Chess Game Project

Nguyễn Hoàng Thuận Phát, Huỳnh Tấn Phúc

December 7, 2024

Overview

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

Chapter 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: Stockfish is ran with depth
  - Medium mode: Stockfish is ran with depth
  - Hard mode: Stockfish is ran with depth

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 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 right of the board.
  - Promotion of player's pawn is shown above or under that pawn, which displays four possible promotion to queen, knight, rook and bishop.

  - Buttons to reset the board to new game state, undo last moves, redo last undo operations, save the current game and load a game from external storage.
  - Customization
    * Button to cycle between modern, futuristic and classic board theme
    * Button to go backward to the original board state, that is, the board state right after loading from save file or right after starting new game.
    * Button to go forward to the latest made move.
  - 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.
- Moves are managed automatically, alternating between two colors after each move.
- King piece's cell is highlighted with different colors for its check, checkmate and stalemate statuses.
  - To highlight chess piece's movement, color of initial and ending cells of that piece is mixed with a yellowish color.
  - To highlight check status of king piece, color of its cell is mixed with a redish color.
  - To highlight checkmate status of king piece, color of its cell is replaced with a bright red color.
  - To highlight stalemate status of king piece, color of its cell is replaced with a cyan color.

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 all special moves.

- Selected cell's ending cell is compared in its legal moves and captures list. Each move is simulated while creating the list to ensure that move does not put the king in check.
- In case of stalemate or checkmate, any chess piece cannot be moved. In such case, (need addition for this)

1.5 AI Opponent

Stockfish is initialized before running with

```
uci
setoption name Hash value 128
setoption name Threads value 1
isready
setoption name UCI_LimitStrength value true
```

For each time the game is started from new board or from save file, command `ucinewgame` is sent to Stockfish

There are three modes of AI opponent lining up with three difficulty: Easy, Medium and Hard.

- Easy difficulty: Stockfish is configured with `setoption name UCI_Elo value 1320` and execute `go depth 2 movetime 300` after setting FEN notation for board state.
- Medium difficulty: Stockfish is configured with `setoption name UCI_Elo value 2000` and execute `go depth 5 movetime 500` after setting FEN notation for board state.
- Hard difficulty: Stockfish is configured with `setoption name UCI_Elo value 3190` and execute `go depth 10 movetime 1000` after setting FEN notation for board state.

After letting Stockfish executing commands, the communicator wait for 500 miliseconds before reading the result from `tmp/stockfish_output.txt`

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

1.7.1 Save

1.7.2 Load

There are separate menu dedicated for loading. In that menu, available save files are displayed in cards on the left side. On hovering, the card being hovered will be highlighted, and the board state that is stored within the file that the card stand for is displayed to the right of the game windows as a preview. When user clicks on a card, the board immediately load the respective save file.

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++, using 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.

Chapter 2

Design document

To be written after finished

Chapter 3

Implementation Details

3.1 Organization

The code is organized by header - implementation files. General logic is handled in main file, while detailed logic is handled within each class.

3.2 Libraries

The program used SDL2 and NanoSVG libraries to handle the graphic interface. To execute algorithms of finding suitable move of computer, Stockfish engine along with its detailed configurations are used

3.3 Major component

There are three major component: General logic, board, and communicator.

- General logic is responsible for handling interactions between user and game window. It is also responsible for displaying chess board and other components of the game.
- Board component is responsible for handling interactions between pieces and interactions between user and chess pieces. It also send necessary information back to general logic component.
- Communicator component is responsible for communicating between the game and Stockfish in order to find appropriate move in AI's turn.

Chapter 4

Testing

The project is tested by playtesting, in which simple games are played and all interaction that player might do is performed in order to detect bugs and ensure good experience.

Chapter 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 board's methods seems confusing to use.