AN INTELLIGENT CUBIC PLAYER

PROJECT TEAM

20210064	احمد صلاح الدين احمد سيد عبدربه
20210159	إسلام مجدي محمد الصفي علي الجزار
20210124	احمد هاني سعد محمد علي
20210036	أحمد تحسين صلاح الدين عبد المجيد
20210074	احمد عبد الغني احمد عبد الغني
20210057	احمد سعدالدين محمد علي المراكبي
20210081	احمد علي إبراهيم سليمان

ABOUT PROJECT

The project focuses on developing an AI-driven player for the Cubic game, a three-dimensional variant of Tic-Tac-Toe played on a 4x4x4 grid. Through the implementation of advanced algorithms and heuristic techniques, the AI aims to strategically navigate the spatial complexities of the game, offering an engaging and challenging experience to players.

Problem Description:

Cubic presents a unique challenge due to its 3D grid structure, requiring players to think strategically across three dimensions to achieve a winning configuration of four markers. Traditional strategies for Tic-Tac-Toe need adaptation to encompass the spatial nature of Cubic.

Expected Outcome:

When the project is done, there'll be a smart player for Cubic. It will be good at making smart moves using special techniques. It'll make the game fun and tough by using its brainpower to think in 3D. And it'll have an easy-to-use design so people can play without any trouble and have a great time.

PROJECT COMPONENTS:

1.Minimax Algorithm Implementation:

- Develop a Cubic-tailored Minimax algorithm capable of traversing the 4x4x4 grid, assessing potential moves, and selecting the most advantageous move considering multiple dimensions.

2.Alpha-Beta Pruning Integration:

- Implement Alpha-Beta Pruning within the Minimax algorithm to optimize computational efficiency by eliminating redundant evaluations in the game tree, significantly reducing search time.

3.Heuristic Functions Design:

- Devise heuristic functions specific to Cubic that capture spatial patterns and winning configurations, aiding the AI player in rapid assessment and decision-making during gameplay.

4.User Interface Development:

- Create an intuitive and user-friendly interface allowing human players to interact with the 4x4x4 grid, visualize moves, and observe the AI's decisions, enhancing overall user experience.

MAIN FUNCTIONS:

Game Initialization:

- **initialize_board():** Creates an empty 4x4x4 grid/board to start the game.

Player Moves:

- **make_player_move(board, position):** Allows the human player to make a move by placing their marker at a specified position on the board.
- **get_player_move():** Takes user input to determine the position where the player wants to place their marker.

AI Moves:

- **make_ai_move(board):** Invokes the AI to calculate and make a move on the board based on the implemented algorithms (Minimax with Alpha-Beta Pruning).
- **minimax(board, depth, maximizing_player):** Recursive function implementing the Minimax algorithm to evaluate possible moves and select the best move for the AI.

Board Evaluation:

- **check_win(board)**: Checks if the current board configuration constitutes a winning position for either the player or the AI.
- **check_draw(board):** Determines if the game has ended in a draw due to a completely filled board with no winning configuration.

User Interface:

- **display_board(board):** Renders the current state of the board for the user to visualize the game.
- **show_game_result(result):** Displays the result of the game - whether it's a win, loss, or draw.

Game Loop:

- **start_game():** Controls the flow of the game, allowing players to make moves consecutively until there's a winner or a draw.
- **play_again():** Asks the user if they want to play another round and restarts the game accordingly.

SIMILAR APPLICATIONS

1. Chess and Chess Engines:

- Chess: The classic board game where players use strategic thinking and planning to outmaneuver opponents.
- Chess Engines (e.g., Stockfish, Komodo): These engines use sophisticated algorithms (including variants of Minimax and Alpha-Beta Pruning) to analyze positions and make moves.

2. Go and Go AI:

- Go: A board game with simple rules but vast complexity. Players aim to control more territory than opponents.
- AI Players (e.g., AlphaGo, Leela Zero): These AI players utilize neural networks and advanced algorithms to play Go at a high level.

3. Connect Four:

- Connect Four: A two-player connection game where players drop colored discs into a grid and aim to form a line of four discs.
- AI Implementations: Some versions of Connect Four use AI algorithms similar to those employed in Tic-Tac-Toe variants.

4.Online Multiplayer Games:

- Online Multiplayer Games (e.g., Dota 2, League of Legends): AI-driven bots are used for practice or even competing against human players, employing complex decision-making algorithms.

A LITERATURE REVIEW OF ACADEMIC PUBLICATIONS

1.Qubic: 4x4x4 Tic-Tac-Toe By Oren Patashnik

- URL: https://www.jstor.org/stable/2689613

2. 3 Dimensional Tic-Tac-Toe By Danial C. Hanson

- https://www.ripon.edu/wp-content/uploads/2014/11/Summation-2010.pdf

3. 3D tic-tac-toe

- https://en.wikipedia.org/wiki/3D_tic-tac-toe

4.Temporal Difference Learning for the Game Tic-Tac-Toe 3D

- https://www.ai.rug.nl/~mwiering/GROUP/ARTICLES/TTT3D_FINAL.pdf

5. 3D Tic Tac Toe Neural Net

- https://studylib.net/doc/7806248/3d-tic-tac-toe-neural-net

DETAILS OF MINIMAX ALGORITHM

Minimax Algorithm:

Overview:

- Purpose: Minimax is a decision-making algorithm used in two-player games involving perfect information (where both players have full knowledge of the game state).
- Objective: It aims to find the best move for a player by considering all possible future moves and their outcomes.

Process:

1. Tree Exploration: Minimax explores a game tree representing all possible moves and their consequences. Each node represents a game state, and branches represent possible moves.
2. Recursive Evaluation: Starting from the current game state, the algorithm explores possible moves and predicts the outcomes by assuming that both players play optimally.
3. Maximizing and Minimizing: It alternates between two players:
    - The maximizing player aims to maximize its advantage by selecting moves with the highest score.
    - The minimizing player aims to minimize the advantage of the opponent by selecting moves with the lowest score.
4. Backtracking: The algorithm evaluates each possible outcome recursively until it reaches a terminal state (win, lose, or draw).
5. Decision-Making: At each level, the player selects the move leading to the highest score if it's the maximizing player's turn and the move leading to the lowest score if it's the minimizing player's turn.

APPLYING MINIMAX TO OUR GAME

Heuristic Function with Minimax Algorithm:

1. Evaluating the Board State:
   - evaluate_board(board): This function assesses the current state of the board and assigns a score based on certain conditions specific to the Cubic game.
   - Factors to Consider:
     - Presence of potential winning configurations.
     - Patterns or spatial arrangements that indicate an advantage or disadvantage.
     - Number of markers in lines across different dimensions.
2. Integration with Minimax:
   - minimax(board, depth, maximizing_player): Within the Minimax algorithm, the evaluate_board() function is used to assign scores to different game states at the leaf nodes of the game tree.
   - Score Allocation: The heuristic function provides scores that guide the AI in choosing the most favorable move during the search process.

1.

DETAILS OF ALPHA-BETA PRUNING ALGORITHM

Overview:

- Purpose: Alpha-Beta Pruning is an optimization technique applied to the Minimax algorithm to reduce the number of nodes evaluated in the game tree.
- Objective: It aims to speed up the search by pruning branches of the tree that cannot influence the final decision.

Process:

1. Pruning Conditions: During the traversal of the game tree, the algorithm maintains two values: alpha (the best value for the maximizing player) and beta (the best value for the minimizing player).
2. Pruning Criteria: When the algorithm encounters a node with a value worse than the current alpha or beta values, it stops evaluating further nodes in that branch.
    - If the maximizing player finds a value greater than or equal to beta, it doesn't need to explore further nodes in that branch because the minimizing player won't choose it.
    - If the minimizing player finds a value less than or equal to alpha, it doesn't need to explore further nodes because the maximizing player won't choose it.
3. Efficiency Gain: Alpha-Beta Pruning discards unnecessary branches early in the search, significantly reducing the number of nodes evaluated without affecting the final result.

APPLYING ALPHA-BETA PRUNING TO OUR GAME

Heuristic Function with Alpha-Beta Pruning:

1. Evaluating the Board State:

- evaluate_board(board): Similar to the function used with Minimax, it evaluates the current board state based on Cubic-specific conditions to assign a score.

2. Integration with Alpha-Beta Pruning:

- minimax_alpha_beta(board, depth, alpha, beta, maximizing_player): In this implementation, the evaluate_board() function is utilized at leaf nodes, similar to the Minimax version.
- Pruning Optimization: The heuristic function provides scores to guide the pruning process in deciding which branches of the tree to explore or discard.
- Alpha-Beta Pruning Conditions: The scores generated by the heuristic function assist in updating alpha and beta values, enabling efficient pruning by discarding irrelevant branches early.

1.

THE FUTURE MODIFICATIONS YOU’D LIKE TO TRY

The code has a lot of loops (those repetitive sections) that might be making it run slower. Imagine loops like going around in circles – if there are too many circles, it takes longer to finish.

Making the Code Faster:
We can try to use fewer loops but still get the same results. It's like finding a shorter path to the same destination – quicker and more efficient.

The algorithm made choices based on what it thought would help it win quickly. It might not have been thinking too far ahead or considering the bigger picture.

The algorithm might struggle to recognize certain winning patterns or see when the opponent is about to win.

ADVANTAGES & DISADVANTAGES

Advantages:

- **Spatial Complexity:** The 4x4x4 grid introduces a unique challenge, making the game more engaging for players who enjoy spatial reasoning and complex strategy.
- **Creative Gameplay:** Cubic encourages creative thinking as players explore various ways to create winning lines within three dimensions. The open nature of the gameplay allows for inventive and unexpected moves, fostering a sense of creativity and uniqueness in each game.

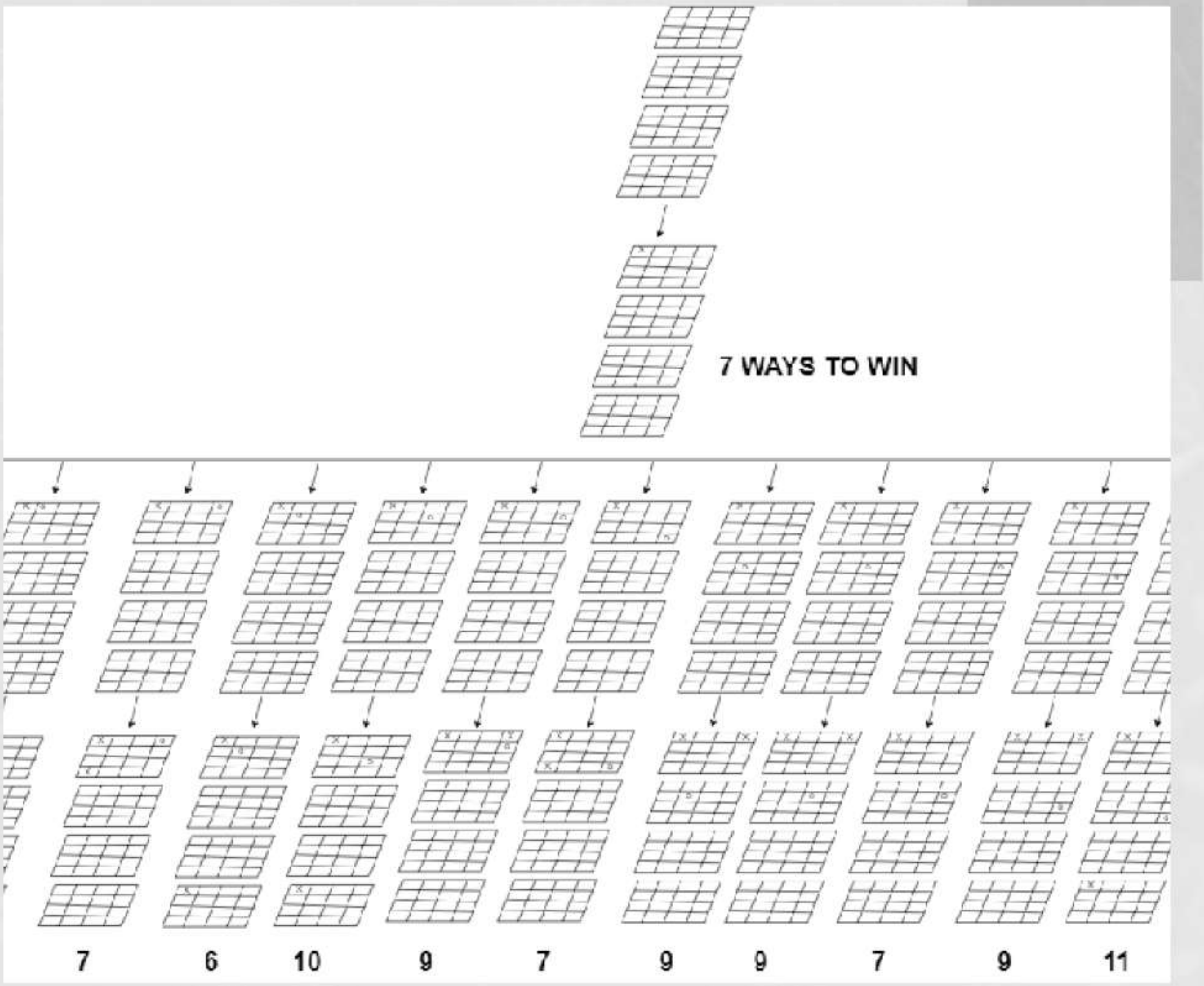
Disadvantages

- **Increased Decision Complexity:** The larger game space increases the number of possible moves, making it more challenging for players to analyze and choose the optimal move within a reasonable timeframe.
- **Learning Curve:** The additional dimension may pose a learning curve for new players. Understanding the spatial dynamics and developing effective strategies may take time.

DIAGRAMS:

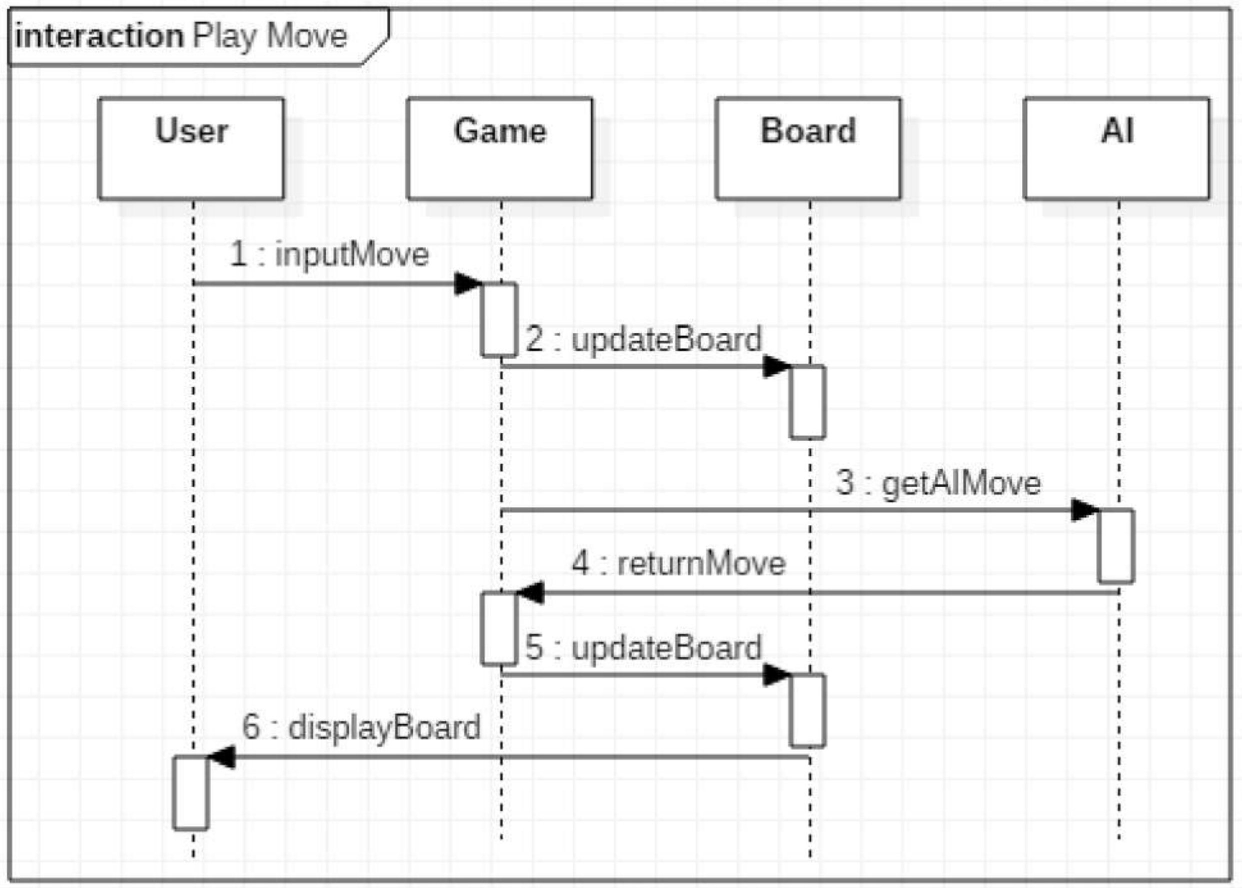
1.OVERVIEW ABOUT BEST WAY TO WIN THE GAME.

1.



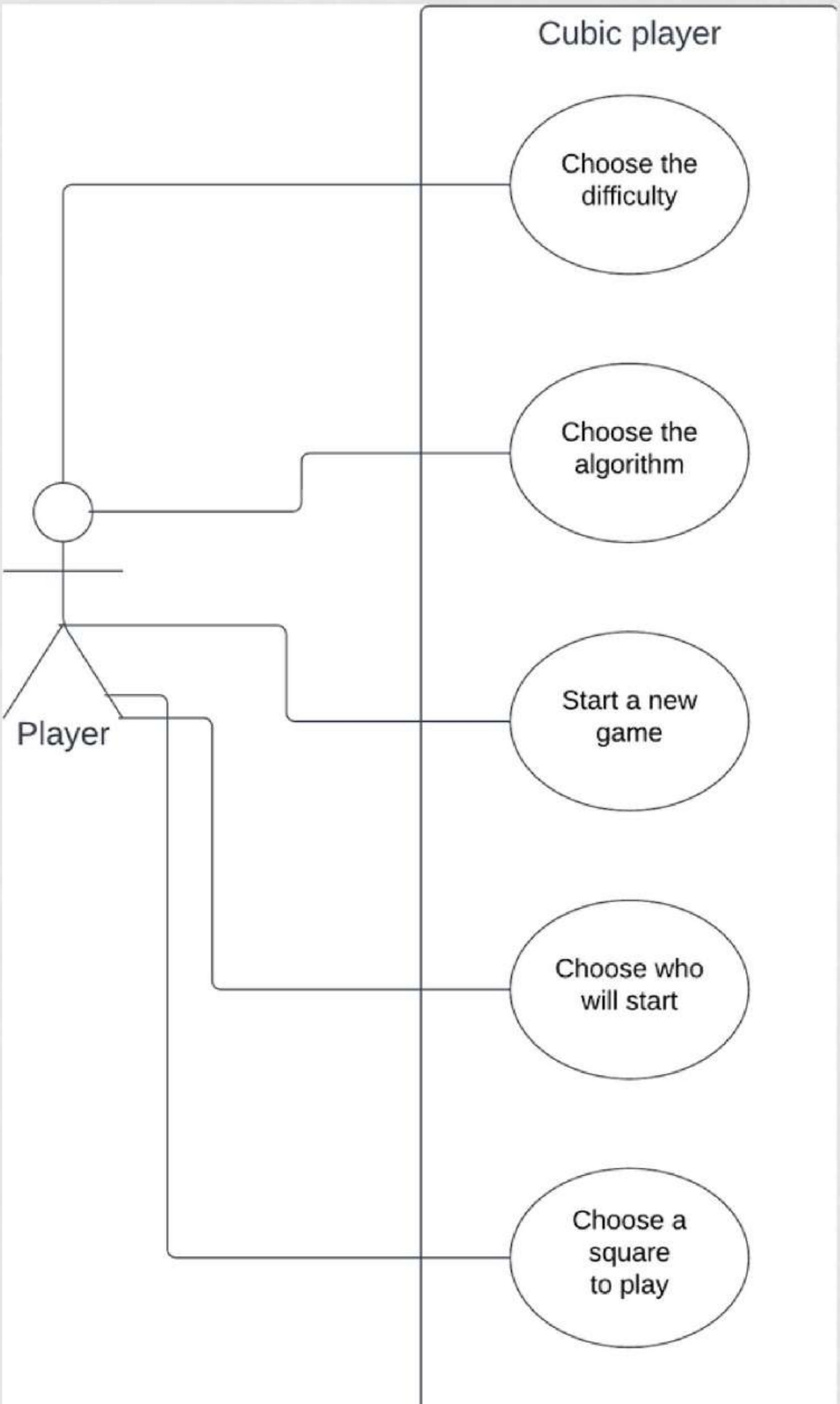
DIAGRAMS:

2.Sequence Diagram



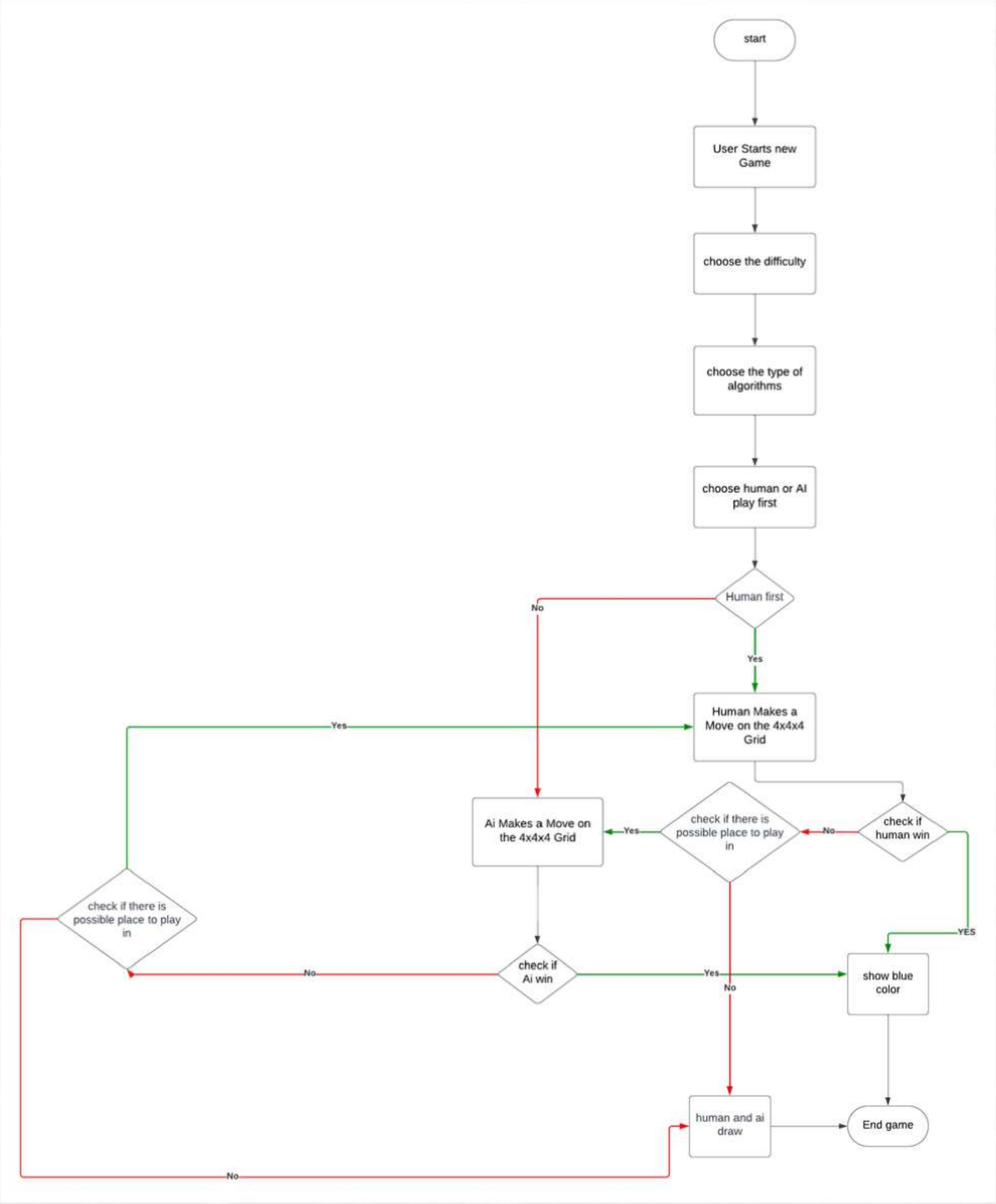
DIAGRAMS:

3.USE CASE (UML)



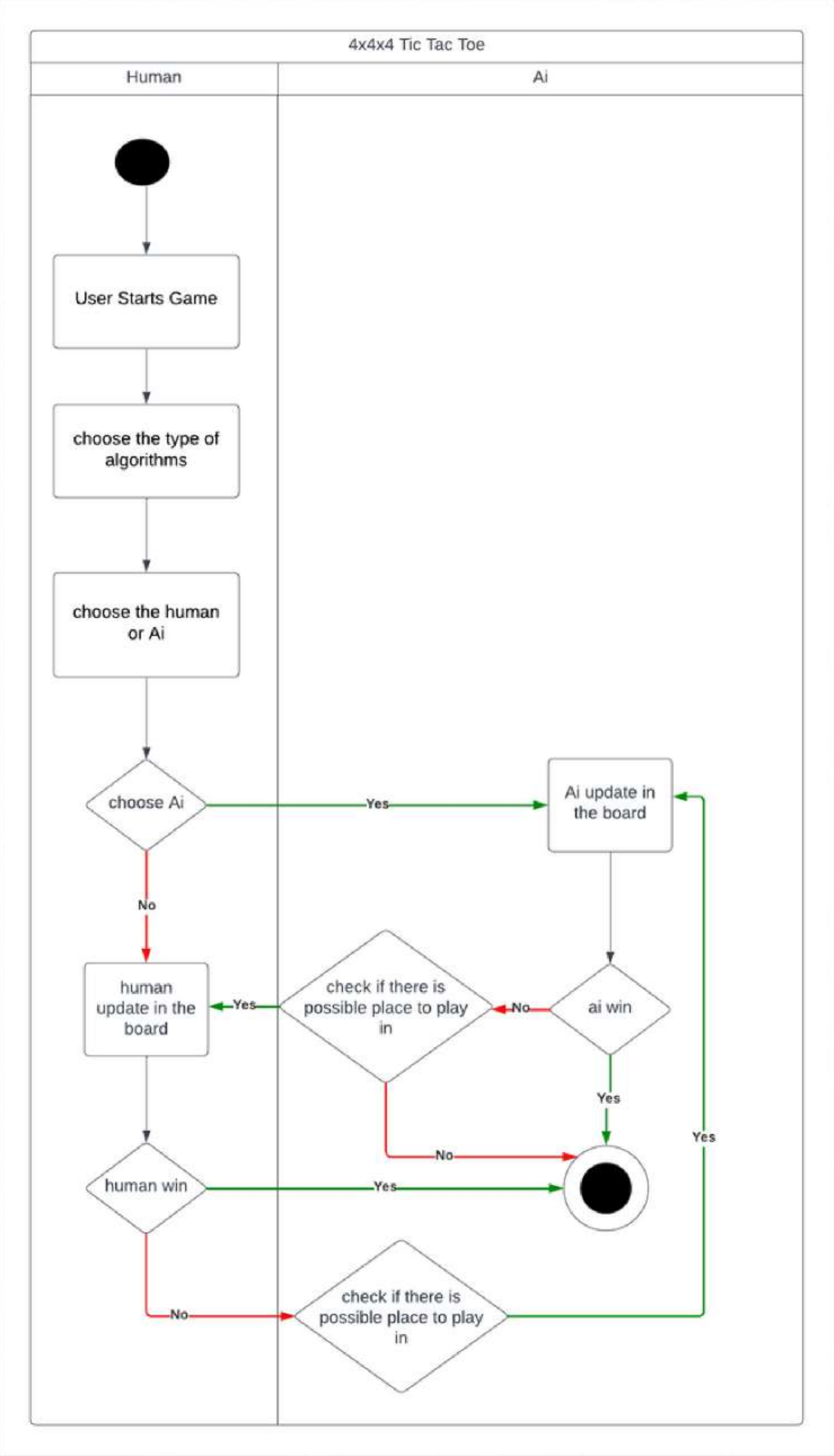
DIAGRAMS:

4.FLOW CHARTS


start
User Starts new Game
choose the difficulty
choose the type of algorithms
choose human or AI play first
Human first
No
Yes
Human Makes a Move on the 4x4x4 Grid
Yes
Ai Makes a Move on the 4x4x4 Grid
Yes
check if there is possible place to play in
No
check if human win
YES
check if there is possible place to play in
No
check if Ai win
Yes
No
show blue color
human and ai draw
End game
No

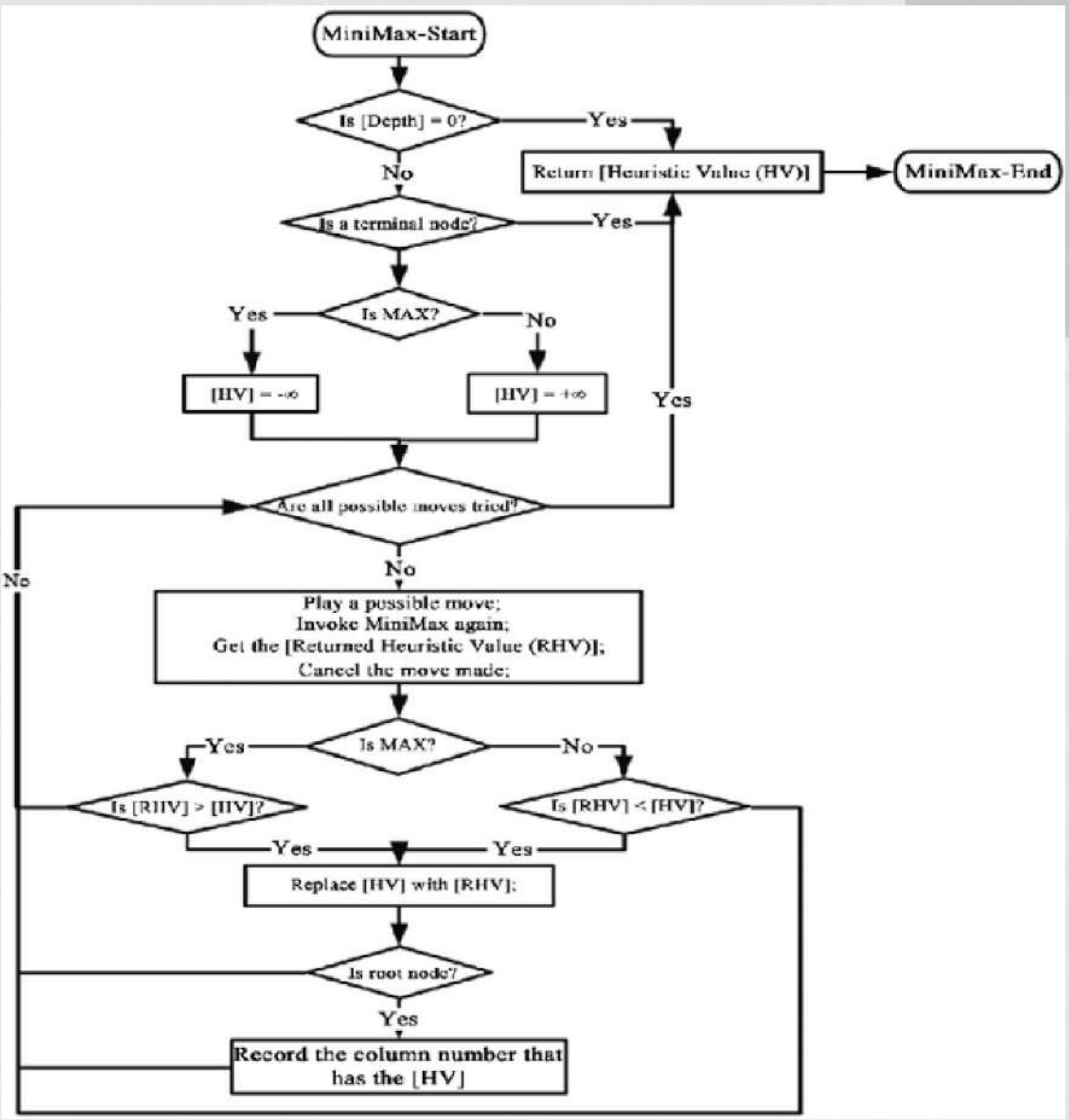
DIAGRAMS:

5.Activity diagrams:


4x4x4 Tic Tac Toe
Human
Ai
User Starts Game
choose the type of algorithms
choose the human or Ai
choose Ai
Yes
No
Ai update in the board
human update in the board
Yes
check if there is possible place to play in
No
ai win
Yes
No
Yes
human win
Yes
No
check if there is possible place to play in

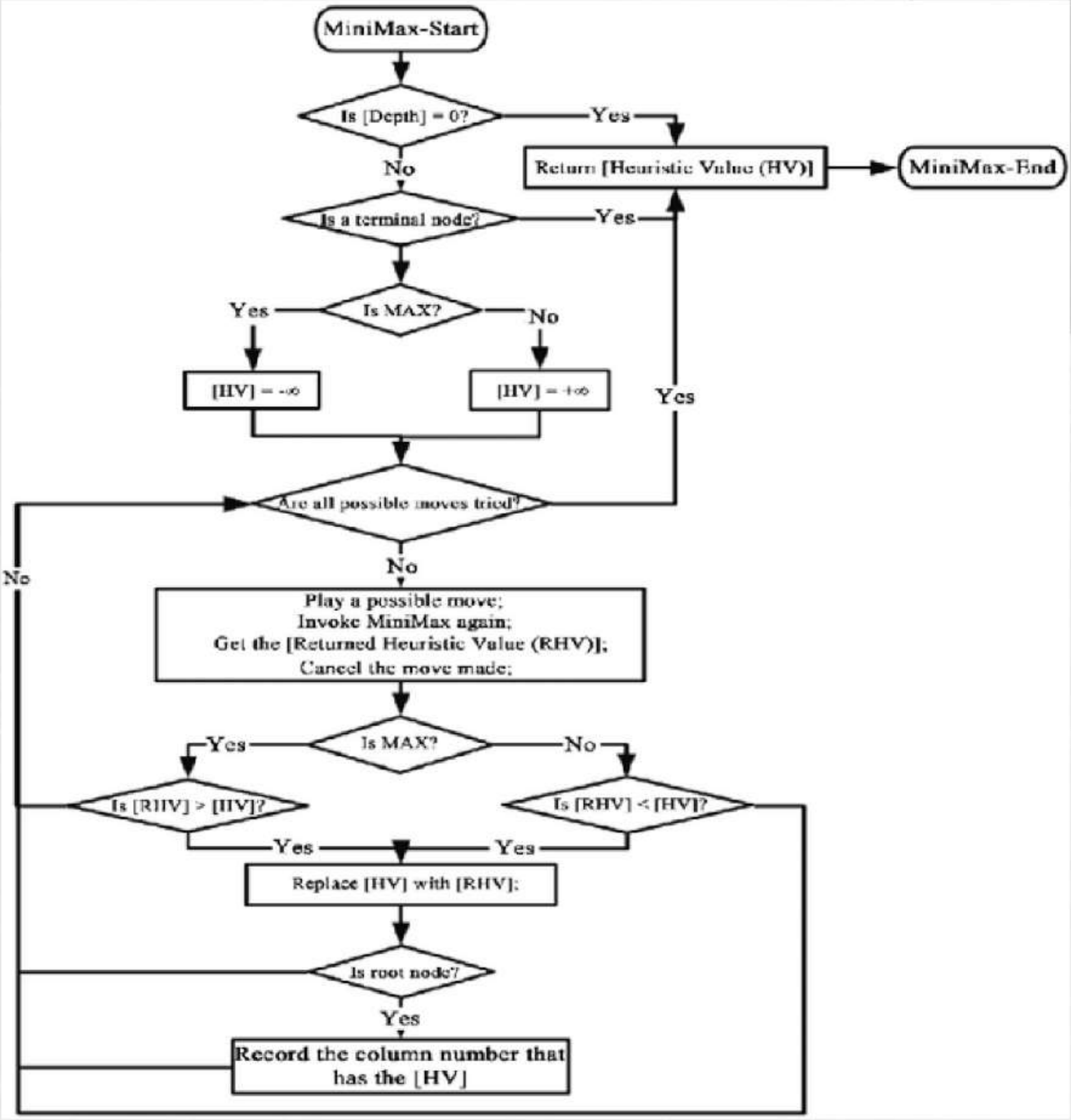
DIAGRAMS:

6. MiniMax algorithm



DIAGRAMS:

6. MiniMax algorithm



CONCLUSION:

The completion of this project will demonstrate the integration of advanced AI algorithms and heuristic approaches tailored specifically for the Cubic game. The aim is to develop an AI system capable of effectively playing on the 4x4x4 grid. The project emphasizes the utilization of artificial intelligence techniques to tackle the complexities of spatial reasoning in gaming scenarios. Its primary objective is to showcase the successful application of AI strategies in mastering the Cubic game within the scope of an academic assignment for an AI course.

Document URL:

https://github.com/ahmed-sala/ai-project