Chess Game Project

Nguyễn Hoàng Thuận Phát, Huỳnh Tấn Phúc

November 21, 2024

Overview

The project aims to develop a chess game that is smooth, lightweight in size and performance while providing decent experience to player.

1 REQUIREMENT

1.1 Game modes

- 2-player mode
  - 2 players can play against each other on the same instance
  - Player's turn is managed automatically, alternating between player 1 (white) and player 2 (black)
- Versus AI mode: Player as white pieces plays against black pieces controlled by Stockfish chess engine, with three level of difficulty
  - Easy mode:
  - Medium mode:
  - Hard mode:

1.2 User Interface(UI)

User interface and its elements is created with SDL2 graphic libraries and nanoSVG for SVG image file conversion

- A chessboard with movable pieces
- Chessboard's colors and pieces's styles can be chosen from different color palettes.
- Menu before the game is started
  - Start button that start the game.
  - Load button to load saved board and gameplay from storage
  - Quit button to exit the game.
- Menu during gameplay is render to the ? of the board.
  - Promotion of player's pawn is shown in a pop-up menu, which displays four pieces of rook, knight, bishop and queen.
  - Buttons to reset for the new game, undo last moves, redo last undo operations, save the current game and load a game from storage.
  - Button to open menu for customization
    - asa
    - as
    - as
  - Quit game button

1.3 Chess board representation

- Chessboard is drawn using SDL2 graphic libraries.
- Pieces are first parsed to bitmap using nanoSVG library's function, then rendered by SDL2
- Pieces are moved by drag-dropping with mouse.
- Legal moves are highlighted with cell-centered dots for moves and hollow circles for captures. The rendering of legal moves are triggered when player click on a piece or drop a piece to its original cell.
- Illegal moves (including movement to cells that are not highlighted, movement out of board's bound, etc) will return the piece to its original cell and does not count as a move.
- King's position is highlighted with different colors for its check, checkmate and stalemate statuses.
- Moves are managed automatically, alternating between two colors after each move.

1.4 Game Logic

- Pieces' movement
  - Chess pieces' moves are generated on selection based on their position, then get processed to create list of moves and capture in each direction.
  - Special moves' condition are reviewed after each move, based on their condition they will get added to suitable pieces in the next move.
  - Status of king is examined during moves generating and after each move.
- Selected cell's ending cell is compared in its legal move and capture list. Each move is simulated before returning the list to ensure this move does not put the king in danger position(check, checkmate).
- In case of stalemate or checkmate, the function of moving chess pieces refused to work. In such case, (need addition for this)

1.5 AI Opponent

- something
  - something

1.6 Game state management

The game is managed using the combination of 2D array board representation and FEN notation, the latter stores information needed to reconstruct the board from nothing. Turn management is done right after a legal move had been made. After each legal move, both array representation and FEN notation is updated. For undo and redo features, deque of FEN notation is used to maintain the maximum execution of each function. FEN notation of initial state is also saved as a constant string for creation of new game.

1.7 Save and Load Functionality

to be added

1.8 Optional features

While being dragged, selected chess piece follows the mouse's cursor smoothly.

1.9 Technology stack

The game's logic, game state management and GUI(graphic user interface) is written in C and C++, with SDL2 and nanoSVG libraries for graphic. The AI used for player versus computer mode is Stockfish, a popular and strong open-source AI chess engine. Current board state is written to plain text file in FEN notation, which can be decoded to get the complete information of board state stored.

More to be written

2 Design document

To be written after finished

3 Implementation Details

The code in organized in separate header and implementation files. nanoSVG libraries is used for converting vector images into bitmap images. SDL2 is used to render bitmap images and

4 Testing

5 Future Improvements

1. Implementation of board rotation for better user experience.
2. Fix the inconsistency between x-axis, y-axis of screen and row, column of 2D array representation. The inconsistency made methods seemed confusing to use.