CS 246: Object Oriented Programming

Assignment 5: Chess

By: Kriti Sodhi, Jemima Vijayasenan, Malvika Patel

December 6, 2022

Introduction

To begin, our group attempted to create a functioning chess game with text and graphics display for our final project. We attempted to utilize many of the course concepts such as leveraging polymorphism in our design to create a polished final product. Although the project took a considerable amount of time to complete, we are very proud of our final submission and everything we learned throughout the process.

Overview

We designed our program to follow the Model-View-Controller (MVC) model so that all user interactions are handled in the controllers, our main and game class, the view is displayed through the observer pattern in our board class, and the data-related logic is mainly handled in our pieces class.

- Talk about the main class
- Talk about the observer pattern in the board class
- Talk about the pieces
    - We leverage polymorphism through the Pieces base class and the concrete subclasses for each of the possible pieces, king, queen, rook, knight, bishop, and pawn.
    - Logic for check, checkmate, … is handled here

Updated UML

- Talk about changes from our

Design

- Mvc
- Observer method
- Decorator?

Resilience to Change

- Multi player

Answers to Questions

1. **Chess programs usually come with a book of standard opening move sequences, which list accepted opening moves and responses to opponents' moves, for the first dozen or so moves of the game. Although you are not required to support this, discuss how you would implement a book of standard openings if required.**

To implement a book of standard opening move sequences for our chess program, we would refer to the *FCO Fundamental Chess Openings* by Paul Van Der Sterren and insert popular opening move sequences into a map from the STL library. For example, we would have a map<string, string> key, value type that would store potential moves for each of the opening move sequences and potential responses to opponent moves. For example, an important strategy is to occupy as large a portion of the center as possible. Hence, the map would have various opening moves designed for this strategy. We would then have an assesMove function to evaluate opponent moves and call the appropriate key-value pair to respond to the move made by the opponent.

2. **How would you implement a feature that would allow a player to undo their last move? What about an unlimited number of undos?**

In order to implement an undo feature in our game of chess, we would have a list of all player moves stored within a vector. This vector would be created at the beginning of a game and, each time a player completes a valid move, a move object with the respective information about the player who made the move, the piece that was altered, whether the piece was captured or not, the new coordinates of the piece after the move, and the coordinates of the piece before the move would be pushed to the back of the vector. In order to undo the previous move when the player chooses to do so, we would pop back the last move off of the vector. Then, the information described above would be provided to the reverseMove function, which is responsible for undoing the move or potentially calling the uncapturePiece move for the case where the piece was captured before the undo. This solution demonstrates high cohesion and low coupling with each of the reverseMove and uncapturePiece functions completing one task and limited dependencies for the undo process. Finally, if we wanted to allow a player to undo an unlimited number of moves, we would have a for loop within our undo function that would apply the above process of undoing the move by popping off the respective number of moves from the vector one at a time.

3. **Variations on chess abound. For example, four-handed chess is a variant that is played by four players (search for it!). Outline the changes that would be necessary to make your program into a four-handed chess game.**

To change the implementation of the chess game to implement the four-handed chess variant, we would need to modify the board, the number of players, as well as the rules of the original game. Since we designed our game to easily plan for change using the MVC model, this task would just require small modifications to the existing model. In order to modify the board so that it changes from an 8x8 board to be extended by three rows of 8 squares on each side, we would just supply these changes in the width and height to our board class. This would allow our observers to get notified at the beginning of the game and change this layout.

Furthermore, the design of our Player class allows us to easily add or remove players in each game. To change the game from 2 players to 4 players, we would simply create 4 instances of our Player class instead of the classical 2-player instances with the additional colour enumerations of red, blue, yellow, and green. Finally, since we implemented all the logic of our game within modules of our Model, we would be able to easily add new rules or change existing rules with directives. This will allow us to toggle features that are different between the two variants of the game on and off to save compilation time.

Extra Credit Features

Final Questions

1. What lessons did this project teach you about developing software in teams? If you worked alone, what lessons did you learn about writing large programs?

Software development is a thorough process which requires software developers to plan, write, execute and test code rigorously. This project was the first step for the three of us to explore the world of software development. After the first coop, we all had a decent understanding of software development and its processes. However, by developing and deploying this project from scratch allowed us to use the software development life cycle to plan, create, test and deploy our chess implementation. We recognized that software development is an iterative process that consists of several changes, phases for building and running software applications. Since we were working on the project as a group, we used GitHub to share our code. Using GitHub allowed us to learn Git, a tool used for coordinating tasks amongst collaborators when developing source code. This is an essential skill we learnt because Git is a tool used widely in the software development industry.

   a. talk about debugging gdb
   b. give examples of git commands?

2. What would you have done differently if you had the chance to start over?

- Nvi
- pImpl
- No protected methods
- Unique ptrs from the start
- Plan for change: not hardcode 2 players
- Have a move generator from the beginning

Conclusion