ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

CS160 - INTRODUCTION TO COMPUTER SCIENCE

Final Project

Đề tài: Chess

Môn học: Introduction to Computer Science

Sinh viên thực hiện:

Huỳnh Chí Tôn (24125020)

Nguyễn Trọng Văn Viết(24125023)

Giáo viên hướng dẫn:

TPLộc

ĐNKha

Ngày 18 tháng 12 năm 2024

Mục lục

1 Overview:

Our work in Github: 24125020 24125023

Our work in Google Drive: 24125020 24125023

Creating a chess game with friendly-user interface and multiple modes of playing .

Using library SFML create user interface (UI) and chess.hpp as rule for chess .

2 Game mode:

2.1 2-Player Mode:

Two players can play against each other on the same device.

The game could manage turns automatically, alternating between Player 1 (white) and Player 2 (black).

2.2 Player vs AI Mode:

The user can select the color they want to play and choose between three difficulty levels of AI :

Easy: The AI makes completely random move with no strategy.

Medium: The AI uses minimax algorithms with depth 2 .

Hard: The AI uses minimax algorithms with depth 4.

3 User Interface (UI):

create a user-friendly interface using SFML library

Have a minimal home screen with buttons for player to navigate.

Buttons in the game : redo, undo, save the current state of the game and load the saved game.

Player can also customize the UI including :

Board color : Classic, Modern, Cartoon

Piece design : Classic, Modern, Cartoon

Sound effects : Checkmate, Check, Move, Capture

Background music : Mozard classical music

4 Chess Board Representation:

Using Sfml to create the UI

There are 64 pieces of chess pieces on the board with numbers of buttons on the right side for players to navigate

Player can select and move or drag with smooth animation

Highlight legal moves when selecting a piece

Having warning notifications when checking and pop-up when checkmate or stalemate

5 Game logic:

Utilize Chess.hpp library to create logic for pieces such as checkmate, stalemate, promotion, and capture.

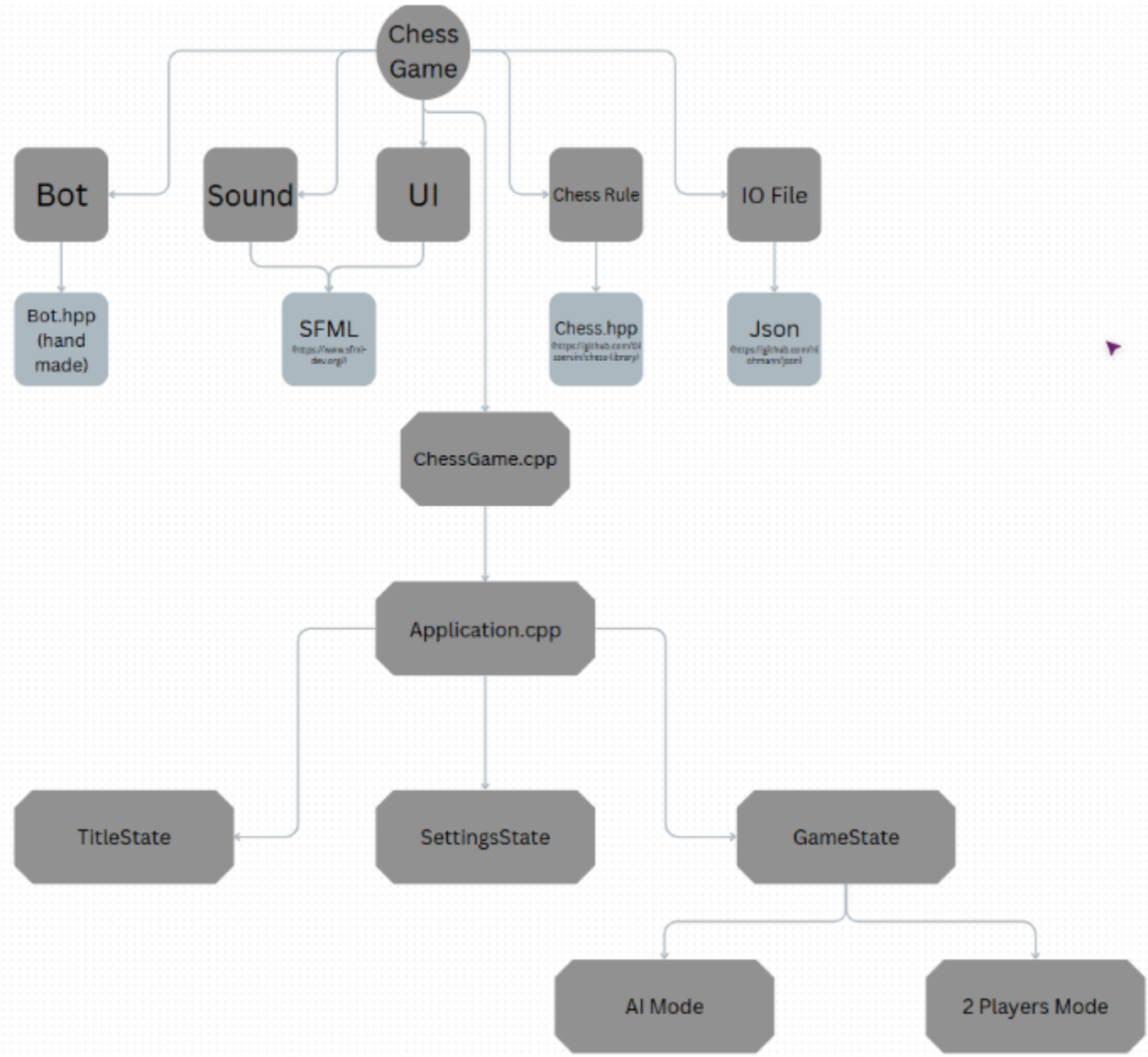
Ensure that you have no errors and no conflicts when playing

6 Game State Management:

Present a list of historical moves list to undo and redo if the player want by the undo and redo button on the screen .

Having a reset button to clear the historical novelist and go back to the starting position.

There is a save button on the screen for the player to download the state and legal moves list of the board into the storage on the device disk.



Hình 1: sơ đồ hoạt động

7 Implementation detail :

7.1 Code organization:

Scr/: contain sources for application

Include/: contain libraries for application

Media/: contain media files (sound, music, font, pieces photos, . . .)

Build/: Generated files during the build process

7.2 Libraries Used :

Standard C++ library

SFML for graphics

nlohmann/json for data management

Chess.hpp(Disservin/chess-library: C++ chess library) for game rule

7.3 Major Components:

7.3.1 Core Logic:

-For single player mode: Push the fencode of the current board to Bot.hpp and then it will throw a move. Move its turn and the player will make a new move. Afterward push the fencode of the current board to Bot.hpp again. Loop until endgame.

- For two player mode: 2 player make move respectively until endgame.

- Application is divided into many states. If you move from one state to another state e.g. TitleState to SettingState, TitleState still here, but SettingState is rendered above it and then we can only see SettingSate, after settings, SettingState will be popped and you can see TittleState. For more clear, The way I store State seems like a stack. A new state when an event occurs, will be pushback to the stateLists and If you are done working in that state, you go back, and that state will be popback from stateLists. When handling the event, I will handle it from top to the beginning, if an element is updated, I will stop that process. When drawing, I will draw from the beginning to the top.

7.3.2 User Interface and data management:

- Use SFML to manage events and render to screen.
- Use Json to save data from game.

8 Save and Load functionality:

Have save button on the screen for player to download the state and legals movelist of the board into the storage on device disk.

To open a saved game play can click the LOAD button to open a save game in the UI of the app

9 Optional Features:

Having a slide for move list history represented in algebraic chess notation

Smooth animations with elegant sound effects

Fast response time and straight forward for users to use

Handle input through click by mouse

10 Technology Stacks:

Using C++ as main language for all applications

Algorithms: using minimax with Alpha-Beta Prunning to make decision depending on level of difficulty

Using SFML for UI

Save game through Json File

Load and save game from a dialog appearing in the game

11 Testing:

Chess moves are sometimes unnatural because the best moves of the bot if there are many moves maintaining the value of pieces it will choose randomly.
When in mode Hard, Bot will move slower than normal because there are too many possible moves that can happen in some situations.
When AI Mode, when you redo, if you are slow, bot will make move, and the stack belonging to previous moves will be cleared and you cannot redo anymore.

12 Work division

Team members : Huỳnh Chí Tôn 24125020 Nguyễn Trọng Văn Viết 241250023
Responsibility : Huỳnh Chí Tôn : Game Logic , User interface , Testing
Nguyễn Trọng Văn Viết : AI development, Testing , Documentation
Collaboration : Git , Github , SFML , Chess.hpp , Visual studio.

13 Tutorials:

- Begining at Title Screen. (Play, Settings Sound, Exit game)
- In Play Screen
+ New game: create a new game
+ Load saved game: open an existing game (you can't undo, redo because we only save the game states except list move).
+ Back to Title Screen
- In Settings Screen:
+ You just can setting audio when you in title screen.
+ You just can setting music theme in title screen and setting music mission in game play.
- New game:
+ If you want to play with AI (1 player Mode): 1 player -> Choose Level (easy, medium, hard) -> Choose Team (white, black, random).
+ If you want to play 2 player Mode: 2 player

- You can flip the board by reversing button right under screen. (The reason why we think don't need to choose team in 2 players mode).
- Pay attention to top of the screen, it will announce that turn of what player, game states (check, check make).
- When you choose a piece, its cell will become white, all its legal move will become yellow. Last cell of already moved chess is red, and new cell of it is orange.
- You can reset game, undo, redo, save game, new game by clock on buttons right under screen. - Scroll to see made moves.
- Other options: press Esc(ape), you can settings (piece, audio, board), go to title screen (back to menu button).