**1. Introduction**

This project implements an intelligent Tic-Tac-Toe game using:

* Minimax algorithm
* Alpha-Beta Pruning optimization
* Python
* User-friendly GUI (Graphical User Interface)

**Goals**:

* Understand and implement Minimax.
* Apply Alpha-Beta Pruning to improve performance.
* Integrate both into a GUI-based Tic-Tac-Toe.
* Compare performance: Minimax vs Alpha-Beta.

**2. Algorithms Used**

**➤ Minimax Algorithm**

* Recursive decision-making.
* Explores all possible moves to find the best one.
* Assumes opponent also plays optimally.
* **Time Complexity**: O(b^d) (branching factor b, depth d).

**➤ Alpha-Beta Pruning**

* Optimized version of Minimax.
* Prunes unnecessary branches.
* Much faster.
* **Best Case Time Complexity**: O(b^(d/2)).

**3. Code Structure**

**tictactoe\_ai.py (Game Logic and AI)**

* TicTacToe class:
  + Board state, moves, win/draw detection.
* AlphaBetaAI class:
  + AI move selection using Alpha-Beta Pruning.

**tictactoe\_gui.py (Graphical User Interface)**

* Built with Tkinter.
* 3x3 clickable grid.
* Human plays 'X', AI plays 'O'.
* Winner/draw messages via popup.
* Centered, resizable, colorful layout.

**4. GUI Features**

* Centered minimalistic layout.
* Button colors:
  + 'X' = Blue
  + 'O' = Green
* Responsive AI move (short delay).
* Board disables after game ends.
* Also has restart functionality.

**5. How the Game Works**

1. Human plays 'X' first.
2. Each move triggers the AI to move (Alpha-Beta AI).
3. Board checks for winner or draw after each turn.
4. End result shown via popup and disables the board.

**6. Minimax vs Alpha-Beta Performance**

| **Criteria** | **Minimax** | **Alpha-Beta Pruning** |
| --- | --- | --- |
| Nodes explored | High | Fewer (pruned) |
| Speed | Slower | Faster |
| Optimal move | Yes | Yes |
| Efficiency (deep tree) | Poor | Much better |

Alpha-Beta Pruning drastically reduces computation but still returns the optimal result!

**7. Requirements**

* Python 3.x
* Tkinter (comes pre-installed with Python)
* To play:
  + Run tictactoe\_gui.py

**Code Snippets**

**tictactoe\_ai.py**

* TicTacToe Class
* MinimaxAI Class
* AlphaBetaAI Class
* Performance Comparison (compare\_algorithms())

(Minimax and Alpha-Beta logic you already saw — full class structure.)

**tictactoe\_gui.py**

* Tkinter window (root)
* 3x3 Grid of Buttons
* TicTacToeGUI Class handles:
  + Player move
  + AI move
  + End game detection and messagebox

**Output**

* Program compares Minimax vs Alpha-Beta Pruning.
* Then GUI launches for human vs AI play.