Chess Game Project

Nguyễn Hoàng Thuận Phát, Huỳnh Tấn Phúc

Submission day: December 18, 2024

Overview

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

Contents

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 high variation and low evaluation level. It searches for next move to depth 1 in 100 milisecods, and has limited computing resource.
  - Medium mode: Stockfish is ran with moderate variation and moderate evaluation level.Stockfish searches for next move to depth 3 in 400 miliseconds and more computing resource are allocated
  - Hard mode. Stockfish are given the most computing resource. It plays perfectly without variation, searches to depth 20 in 2000 miliseconds

  AI's turn is delayed for 500 miliseconds in addition to Stockfish's search time to avoid runtime errors.

1.2 User Interface(UI)

- A chessboard with movable pieces
- A start menu that renders before a game is started
  - Start buttons that start the game in two-player mode and single player mode. If the game is started in single player mode, there is a popup menu for the user to select the difficulty of AI.
  - Load button to load saved board states from storage.
  - Quit button to exit the game.
- An in-game menu is rendered to the right and bottom portions of the board.
  - Promotion menu is rendered to the top or bottom of pawn under promotion, with four options to promote to queen, bishop, kngiht or rook.

  - Right menu bar
    * Button to save and load current board configuration, including board state, game mode and AI diffculty if there is.
    * Button to access setting menu
  - Bottom menu bar
    * Button to undo the latest move and undo to the initial board state.
    * Button to redo the latest undo and redo to the latest board state.

- A load menu displays buttons to load saved games, each of which shows previews of board when hovered and load the actual game when clicked.
- A save menu displays buttons to save current game. On hover, each button display preview of saved game that is going to be overwritten if current game is saved. When pressed, a confirmation menu appears to ask if user want to overwrite. There are also a clear button for each save file to delete that save.
- A setting menu has buttons of switching theme along with the theme's designated pieces set, adjust volume of all sound effect, mute and unmute sound, reset the game to starting position, and quit to menu.

1.3 Chess board representation

- Chessboard is drawn using SDL2 graphic libraries.
- Pieces are first parsed to bitmap using nanoSVG library's function, then rasterized and rendered by SDL2's functions.
- 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. Current player's turn is displayed by an indicator on the top portion of the screen.
- 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 mixed 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, a pop-up menu is displayed to ask if user want to continue another game. If yes button is clicked, the game reset to starting configration with the current game mode. If no button is clicked, the game quits. If the menu is turned off, the game still function as normal, yet the board cannot be interacted with.

1.5 AI Opponent

Stockfish is initialized before running with

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

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 Skill level value 0
  setoption name Threads value 1
  setoption name Hash value 16
  setoption name MultiPV value 10
  ```

  and execute `go depth 2 movetime 300` after setting FEN notation for board state.

- Medium difficulty: Stockfish is configured with

  ```
  setoption name Skill level value 3
  setoption name Threads value 2
  setoption name Hash value 64
  setoption name MultiPV value 5
  ```

  and execute `go depth 3 movetime 400` after setting FEN notation for board state.

- Hard difficulty: Stockfish is configured with

```
setoption name Skill level value 20
setoption name Threads value 8
setoption name Hash value 2048
setoption name MultiPV value 1
```

and execute `go depth 20 movetime 2000` 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 scratch. Turn management is done right after a legal move had been made. After each legal move, both array representation and FEN notation of the board is updated.

For undo and redo features, two stacks of FEN notation is used to store information for those operation, with the top of undo stack is the current board state. 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

There is separate menu dedicated for saving. 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. When user clicks on a card, the confirmation box appears, asking if user want to overwrite the save file. If the user confirms, the game will proceed to overwrite that save file, otherwise the confirmation box will close.

1.7.2 Load

There is 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, followed by numbers indicating current game mode and difficulty of AI opponent.

Chapter 2

Design document

2.1 System requirement and build

Currently, the game is only developed and tested on Windows operating system. To compile the program from source files, the directory must first be at the same path as the makefile file, that is, the `CS160_Chess` folder. Also, makefile must be installed.

Then, user can build the project by running `./buildscriptwindows.ps1` or run `make all`. After that, the executable is named `chess.exe`, which is located at `CS160_Chess` folder. The executable will be automatically be ran after being built.

An alternative way for building the project if the above methods doesn't work: First, run

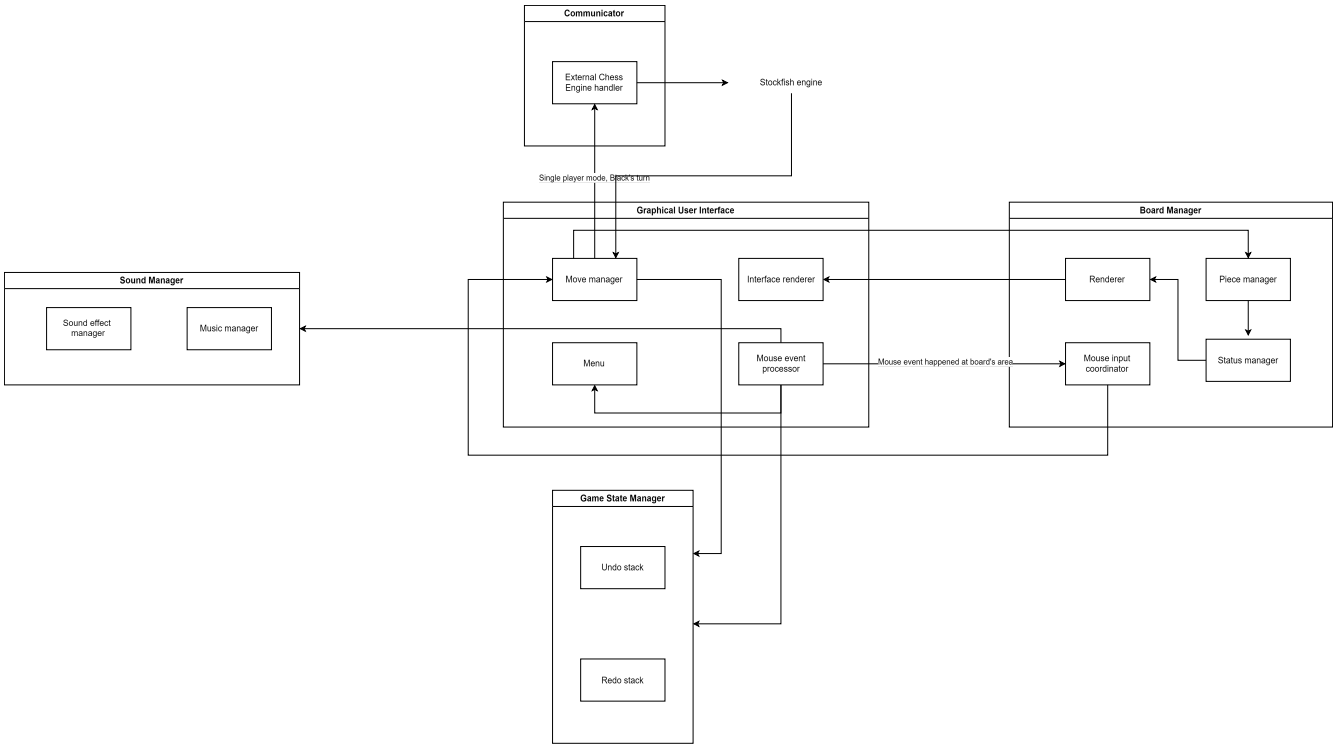
```
Get-ChildItem -Path src -Filter *.o | Remove-Item -Force | Get-ChildItem -Path src -Filter
    chess.exe | Remove-Item -Force
```

in powershell. Then, execute the following commamds:

```
g++ -c src/main.cpp -o src/main.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/board.cpp -o src/board.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/color.cpp -o src/color.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/colorScheme.cpp -o src/colorScheme.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -
    lSDL2_ttf -lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/
    SDL2/include/SDL2
g++ -c src/pieces.cpp -o src/pieces.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/coordinate.cpp -o src/coordinate.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf
     -lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/button.cpp -o src/button.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/communicator.cpp -o src/communicator.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -
    lSDL2_ttf -lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/
```

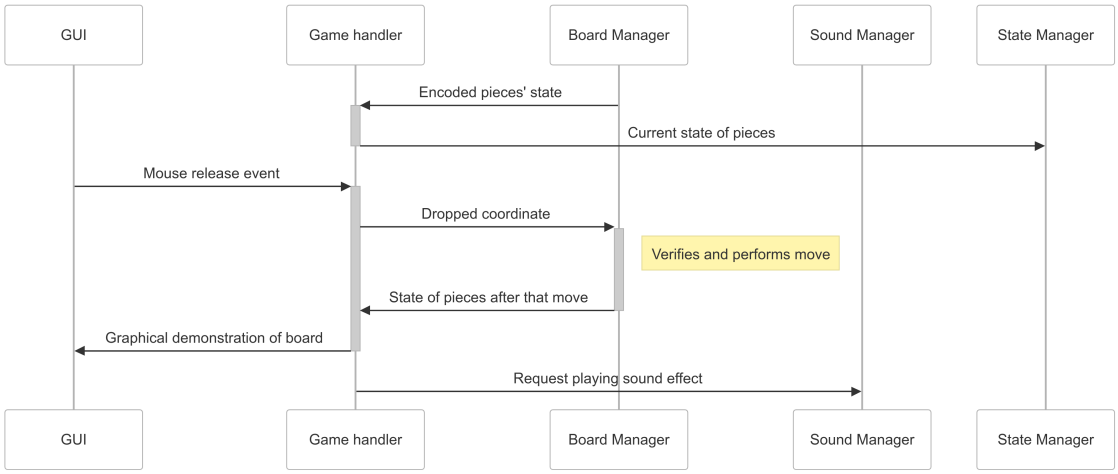
```
    SDL2/include/SDL2
g++ -c src/gameStateManager.cpp -o src/gameStateManager.o -std=c++20 -lmingw32 -lSDL2main -
    lSDL2 -lSDL2_ttf -lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include
     -Isrc/SDL2/include/SDL2
g++ -c src/popup.cpp -o src/popup.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/sound.cpp -o src/sound.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -c src/music.cpp -o src/music.o -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -
    lSDL2_image -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/
    include/SDL2
g++ -o chess $(wildcard src/*.o) -std=c++20 -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -lSDL2_image
     -lSDL2_mixer -Lsrc/SDL2/lib -Lsrc/SDL2/bin -Isrc/SDL2/include -Isrc/SDL2/include/SDL2
./chess
```

2.2 Architecture design

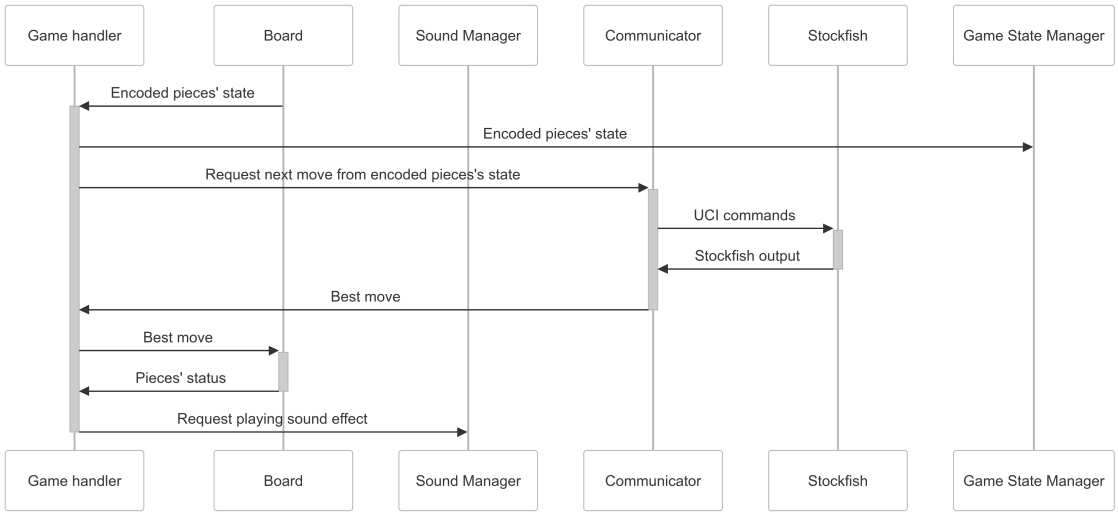


2.3 Processes

2.3.1 Player's move



2.3.2 AI's move



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. Stockfish is used to find the next move for AI opponent.

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.

While testing, we find that at any given moment, the game only run single task, that is, while the AI is evaluating best move any input is blocked. Also, the game does not have any feature of checking pawn that is eligible for promotion on loading saved game.

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.