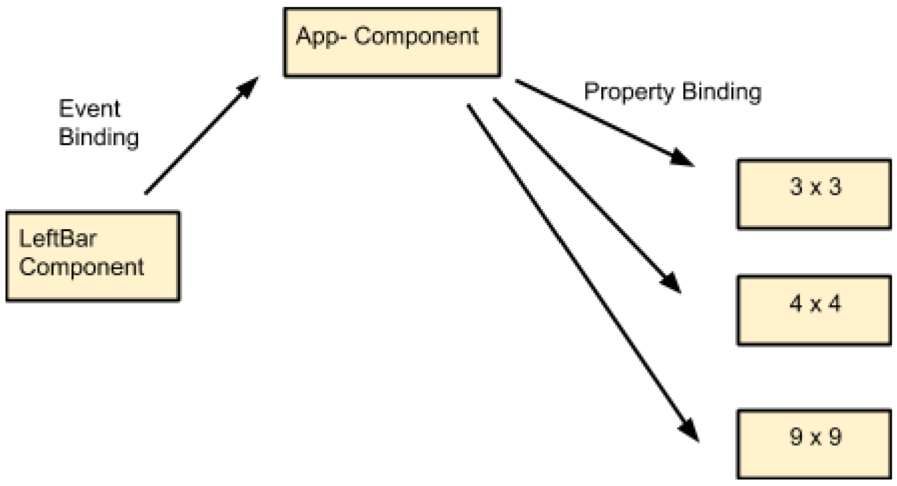
We have used different techniques of angular components to improve the flow of the data through various components. This includes Property Binding, Event Binding and Interpolation. Data binding plays an important role in communication between a template and its component, and is also important for communication between parent and child components.

Components Benefits in Performance

We have made three different components for different games which improves the readability and maintainability of the code. There were differences in the terminal states which led to different components in 3 x 3 and 4 x 4 so that any one component will not be cumbersome to read. This increased the performance of the game and helped us identify various bugs, which we could fix promptly.

Navigation

Since the buttons are in the LeftBar Component and games are in their respective components, we needed navigation techniques to implement it smoothly. We used Data binding techniques and passed the data from child LeftBar Component to parent app-Component, and then passed it from parent app-Component to respective games - child components which helped in functioning of one game at a time. The leftBar Component passes three properties, GameType, PlayerType and OpponentPlayer name on clicking the respective buttons. These are then passed as data to child components from app-Component which are then used to perform necessary actions in respective games.



Algorithms used by AI agent

Minimax

Minimax algorithm is a decision making algorithm used to find the optimal next move of a player in a zero-sum, perfect information game.It assumes that the opponent plays optimally.

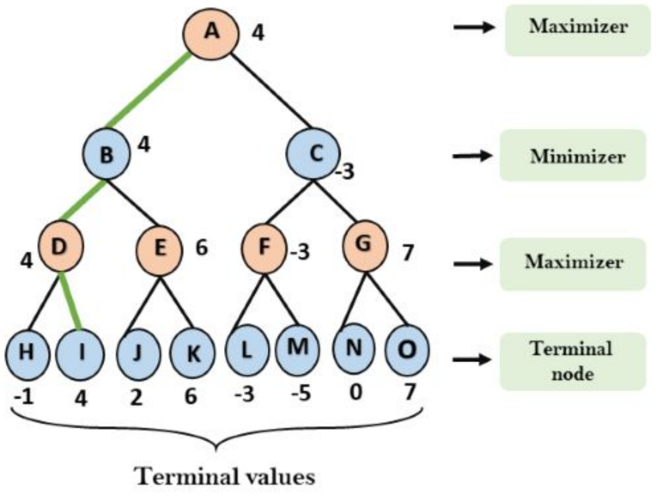
The minimax algorithm consists of

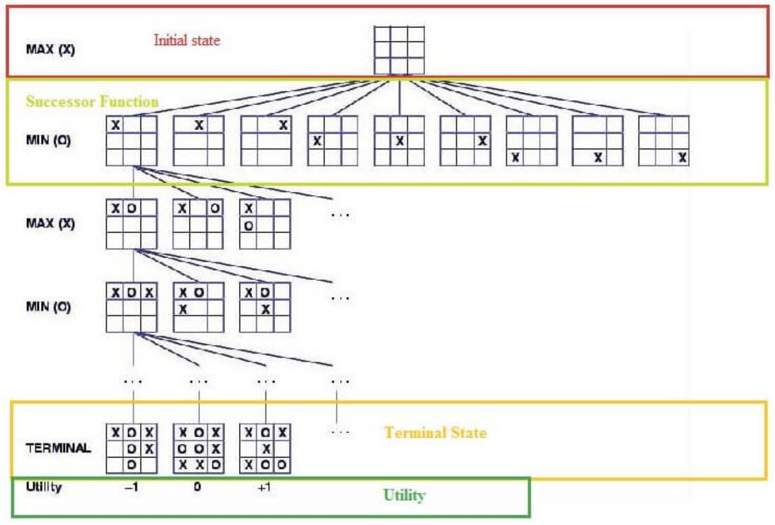
1. Game Tree: is a structure in the form of a tree consisting of all the possible moves which allow you to move from a state of the game to the next state.
2. Initial State: It comprises the position of the board and showing whose move it is.
3. Terminal States: It is the position of the board when the game gets over.
4. Utility Function: It is a function which assigns a numeric value for the outcome of a game.

The game has two players, the maximizer and the minimizer. The maximizer tries to get the highest possible score and the minimizer tries to get the lowest possible score.

The process of the minimax algorithm is as follows:

1. Generate the game tree from the current state of the game all the way to the terminal states.
2. The utility function is applied to get the utility values for all the terminal states.
3. The utility of the higher nodes is calculated using the values of the terminal nodes as follows:
    a. If the node represents move of the maximizer, then the child node having the maximum utility value is selected.
    b. If the node represents move of the minimizer, then the child node having the minimum utility value is selected.
4. Eventually, all the backed-up values reach to the root of the tree.At that point, the maximizer has to choose the child node with the highest value.
5. Hence the optimal move for the maximizer would be the move corresponding to the node with maximum utility value.





Why Minimax?

The minimax algorithm carries out a depth first search on the game tree. It calculates all the outcomes till the end and then chooses the best move for the agent.It always gives the best move and hence produces an Artificially intelligent agent that is unbeatable.

The time complexity of minimax is O(b^h) where b is the branching factor and h is the maximum depth of the game tree and the space complexity is O(h) similar to the DFS algorithm.

This algorithm is optimal in games where the branching factor is low.

In 3x3 and 4x4 Tic Tac Toe games, the maximum branching factor will be 9 and 16 respectively and hence the minimax algorithm is the best algorithm to decide the agent’s optimal move.

This algorithm is not optimal to implement the 9 x 9 Tic Tac Toe game due to the high branching factor of the game.

Optimization

Alpha Beta Pruning

When alpha-beta pruning is applied to a standard minimax algorithm, it returns the same move as the standard one, but it removes (prunes) all the nodes that are possibly not affecting the final decision.

1. Alpha is the best move so far for the maximizer and should be the highest possible value.
2. Beta is the best move so far for the minimizer and should be the lowest possible value.
3. Each node has to keep a record of its alpha and beta values.
4. Alpha is updated only at the maximizer’s turn and Beta is updated only at the minimizer’s turn.
5. Alpha is initialized to -Infinity and Beta is initialized to +Infinity. Pruning is done only when alpha >= beta at that node.
6. While backtracking only the node’s values are passed to the upper nodes while alpha and beta values are passed to the child nodes.

Additional Features

1. Hints

The user can ask for hints by clicking on the hints button. The cell will be highlighted in yellow. This feature uses the minimax algorithm to find the best next move for the Human. It is implemented considering that the human will be the maximizing player.

2. Undo

As the game progresses the moves of the players are stored in a stack. When the user clicks on undo his move is popped from the stack and deleted from the board. In the case of Human vs Computer two moves, that is the last move of the human as well as the computer.

Monte Carlo Tree Search

About MCTS

Monte Carlo Tree Search is a probabilistic search algorithm used for making decisions while solving the game tree. It exploits strategies to find the best move and keep on exploring for future moves. It expands the search tree based on the basis of random sampling.

MCTS Working

The Monte Carlo Tree Search consists of four main functions namely - Selection, Expansion, Simulation and Backpropagation.

(i) Selection - This function is used to select the best possible node for future exploitation. The best child is selected on the basis of a UCT(Upper Confidence Bound in Trees)value. This helps in developing future moves.

- UCT value : This is based on the number of times a node is visited and corresponding parent of that node along with the winning score at that point of play.

(ii) Expansion - Based on the selected node, the expansion function adds available child nodes to the tree. Thus helps to expand the tree for the next level.

(iii) Simulation - This function is used to carry out random game play on available spots and check the winner of the game. This is basically trying out every possible move from that point of game.

(iv) Backpropagation - This function helps in updating the values of win score of winning child nodes in the simulation function. Also it increases the visit count of nodes which helps in selecting future nodes. It will update the values of all nodes until the root is reached.

Why MCTS?

Monte Carlo Tree Search is used in many board games. This helps in determining best possible moves for the machine. Since the Ultimate TicTacToe game consisting of a 9 x 9 board cannot be solved optimally in a given time limit using the Minimax Algorithm, we decided to apply Monte Carlo Tree Search Algorithm for the game and improve the decision-making and performance of the game.

MCTS in the 9 x 9

In the Ultimate TicTacToe game, opponents have to play in the board corresponding to the board cell of the move of their opponent. This is a strategic game and requires best moves in every small board.

Implementation in the 9 x 9 Game:

a. After every human move, Monte Carlo Tree Search function is provided with the current global board of the game. The machine has to choose any empty available spot in the board corresponding to the cell of the human's move, optimally. If the current board is full or won by the human, the machine will select any random small board and make its move in the available spot there.

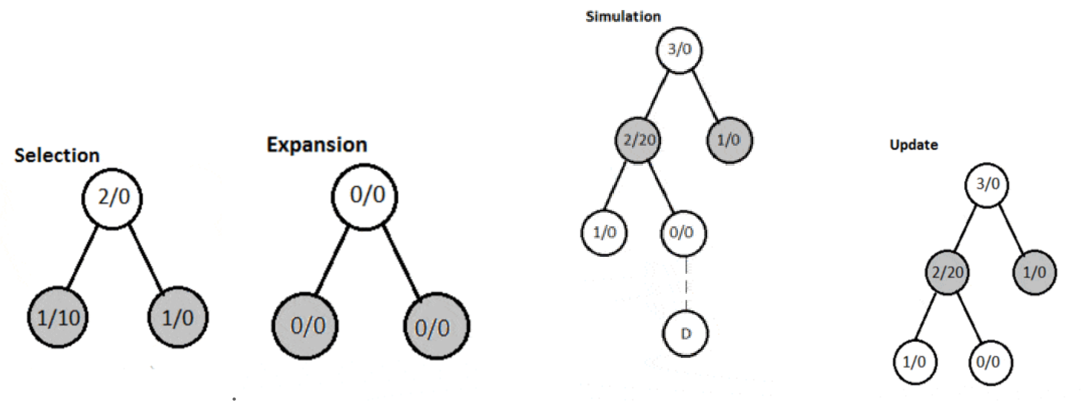
b. The current move of the human is made the root of the search tree each time the function is called. The available spots in the corresponding small board where the machine has to make its move, are the children nodes of the root.

c. Selection function is called on the root node. This will select the best child out of the children nodes using the UCT value. At the start, when the number of visits are zero, UCT value is set to be the maximum integer value available.

d. After selecting the best Node, it is sent to the expansion function to get all the available spots to make a move from the selected node.

e. After expansion is done, the node is sent to the simulation function to carry out the required gameplay until terminal conditions are reached. Terminal conditions are either win or draw. If in the middle, the machine is “sent” to a filled or terminal condition board, it chooses the next board randomly.

f. The update function updates the winsocres and number of visits hereafter. The best node which is also the best move having the most winscore is finally chosen to be the move played by the machine.These four functions are iterated many number of times to get the simulation values.



Performance benefits of MCTS

Since the iterations are random and the playout is made on statistical basis, performance increases significantly as the machine focuses on the nodes which yields wins.

This Algorithm is suitable for games with high branching factors as it does not consume time playing out all possible branches.

Execution of the algorithm gives a bestmove in every iteration so it can be stopped anywhere to get the result earlier.

The Algorithm can learn from its previous selections and thus helps to improve each time.

It does not require any evaluation function to make a move.

Future scope in 9 x 9

The present implementation of the game allows only Human players to go first for now.

The current game implementation is based on light playout yielding wins for both humans and machines.

Heavy playout using Reinforcement Learning techniques will make the current game unbeatable and this will be done in near future.

Providing Hints and having Undo button features are yet to be implemented in the game.

Monte Carlo Tree Search is a heuristic algorithm which uses both exploration and exploitation and gives optimal results. It does not search the whole tree exponentially and reduces the computational power of the system. If thoughtfully implemented, it can help us to solve most of the decision-making problems effectively.

Achievements

Technical Achievements

1. Different difficulty levels in the 3x3 game using the depth factor in minimax algorithm. Alpha beta pruning was used to make the agent unbeatable(Level Pro).
2. Minimax Algorithm with Alpha beta pruning extended to a 4X4 board.
3. 9 x 9 ultimate Tic Tac Toe implemented using Monte Carlo Tree Search Algorithm.
4. Feature for two humans to play any of the three games on the same machine.
5. Feature to choose the starting player in 3 x 3 and 4 x 4. The current implementation of 9 x 9 allows only the user to start the game.
6. Feature to provide hints to the user in 3 x 3 and 4 x 4 games (Human vs Human and Human vs Agent)
7. Feature to undo a move made by user in 3 x 3 and 4 x 4 games (Human vs Human and Human vs Agent)

Learnings

1. Basic concepts on AI
2. Minimax Algorithm
3. Monte Carlo Tree Search Algorithm
4. Using Visual Studio Code and GitHub
5. Angular

Recommendations & Future Scope

1. Providing Hints and Undo option features for 9 X 9 game.
2. Optimizing 9 x 9 Monte Carlo Tree Search Algorithm using a heavy playout (Reinforcement learning)
3. Implementing features which will allow humans to play against other humans in different locations over the internet by
   - Providing a link to create a private playroom.
   - Can send direct invites to facebook friends (other social media).
4. Implementing an option to play a league (multiple games and players) and maintaining a leaderboard.
5. Recording the game and providing a replay feature for the user along with hints so that the user can realize which moves of his/her were incorrect.