

### **1. Problem Description**

Ultimate Tic-Tac-Toe is a complex extension of the classic Tic-Tac-Toe game. The board consists of a 3×3 grid of smaller 3×3 Tic-Tac-Toe boards. Players alternate turns, marking positions on the smaller boards to form a winning line on the larger grid.

#### **Rules:**

* A player wins a small board by aligning marks in a row, column, or diagonal.
* Winning three small boards in a line on the larger board wins the game.
* The next move's active board is determined by the position of the last mark.
* If the targeted small board is won or full, the player can move to any empty cell.

### **2. Design Overview**

#### **Core Components:**

1. **UltimateTicTacToe Class**:  
   * Handles the game logic, rules, and board state.
   * Implements methods for making moves, checking wins, and validating legality.
2. **UltimateTicTacToeGUI Class**:  
   * Uses tkinter to create a graphical interface for the game.
   * Manages user interactions, updates the board display, and integrates AI logic.

### **3. Game Logic:**

#### **Key Methods in UltimateTicTacToe:**

* **make\_move(i, j, x, y)**: Places the current player's mark at (i, j, x, y) if the move is legal, updates the game state, and switches players.
* **is\_legal\_move(i, j, x, y)**: Validates moves based on the current board state and rules.
* **check\_small\_board\_win(i, j)**: Determines if a player has won a small board (i, j) or if it is a draw.
* **check\_large\_board\_win()**: Determines if a player has won the game on the larger board.
* **get\_best\_move\_for\_player(player)**: Implements a basic AI to choose optimal moves using forward checking and minimax algorithms.

### **4. Artificial Intelligence:**

The AI agent is implemented within the UltimateTicTacToe class and utilizes:

* **Minimax Algorithm**: Recursively evaluates moves to maximize the player's chance of winning.
* **Forward Checking**: Ensures valid states by pruning illegal moves early.
* **Heuristic Evaluation**: Assigns scores based on the state of the boards to determine the optimal move.
* **Depth-Limited Search**: Limits the depth to prevent excessive computation during the minimax evaluation.

### **5. GUI Features:**

#### **Interface Design:**

* A 3×3 grid of smaller 3×3 boards is displayed using tkinter.Canvas.
* Players can interact with the game via mouse clicks to place marks.

#### **Interactive Elements:**

* **Dynamic Board Updates**: The board visually updates after each move, reflecting the current state.
* **Turn Indicator**: Displays which player's turn it is ("Your turn (X)" or "AI's turn (O)").
* **Endgame Alerts**: Displays the winner or draw message using tkinter.messagebox.

### **6. Strengths of the Implementation:**

1. **Rule Compliance**: Adheres to all the rules of Ultimate Tic-Tac-Toe.
2. **AI Integration**: Includes a functional AI agent to play against.
3. **GUI**: Provides an intuitive and user-friendly interface for gameplay.
4. **Scalability**: Code structure is modular, making it easy to enhance (e.g., adding keyboard navigation or animations).

### **7. Areas for Improvement:**

1. **Keyboard Interaction**: Replace mouse-based input with keyboard navigation for accessibility.
2. **Performance Optimization**: AI computation can be enhanced with techniques like alpha-beta pruning.
3. **Aesthetic Enhancements**: Use color coding and animations to improve the visual appeal.

### **8. How the Implementation Matches Assignment Requirements**

The implementation matches the following assignment tasks from the provided PDF​:

* **Task 1 (Constraint Formulation)**:  
  + The game rules are encoded as constraints in methods like is\_legal\_move and check\_small\_board\_win.
* **Task 2 (CSP Solver)**:  
  + The AI uses backtracking with forward checking and basic constraint satisfaction.
* **Task 3 (Experimentation)**:  
  + Implements minimax with heuristic evaluation and allows human-AI interaction for testing.

### **9. Conclusion**

This implementation provides a fully functional Ultimate Tic-Tac-Toe game with:

* A well-structured game logic.
* A basic AI opponent.
* An interactive graphical interface.

Further enhancements, particularly in AI optimization and GUI aesthetics, could elevate the overall experience and efficiency.