Advanced Tic Tac Toe

Performance Specifications And Optimization Efforts

By

The X Factor

1.Optimization Efforts:

A)Optimization efforts made in the single player (player versus AI) mode:

➢ Alpha pruning is a significant optimization technique applied to the AI opponent in our Tic-Tac-Toe game. This technique enhances the decision-making efficiency of the minimax algorithm, which is commonly used for two-player turn-based games. Let's first discuss the Minimax algorithm.

➢ The minimax algorithm evaluates potential future moves, assuming both players play optimally. It generates a game tree like structure (using recursion) where each node represents a game state, and each branch represents a possible move. The algorithm assigns scores to terminal states (win, loss, or draw) and propagates these scores back up the tree to decide the best move at the current state.

➢ Alpha pruning optimizes the minimax algorithm by reducing the number of nodes evaluated. This is achieved by eliminating branches that cannot influence the final decision. Alpha represents the best score the maximizing player (AI) can guarantee, while beta represents the best score the minimizing player (opponent) can guarantee.

➢ Alpha pruning is used in a Tic-tac-toe AI opponent, so initially the AI begins evaluating the first move with alpha at negative infinity and beta at positive infinity, then the AI updates alpha for better moves for itself and beta for better moves for the opponent, if a move's score makes further evaluation redundant (worse than alpha for the AI or better than beta for the opponent), the branch is pruned, which ultimately leads to the best game choices made in the shortest time possible.

B)Optimizations in the database implementations:

➢ Using SQLite for implementing the database is very beneficial in terms of time and performance when compared to its counter parts like SQL and MySQL which are used for more complex applications and require servers to uphold connections.

➢ Performance preservation while protecting the players' passwords instead of using an encryption algorithm, the passwords are stored in a whole separate table that is not, like the rest of the tables in the databases, indexed by the usernames of the players as the primary key but it is indexed by the player number that is obtained implicitly from the username using a private method in the "database_t" class.

➢ In the project implementation using the database functions we use the Add_To_Score_Table() when the sign up button to add a new player to the Score table and initialize the number of games won, lost and draw to zero so that when a match concludes we are able to call Update_Score_Table() without checking if the player is a new player or not.

2.Time, CPU and Memory Usage Measurement:

A)During regular sign-up operation:

1- Time taken = 85.1363 milliseconds.
2.CPU usage is almost 0%
3.Memory Usage is 72 bytes

B)During Logging in one player to play against AI:

1.Time taken = 5.0269 milliseconds
2.CPU usage almost 0%
3.Memory Usage = 72 bytes

C)During logging in two players to play against each other:

1.Time taken = 5.1787 milliseconds
2.CPU usage almost 0%
3.Memory usage = 96 bytes

D)During AI opponent's move calculation:

1.Average time taken = 12.8056 milliseconds
2.CPU usage almost 0%
3.Memory usage = 24 bytes

E)General Specifications:

1.Application total size = 58.5 Mega bytes.
2.Application has extremely low overall CPU usage.