MiniChess 5x6 Game Development Report

Project Title: ChessChamp – A MiniChess (5x6) Engine with GUI and AI

Overview:
This project presents the development of a MiniChess game engine on a 5x6 board. The implementation includes a complete GUI using Pygame, a player setup menu, customizable themes, undo/redo functionality, and an AI opponent powered by a minimax algorithm with alpha-beta pruning. The game supports both human-vs-human and human-vs-AI modes.

Modules and Responsibilities:

1. **main.py**
    - Entry point of the game.
    - Initializes the game engine and starts the main loop.
2. **game.py**
    - Contains the `MiniChess5x6` class responsible for managing game states.
    - Handles transitions between menus, gameplay, AI moves, undo/redo, and theme selection.
    - Integrates the GUI and game logic.
3. **ui.py**
    - Handles all GUI drawing: main menu, theme menu, game board, and game-over screens.
    - Manages UI button interactions and renders chess pieces using Unicode.
    - Offers multiple theme options and highlights game states such as valid moves and selected pieces.
4. **game_setup.py**
    - Displays a setup menu allowing users to choose player types (human or AI) for white and black.
    - Returns selected configuration to `game.py` before game start.
5. **board.py**
    - Implements the chessboard and core game logic (e.g., legal moves, check/checkmate, stalemate).
    - Supports piece movements, validations, and history tracking for undo/redo.
    - Manages turn switching and game-end conditions.
6. **ai.py**
    - Implements the `MiniChessAI` class using a depth-limited minimax algorithm with alpha-beta pruning. The AI searches to a fixed depth of **3**, using **iterative deepening** within a **1-second time limit** to ensure responsiveness.
    - The AI selects the best move using a combination of the following:
        - **Minimax Search**: Explores all legal move sequences up to depth 3.
        - **Alpha-Beta Pruning**: Prunes branches that can't improve the result, reducing the number of evaluations.
        - **Evaluation Function**: Scores each board state based on:

            - Material count using standard piece values.
            - Control of central squares (bonus for occupying key positions like (2,2), (2,3), etc.).
            - Mobility (number of legal moves available to the current player vs. opponent).
            - King safety (bonus if the king is well-positioned or back-ranked).
            - Tactical vulnerability (if the moved piece becomes threatened) and aggression (proximity to enemy pieces).
        - The AI evaluates each move and selects the one with the highest score after searching the game tree.
    - The search loop terminates early if the 1-second budget is reached, ensuring the AI always returns a move within the allowed timeframe.
7. **constants.py**
    - Stores all constants: colors, board sizes, fonts, piece Unicode, and theme configurations.

Key Features:

- **Graphical User Interface:** Built with Pygame, featuring a modern, responsive design with theme customization.
- **MiniChess Ruleset:** Standard pieces placed on a 5x6 board, including check, checkmate, and stalemate detection.
- **Undo/Redo:** Navigation through the move history stack.
- **AI Integration:** A fast, strategic AI opponent using optimized search and evaluation functions.
- **Custom Themes:** 16 unique board themes for visual diversity.

AI Evaluation Function Highlights:

- Material count with predefined values.
- Center control bonus for occupying key squares.
- Mobility score based on available legal moves.
- Proximity to threats and king placement heuristics.
- Evaluation adjusts based on tactical risks and aggressive positions.

Challenges Faced:

- Balancing AI thinking time and responsiveness.
- Preventing illegal moves like moving into check.
- Implementing an intuitive GUI for player interactions.

Conclusion:
ChessChamp successfully brings the MiniChess variant to life with polished visuals and intelligent gameplay. It allows users to enjoy rapid matches against human or AI opponents with options for customization and replays. The architecture is modular, making future extensions (like networked multiplayer or enhanced AI depth) feasible.

Future Improvements:

- Add support for move notation and PGN export.
- Introduce difficulty levels for AI.
- Implement a tutorial mode or hints for beginners.
- Animate piece movement for enhanced user experience.