COLLEGE OF ENGINEERING, GUINDY

Chennai - 600025

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY - IST

Mini Project Report on

Tic-Tac-Toe Game

in

DATA STRUCTURES - IT23302

Submitted by

Navinesharan S – 2023115015

Gowsika Sri S – 2023115028

Mohan Vishnu Kumar M – 2023115026

Tac-Toe Engine: A Data-Driven Framework with Adaptive AI, Multiplayer Access, and Real-Time Undo

Abstract

This project presents an advanced implementation of the classic Tic-Tac-Toe game using C programming, enhanced with modern computational techniques and efficient data structures. The game is designed to provide an engaging and interactive experience through features such as customizable AI difficulty levels, multiplayer access, undo functionality, and player statistics management. The project leverages an array of advanced data structures, including **arrays**, **stacks**, **linked lists**, **decision trees**, and **hash tables**, each serving a critical role in enhancing gameplay and system efficiency.

Key techniques like the **Minimax Algorithm** and decision-tree-based evaluation enable strategic AI decision-making, while the use of stack ensures smooth undo functionality and seamless concurrency control. This project emphasizes the practical application of data structures and algorithms, showcasing how theoretical concepts can lead to robust, user-friendly software systems. It also serves as an educational tool for understanding the interplay between algorithms and data structures in real-world applications.

Introduction

Tic-Tac-Toe is a universally recognized two-player game that is simple to play and challenging to master. While traditionally played on paper, this project reimagines the game in a digital format, introducing advanced features that enhance its interactivity and usability. The inclusion of **artificial intelligence** (AI) for single-player modes allows users to test their skills against a strategic opponent, while multiplayer functionality ensures engaging gameplay for friends and family.

This project stands out because it emphasizes utilizing core data structures and algorithms to address challenges such as move validation, multiplayer access, and AI decision-making. By integrating structures like decision trees for AI evaluations, the game provides a smooth, user-friendly experience while demonstrating the practical use of theoretical computing principles.

The primary objectives of this project are,

1. To create an engaging Tic-Tac-Toe game with robust AI and user-friendly features.
2. To implement various data structures in meaningful and efficient ways.
3. To highlight the importance of computational thinking in solving common software development challenges.

Data Structures Used

This section provides a detailed overview of the project's data structures and their significance.

Data Structure	Purpose in the Project	Reason for Use
Array	Represent the 3x3 game board.	Provides simple and constant-time access for updating and checking the board state. Efficient for fixed-size grids.
Stack	Implement the undo functionality.	LIFO property ensures efficient tracking and reversal of the most recent moves for both the player and AI.
Linked List	Manage dynamic game session data (e.g., storing past game results and session details).	Dynamic memory allocation avoids memory wastage and allows efficient addition/removal of game sessions.
Hash Table	Store player statistics, indexed by player names.	Offers constant-time complexity for lookups and updates, preventing duplicate entries for players with the same name.
Decision Tree	Facilitate AI decision-making in Medium and Impossible modes by simulating potential game states.	Enables logical and strategic evaluation of moves, predicting outcomes several steps ahead to maximize AI performance.

Features of the Game

The game integrates several features designed to enhance interactivity, challenge, and user engagement.

1. Interactive Menus

The game includes a **main menu** with the following options:

- **Play Game*:*** Opens a submenu to select single-player or multiplayer mode.
- **View Last Game's History**: Displays the sequence of moves in the last played game.
- **Player Statistics**: Displays win/loss/draw statistics for all players, including AI.
- **Exit**: Exits the game.

In the **Play Game** submenu, players can choose:

- **Single-Player Mode**: Includes three AI difficulty levels (Easy, Medium, Impossible).
- **Multiplayer Mode**: Two players take turns manually.

---

2. AI Difficulty Levels

- **Easy Mode**: AI selects random moves with no strategy, suitable for beginners.
- **Medium Mode**: AI uses decision-tree logic to block winning moves and make moderately strategic choices.
- **Impossible Mode**: Implements the Minimax Algorithm with optimization, ensuring unbeatable gameplay.

---

3. Undo Functionality

- Players can undo their last move in Easy and Impossible modes.
- Both the player's and AI's last moves are removed, allowing players to retry their strategy.
- Stacks efficiently handle undo operations by tracking moves in LIFO order.

---

4. Multiplayer Accessibility

- Allows two players to compete against each other.
- Players can play with their names and their scores get updated in the hash table.

---

5. Dynamic Game Session Tracking

- Linked lists store game session details dynamically, including results and player details.
- Nodes are added for each game session, ensuring flexibility and efficient memory usage.

6. Player Statistics

- Hash tables store records for all players, indexed by their names.
- Includes total wins, losses, and draws for both human players and AI.
- Efficiently manages player data for long-term gameplay sessions.

Techniques Implemented

1. Minimax Algorithm

The **Minimax Algorithm** is a classic decision-making algorithm used in game theory. In the Impossible mode, it ensures that the AI evaluates all possible moves and counter-moves to select the one that maximizes its chances of winning or forces a draw. This involves:

- Creating a **decision tree** of all potential game states.
- Assigning scores to terminal states (win, loss, draw).
- Propagating these scores back up the tree to determine the optimal move for the AI.

2. Decision Tree

- Decision trees represent the game's possible states for Medium and Impossible modes.
- AI evaluates the tree to make informed decisions, blocking winning moves and prioritizing its victory.

3. Undo Functionality

- Implemented using stacks, enabling players to reverse their moves efficiently.
- Tracks the sequence of player and AI moves for easy rollback.
- This dynamic and efficient approach provides players with greater control during gameplay.

4. Multiplayer Access

- Two players can play this game simultaneously, with their names and the game manages their data in different areas in the hash table.

5. Dynamic Session Management

- Linked lists manage multiple game sessions, dynamically allocating memory for each session.

6. Player Statistics

- Hash tables ensure fast and efficient tracking of player records, avoiding duplication.

Conclusion

This project successfully demonstrates the power of data structures in creating an engaging, interactive, and intelligent gaming system. The integration of stacks, queues, linked lists, decision trees, hash tables, and arrays highlights how theoretical concepts can be applied to solve real-world problems in software development. The result is a robust, feature-rich Tic-Tac-Toe game that offers challenges for players of all skill levels.

References

1. Artificial Intelligence: A Modern Approach – Stuart Russell, Peter Norvig
2. Data Structures and Algorithms in C – Mark Allen Weiss
3. Data Structures Using C – Reema Thareja
4. Decision Trees in Game AI – Journal of Computational Intelligence

---