A Project Report On

Tic Tac Toe game

Submitted in partial fulfilment for the degree of
Bachelor of Technology in
Information Technology

Submitted by
Maitree Pandey (44)
Tanvi S. Patil (50)
Shivani Pawar (54)
Laxmi Ranjvan (59)

Under the guidance of
Prof. Dr. Anita Morey

USHA MITTAL INSTITUTE OF TECHNOLOGY

S.N.D.T Women's University,Mumbai
2024-25

Declaration

I, Maitree Pandey, along with Tanvi S. Patil, Shivani Pawar, and Laxmi Ranjvan, hereby declare that the work presented in this project, entitled “Tic Tac Toe Game,” is entirely our own. The content has been developed through our independent efforts, research, and scholarly contributions. We affirm that the ideas, concepts, and contributions in this work are the result of our intellectual endeavors and that all data, figures, tables, and findings presented are authentic, without fabrication or manipulation. No AI-based tools have been used to generate significant portions of this project, including content, research objectives, hypotheses, or analysis. We have properly cited and referenced all external sources, ensuring no plagiarism or unauthorized use of intellectual property. This work has been conducted independently, without any collaboration or assistance that would compromise its originality. Adhering to the principles of academic integrity and ethical research, we understand the consequences of academic dishonesty and affirm that this declaration accurately reflects the authenticity of our work.

Date: *[17-03-2025]*

Signature:

CERTIFICATE

This is to certify that (Ms.Maitree Pandey, Ms.Tanvi.S.Patil ,Ms.Shivani Pawar, Ms.Laxmi Ranjvan) has completed the Mini Project report on the topic *"Tic Tac Toe game"* satisfactorily in partial fulfillment for the Bachelor's Degree in (Information Technology) under the guidance of Dr. Anita Morey during the year 2024-25 as prescribed by Shreemati Nathibai Damodar Thackersey Women's University (SNDTWU)

Guide

Prof. Dr. Anita Morey

Head of Department

Dr. Sanjay Shitole

Principal
Dr.Yogesh Nerkar

Examiner 1

Examiner 2

Acknowledgement

We have a great pleasure to express our gratitude to all those who have contributed and motivated during our project work. We thank our principal, Dr. Yogesh Nerker, our HoD, Dr. Sanjay Shitole and our guide, Prof. Dr. Anita Morey for mentoring us.

Date: 17-03-25

Name of the candidates :
Maitree Pandey (44)
Tanvi S. Patil (50)
Shivani Pawar (54)
Laxmi Ranjvan (59)

Abstract

The Tic-Tac-Toe game is a well-known two-player strategy game played on a 3x3 grid, where players take turns placing their assigned symbols, either "X" or "O," with the goal of forming a consecutive line of three symbols horizontally, vertically, or diagonally. This project presents a digital version of the game that allows players to compete against either another human or an AI opponent. The implementation ensures that the game adheres to the traditional rules, validating each move, detecting wins, and determining when the game ends in a draw. The system provides real-time feedback, keeping the gameplay engaging and interactive. This digital version includes various enhancements to improve the user experience. Players can enjoy a seamless interface with smooth transitions between turns, clear visual indications for moves, and an option to restart the game at any time. The AI component, when implemented, can have multiple difficulty levels, ranging from random moves to advanced strategies using algorithms like Minimax. These features allow users to challenge themselves by competing against a more intelligent opponent, creating a dynamic and strategic gaming experience. From a development perspective, this project demonstrates key programming concepts such as user input handling, conditional statements, loops, recursion, and game state management. It can be built using different programming languages, including Python, JavaScript, or C++, and can be adapted for various platforms such as web applications, mobile apps, or desktop software. Additional functionalities like score tracking, multiplayer options, and sound effects can further enhance the overall experience. By developing this project, programmers gain valuable skills in logic-building, UI/UX design, and AI implementation while creating an engaging and interactive game.

Contents

Nomenclature

- **Grid**: The 3x3 board on which the game is played. It consists of nine cells arranged in three rows and three columns. Each cell can either be empty, contain an "X," or contain an "O."
- **Player**: A participant in the game. There are typically two players in Tic-Tac-Toe: Player 1 (usually designated with "X") and Player 2 (usually designated with "O"). In some versions of the game, one player may play against an AI opponent.
- **Move**: The action performed by a player to place their symbol ("X" or "O") in an available cell of the grid.
- **Symbol**: The characters used by the players. Player 1 typically uses "X," while Player 2 uses "O." These symbols are placed in the grid during the game.
- **Turn**: The act of one player making a move. Players take turns to place their symbols on the grid until the game ends.
- **Winning Condition**: A condition where a player achieves three of their symbols in a row, either horizontally, vertically, or diagonally. A player wins when this condition is met.
- **Draw (Tie)**: A result when the grid is filled, and no player has met the winning condition. This signifies that the game ends without a winner.
- **Game State**: The current status of the game, including the arrangement of symbols on the grid and whether the game has ended, is ongoing, or has resulted in a win or draw.
- **AI (Artificial Intelligence)**: A computer-controlled player in single-player mode. The AI can be programmed with different difficulty levels, ranging from random moves to optimal strategies (such as using the Minimax algorithm).
- **Reset**: The action that resets the grid to its initial empty state, allowing the game to start over.
- **Score Tracker**: A component used to keep track of the number of wins, losses, and draws for each player over multiple rounds or games.
- **Valid Move**: A move made by a player that places their symbol in an empty, unoccupied cell. Invalid moves occur when a player tries to place their symbol in a cell already occupied by another symbol.

- **Minimax Algorithm**: A decision-making algorithm commonly used in AI to determine the best possible move. It explores all possible moves and selects the one that minimizes the maximum possible loss for the AI player.
- **Game Over**: The state in which the game has concluded, either because a player has won or because the grid is filled and the game results in a draw.
- **Interface (UI)**: The visual representation or layout where the game is played, allowing users to interact with the grid, make moves, and view the status of the game. This can be either text-based or graphical.
- **Cell**: An individual square in the grid, where each player places their symbol ("X" or "O"). Each cell can be referenced by its row and column indices (e.g., [1, 1] for the center).
- **Victory Path**: A set of three cells aligned in a row, column, or diagonal that contains the same symbol ("X" or "O") from one player, resulting in that player winning the game.
- **Game Logic**: The underlying set of rules and algorithms that govern how the game is played, including checking for win conditions, handling player turns, and determining when the game ends.

Chapter 1

Introduction

Tic-Tac-Toe, also known as Noughts and Crosses, is a classic and widely recognized strategy game that is played by two players on a 3x3 grid. Each player takes turns placing their respective symbol—either "X" or "O"—in an empty cell on the grid. The goal of the game is to align three of their symbols in a row, either horizontally, vertically, or diagonally, before their opponent does. The game is simple to learn, easy to play, and offers a fun and competitive experience, making it popular among players of all ages.

Tic-Tac-Toe can be played on paper or through a digital interface, and the rules remain the same: the game ends when a player wins by aligning three symbols in a row or when all cells are filled without a winner, resulting in a draw. The simplicity of the game makes it an ideal choice for beginners to practice strategic thinking and pattern recognition. Despite its straightforward rules, Tic-Tac-Toe can also present a fun challenge when players attempt to outsmart each other, especially when one of the players is an AI with increasing difficulty levels.

In the digital version of Tic-Tac-Toe, the game logic is implemented through algorithms that manage turn-taking, detect winning conditions, and handle the game's end state. Advanced features such as AI-controlled opponents and different difficulty levels can be added to enhance the gaming experience. The implementation of this game provides an excellent opportunity to practice and reinforce fundamental programming concepts, including conditional logic, loops, arrays, and user input handling.

Overall, Tic-Tac-Toe remains one of the most enduring and accessible games, both in its physical and digital forms, providing endless entertainment while offering insights into the power of simple algorithms and game design.

Tic-Tac-Toe, also known as Noughts and Crosses, is a timeless strategy game that has captivated players of all ages for generations. It is played on a 3x3 grid where two players, typically designated as "X" and "O," take turns marking empty spaces with their respective symbols. The objective is to be the first to align three of their symbols in a row—either horizontally, vertically, or diagonally—on the grid. If all spaces are filled and no player has achieved this goal, the game results in a draw.

1.2 Research Motivation

The motivation for developing a digital Tic-Tac-Toe game lies in its educational value and potential for exploring computer science, game theory, and AI concepts. Despite its simplicity, the game offers valuable learning opportunities in programming and algorithm design.

Educational Significance: Tic-Tac-Toe helps beginners grasp decision-making, pattern recognition, and strategy development. It introduces key programming concepts like game loops, user input handling, conditional logic, and state management, making it an ideal starting point for novice programmers.

Artificial Intelligence Exploration: Implementing an AI opponent using algorithms like Minimax provides insight into game tree exploration, heuristic evaluation, and optimization techniques, offering practical experience in AI development. AI can be further enhanced by incorporating learning mechanisms or adaptive strategies that evolve based on player moves.

Enhancing Player Experience: Introducing difficulty levels and adaptive AI improves engagement, ensuring a balanced and enjoyable challenge while preventing predictable gameplay outcomes. Additional features like interactive UI, score tracking, and animations can make the game more immersive.

1.3 Research Objectives

The primary objective of this research is to develop and analyze a digital version of the Tic-Tac-Toe game, with a focus on both the game mechanics and the implementation of Artificial Intelligence (AI). The research aims to explore and address fundamental topics in game development, algorithm design, and AI implementation, while enhancing the player experience. The specific objectives of this research are as follows:

This project focuses on developing a fully functional digital Tic-Tac-Toe game that supports both two-player mode and AI integration. The game will correctly process user inputs, update the board in real-time, and determine win conditions, including draws, ensuring smooth and fair gameplay. A restart option will be included to allow users to play multiple rounds seamlessly. An AI opponent will be integrated using decision-making algorithms such as Minimax, providing multiple difficulty levels ranging from random moves to optimal strategies. The AI's decision-making process will be optimized to offer a balanced challenge, preventing predictable or overly difficult gameplay. Techniques like Alpha-Beta Pruning will be explored to enhance AI efficiency, minimizing unnecessary computations while maintaining strong strategic play. To ensure seamless performance, efficient game algorithms will be implemented to detect win conditions and determine optimal moves in real time. Various search techniques, including Minimax and Alpha-Beta Pruning, will be analyzed for their computational efficiency, ensuring quick and accurate decision-making.

Finally, the project will assess the performance of different AI algorithms, comparing their speed, efficiency, and accuracy. Computational complexity, search depth, and the number of evaluated nodes will be analyzed to optimize response time. These improvements will enhance the overall gameplay experience, making the AI opponent more dynamic and responsive.

1.4 Scope of the Study

The scope of this study is focused on the development and implementation of a digital Tic-Tac-Toe game, exploring key concepts in game development, artificial intelligence (AI), and algorithm design. The study primarily addresses the design, functionality, and optimization of a Tic-Tac-Toe game with an AI component, emphasizing the educational and computational aspects of game development. The specific areas covered in the scope of the study are as follows:

This study focuses on developing a digital Tic-Tac-Toe game with core mechanics like player moves, win conditions, and board updates. A user-friendly interface, either graphical or text-based, ensures smooth interaction. AI integration enables single-player gameplay, with difficulty levels ranging from random moves to advanced strategies using Minimax and Alpha-Beta Pruning. Efficient algorithms are implemented for AI decision-making, optimizing performance for speed and accuracy. The AI's adaptability ensures a balanced challenge without overwhelming players. A well-designed UI enhances user experience with clear feedback and restart options. Testing and user feedback help refine interactivity and engagement. The study also explores Tic-Tac-Toe's educational value in teaching programming, AI concepts, and strategic thinking. Additionally, it examines how the game improves cognitive skills like problem-solving and pattern recognition. Lastly, its simplicity serves as a gateway to more complex game design and AI development.

Limitations:

The game's complexity is limited to the basic 3x3 grid layout. While the focus is on designing a simple game, the study does not address scaling the game to more complex grid sizes or implementing advanced game features (e.g., multi-board gameplay or 3D Tic-Tac- Toe). The AI implementation, while featuring varying levels of difficulty, does not incorporate machine learning or advanced AI techniques such as neural networks or reinforcement learning. The AI is based on rule-based decision-making, with algorithms like Minimax and Alpha-Beta Pruning. The scope of the study is confined to the development of the game and AI within a controlled environment, not addressing large-scale multiplayer or networked versions of the game.

Technologies and Tools:

The study focuses on using widely accessible and common programming languages, such as Python, JavaScript, or C++, to implement the game and its features. For the AI implementation, tools such as search algorithms (Minimax, Alpha-Beta Pruning) are used to build the decision-making system. The study will also employ a basic graphical user interface (GUI) or command-line interface (CLI) to present the game to users, using technologies like HTML, CSS, and JavaScript (for web-based versions) or Tkinter for Python-based GUI implementations.

1.5 Problem Statement

The current digital implementations of Tic-Tac-Toe often lack an engaging AI opponent and fail to offer a balanced difficulty level, resulting in either overly predictable gameplay or frustratingly difficult challenges. Additionally, many versions do not optimize AI decision-making, leading to poor performance and slow response times. This study aims to address these issues by developing a digital Tic-Tac-Toe game that incorporates an adaptive AI with multiple difficulty levels, uses efficient game algorithms for faster performance, and provides an intuitive, user-friendly interface to enhance the overall player experience.

The current digital implementations of Tic-Tac-Toe often lack an engaging AI opponent and fail to offer a balanced difficulty level, resulting in either overly predictable gameplay or frustratingly difficult challenges. Additionally, many versions do not optimize AI decision-making, leading to poor performance and slow response times. This study aims to address these issues by developing a digital Tic-Tac-Toe game that incorporates an adaptive AI with multiple difficulty levels, uses efficient game algorithms for faster performance, and provides an intuitive, user-friendly interface to enhance the overall player experience. The focus is to create an enjoyable, challenging, and optimized version that caters to players of varying skill levels while ensuring seamless gameplay.

Chapter 2

Review of Literature

The literature surrounding Tic-Tac-Toe primarily focuses on its historical development, its use as an educational tool in game theory and artificial intelligence (AI), and the optimization techniques applied to its AI algorithms. Despite its simplicity, Tic-Tac-Toe remains a significant subject for research due to its educational value and its potential for demonstrating fundamental computational concepts. This section provides a review of the key studies and contributions related to the game's development, AI integration, and educational applications.

1. Historical and Evolutionary Perspective

Tic-Tac-Toe, also known as Noughts and Crosses, dates back to ancient civilizations. Historical records trace its origins to ancient Egypt and the Roman Empire, where a similar grid-based game called *Terni Lapilli* was played (Gibson, 2002). The modern version of the game, played on a 3x3 grid, became popular in the 19th century and has since evolved from a traditional pencil-and-paper game to a digital form. Its transformation into a computer-based game has sparked significant interest in early computer science and AI, where it became one of the first games to be programmed on computers, marking a pivotal moment in the development of artificial intelligence (Shannon, 1950).

2. Tic-Tac-Toe in Artificial Intelligence and Game Theory

Tic-Tac-Toe has been widely studied in the context of game theory and AI, particularly as a benchmark for testing decision-making algorithms. The game's simplicity makes it an ideal model for exploring the basic concepts of AI, particularly in strategic decision-making and game tree exploration (Russell & Norvig, 2010). Early AI implementations of Tic-Tac-Toe focused on the use of *Minimax*, an algorithm designed to optimize decision-making by exploring all possible game states and selecting the best possible move (Luger, 2009).

Minimax operates by evaluating each potential game state using a score function and selecting the move that maximizes the AI's chances of winning while minimizing the opponent's chances.

Synthesis and Discussion

The synthesis and discussion of the literature on Tic-Tac-Toe reveal several important insights that guide the development of a digital version of the game with AI. While Tic-Tac-Toe is fundamentally a simple game, it has played an essential role in computer science, particularly in demonstrating foundational concepts in artificial intelligence (AI), game theory, and algorithm design. Through the review of existing research, it becomes clear that the evolution of AI, game algorithms, and user experience in Tic-Tac-Toe can be improved to create a more engaging and educational experience.

Chapter 3

Software and Hardware Requirements

1. Programming Language: Python

   Python is chosen as the programming language for this project due to its simplicity, readability, and extensive support for AI-based applications. It is widely used in game development and artificial intelligence due to its concise syntax and rich ecosystem of libraries. Python's structured approach allows for easy implementation of game logic, user interactions, and AI algorithms.Simple, readable, and widely used in AI applications. Provides powerful libraries like Tkinter for GUI and NumPy for matrix operations.

2. GUI Library: Tkinter

   Tkinter, a built-in Python library, is used to create the graphical user interface (GUI) for the Tic-Tac-Toe game. It provides a simple yet effective way to build interactive applications without requiring additional installations. Built-in Python library for creating graphical applications. Allows users to click, restart, and interact with the game.

3. Algorithm: Minimax

   The AI opponent in this game is implemented using the **Minimax algorithm**, a well-known decision-making strategy in game theory. The Minimax algorithm evaluates all possible game states and selects the optimal move that maximizes its chances of winning while minimizing the opponent's chances.

   How Minimax Works in Tic-Tac-Toe:

   - The AI simulates every possible move it can make, along with all possible counter-moves by the human player.
   - It assigns a score to each move based on whether it leads to a win (+1), a loss (-1), or a draw (0).
   - The algorithm recursively explores all potential outcomes and chooses the move that ensures the best possible result for the AI.
   - Because Tic-Tac-Toe is a finite game with a small search space, Minimax guarantees optimal play, meaning the AI will never lose if it plays first or makes the best possible moves.

   To further improve efficiency, **Alpha-Beta Pruning** can be applied to reduce unnecessary computations, making AI decision-making faster without compromising optimal play. AI analyzes all possible moves and selects the best one. Ensures the AI never loses and plays optimally.

Chapter 4

Models Fundamentals and Working

4.1 MinMax Algorithm

The Minimax algorithm is a fundamental decision-making approach used in two- player games like Tic-Tac-Toe. It ensures that the AI plays optimally by minimizing potential losses while maximizing potential gains. The algorithm operates on a game tree, where each node represents a possible game state.

1. Game Tree Representation:

The game is visualized as a tree structure, with the root node representing the current board state. Each child node represents a possible move, expanding the tree until all possible game outcomes are represented.

2. Maximizing and Minimizing

The algorithm alternates between two roles:

The maximizer (AI) selects the move with the highest possible score.

The minimizer (opponent) chooses the move that leads to the lowest possible score for the AI.

This process ensures that the AI anticipates the opponent's best response at each step.

3. Leaf Nodes and Evaluation

The algorithm continues to expand the tree until a terminal state is reached (win, loss, or draw). Each terminal node is assigned a score:

+10 for an AI win

-10 for an opponent win

0 for a draw

4. Backtracking Scores

Once all terminal nodes are evaluated, the algorithm backtracks up the tree, selecting the best move:

Maximizer nodes pick the child node with the highest score.

Minimizer nodes pick the child node with the lowest score.

By following this approach, the AI ensures an optimal strategy, either winning or forcing a draw in Tic-Tac-Toe.

Working of Minimax in Tic Tac Toe

The algorithm explores the game tree, where:
Nodes represent game states.
Branches represent possible moves.
Leaf nodes represent terminal states (win, loss, or draw).

The AI assigns a score to each terminal state:
+1 if AI wins.
-1 if the opponent wins.
0 for a draw.

Steps of Minimax Algorithm:
1. Check for Terminal State- If the board has a winner or a draw, return the corresponding score.

2. Generate All Possible Moves- Iterate through empty spaces and simulate placing a move.

3. Recursive Call- Call the Minimax function for the next move.
The maximizing player (AI) picks the highest score.
The minimizing player (opponent) picks the lowest score.

4. Return the Best Move- The AI selects the move that leads to the most favorable outcome.

Advantages of Minimax Algorithm

- Guaranteed Optimal Move: Ensures the best move is always chosen.
- Works for All Two-Player Games: Used in Chess, Checkers, etc.
- Logical and Systematic Approach: Considers all future possibilities.

4.2 Working

1. Main Menu Functionality

The main menu serves as the entry point of the game, offering the following features:

- Play with AI Button: Launches a game mode where the user plays against an AI opponent.
- Two-Player Mode Button: Starts a game mode where two human players take turns.
- Exit Button: Closes the application.

Implementation in the Code

- The main_menu() function is responsible for displaying the main menu.
- It creates buttons using Tkinter that allow the player to choose a game mode.
- The layout is managed using Tkinter Frames and Buttons.

Additionally, the fullscreen feature is implemented:

- The game starts in full-screen mode.
- Pressing the Escape key exits full-screen.

2. AI Mode Functionality

The AI in this Tic Tac Toe game uses the Minimax Algorithm to determine the best possible move. AI Decision-Making Process

1. Check for a winning move
    - If the AI can win in one move, it takes that move.
2. Simulate possible future moves
    - The AI evaluates all potential game states to predict the best outcome.
3. Assign scores to moves
    - Winning moves receive a high score (1).
    - Losing moves receive a low score (-1).
    - Draws receive a neutral score (0).
4. Choose the best move
    - The AI selects the move with the highest score to play optimally.

Implementation in the Code

- The ai_move() function determines the best move using Minimax.
- The minimax() function evaluates all possible board positions.
- The AI plays only when it's its turn, ensuring fair gameplay.

3. Winner Determination

How the game detects a winner The game checks for three matching symbols in:

- Rows: (0,1,2), (3,4,5), (6,7,8)
- Columns: (0,3,6), (1,4,7), (2,5,8)
- Diagonals: (0,4,8), (2,4,6)

If no winner is found and no moves remain, the game results in a draw.

Implementation in the Code

- The check_winner() function checks all possible winning patterns.
- If a winning pattern is found, it displays a message and resets the board.
- If no spaces remain and no winner is found, a draw message appears.

4. Game Controls (Restart and Back)

Features of Game Controls

- Turn Indicator: Displays which player's turn it is.
- Restart Button: Resets the board but keeps the selected mode.
- Back Button:
    - In AI Mode: Returns to AI selection.
    - In Two-Player Mode: Returns to the main menu.

Implementation in the Code

- The reset_board() function clears the board and resets the turn.
- The main_menu() and play_with_ai() functions allow navigation between screens.

5. Code Explanation- Step-by-Step Breakdown

A. Initializing the Game

- The Tkinter window is set up.
- The 3x3 board is defined as a list.
- Player turns and AI processing are initialized.

B. Handling Player Moves

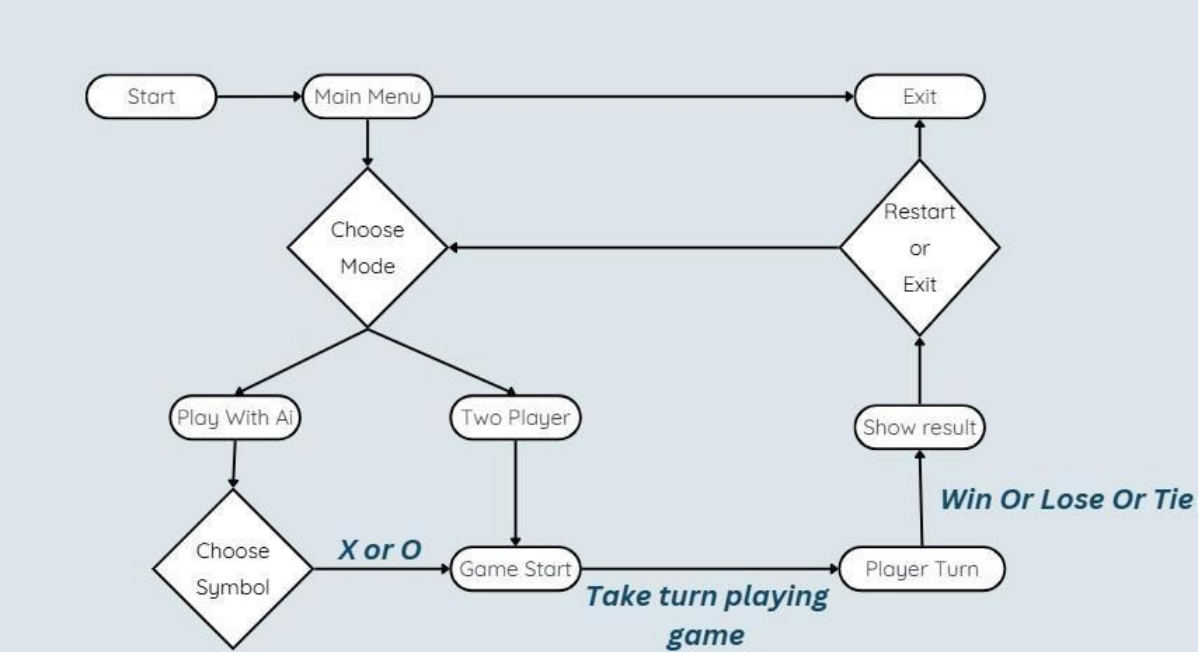
- A button click updates the board with 'X' or 'O'.
- The turn switches automatically.

C. Checking for a Winner

- The game scans for winning patterns.
- If a winner is found, a message appears.
- If the board is full and no one won, it declares a draw.

D. AI Move Execution

- The Minimax Algorithm selects the best move.
- The AI places its symbol and switches turns.



Step 1: Initialize the Game
Create a Tkinter window and configure it to full-screen mode.
Display a main menu with options:
Play with AI
Two Player Mode
Exit

Step 2: Select Game Mode
If "Two Player Mode" is selected:
Start the game with two human players.
If "Play with AI" is selected:
Ask the user to choose X or O.
Start the game with AI as an opponent.

Step 3: Initialize the Game Board
Create a 3×3 grid using Tkinter buttons.
Each button represents a cell on the Tic-Tac-Toe board.
Display the current turn on the screen.

Step 4: Handle Player Moves
When a player clicks on a cell:
Check if the cell is empty.
Update the board state and button text.
Check for a winner or a draw.
Switch turns between X and O.

Step 5: AI Move using Minimax Algorithm
The AI evaluates all possible moves.
It assigns a score to each move using the Minimax algorithm:
If the AI wins, the score is +1.
If the player wins, the score is -1.
If it's a draw, the score is 0.
AI chooses the best possible move to maximize its chance of winning.
The AI places its symbol on the board and updates the UI.

Step 6: Check for Winner
Check all possible winning combinations (rows, columns, diagonals).
If a player has three marks in a row, display a win message.
If all cells are filled and there is no winner, display a draw message.

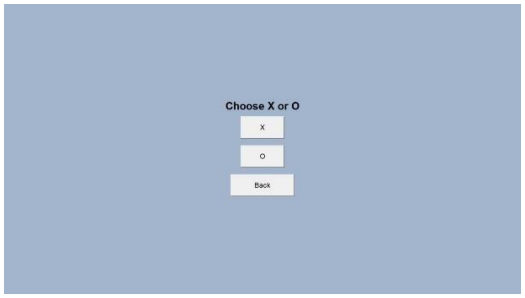
Step 7: Restart or Exit
Provide options to:
Restart the game (reset board state).
Go back to the main menu.
Exit the application.

Chapter 5

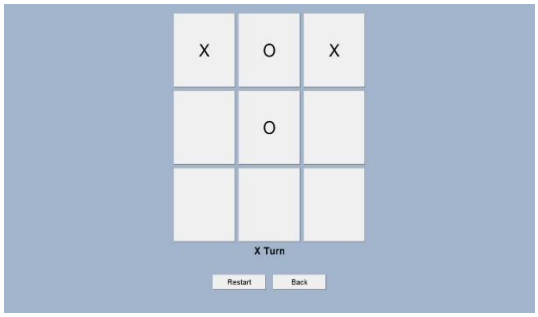
Implementation:

1 . Main Menu:

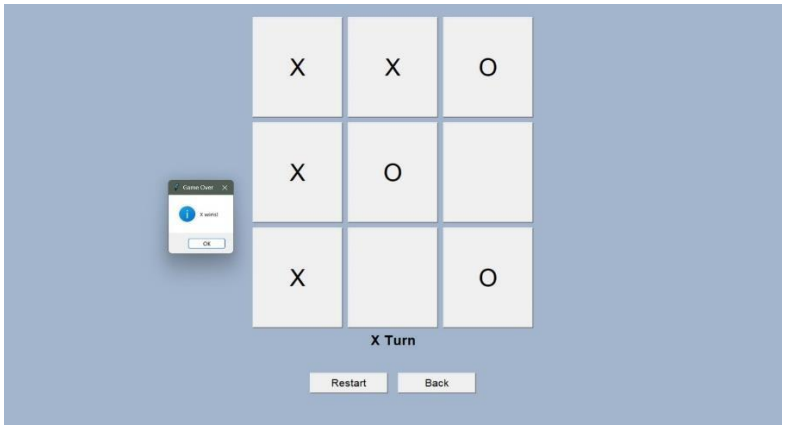
2. Play with ai choose option



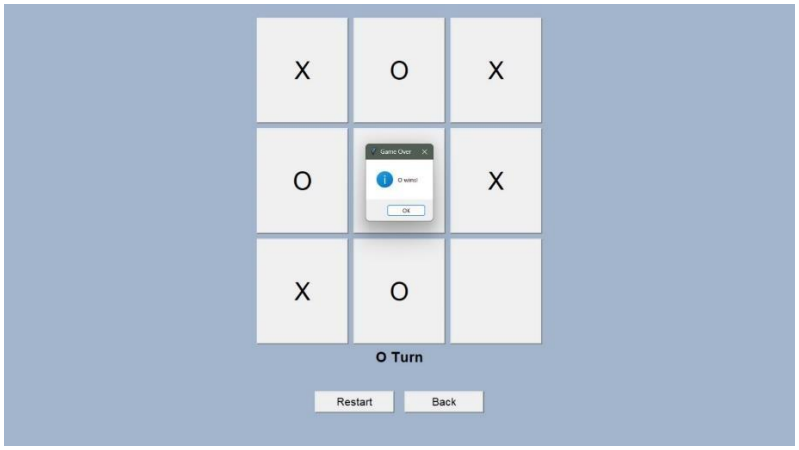
3. Game start in ai mode



4. Two player



5. Winner



Chapter 6

Conclusion

Tic-Tac-Toe's elegant simplicity offers profound insights into strategic interactions. [1] The guaranteed draw under perfect play underscores the importance of both offensive and defensive planning. Controlling the center square exemplifies the power of positional advantage, a concept echoed in numerous strategic domains. The dynamics of creating and neutralizing threats introduce the core idea of anticipating an opponent's moves and formulating counter-strategies. This easily grasped game serves as a crucial stepping stone in developing logical reasoning, pattern recognition, and the ability to think ahead. Its finite nature allows for complete analysis, making it a perfect model for understanding basic game theory principles and the concept of optimal decision-making in a constrained environment, particularly for those new to strategic thinking. [2]

References

- Albrecht, S. (2015). *Teaching algorithmic problem-solving through games*. Journal of Computing Sciences in Colleges, 30(6), 37-44.
- Gibson, W. (2002). *The History of Tic-Tac-Toe: A Look at Ancient Games and their Influence*. Journal of Game History, 1(1), 15-25.
- Hassenzahl, M. (2010). *Experience design: Technology for all the right reasons*. Synthesis Lectures on Human-Centered Informatics, 3(1), 1-95.
- Johnson, R., Preece, J., & McGrath, P. (2006). *Usability engineering for interactive systems: Understanding users and designing user interfaces*. Cambridge University Press.
- Knuth, D., & Moore, R. (1975). *An analysis of Alpha-Beta Pruning*. Artificial Intelligence, 6(4), 293-326.
- Kocsis, L., & Szepesvári, C. (2006). *Bandit based Monte Carlo planning*. Proceedings of the 17th European Conference on Machine Learning, 282-293.
- Luger, G. F. (2009). *Artificial Intelligence: Structures and Strategies for Solving Complex Problems* (5th ed.). Pearson.
- Osborne, M. J., & Rubinstein, A. (1994). *A Course in Game Theory*. MIT Press.
- Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson Education.
- Shannon, C. E. (1950). *Programming a computer for playing chess*. Philosophical Magazine, 41(314), 256-275.