CSC207 Group 55 Progress Report - Chess Game

Aidan Ryan, Ang Li, Giwon Kim, Jong Eun Won,
Kole Robertson, Matthew MacQuarrie-Cottle

15 October 2021

Domain

Create a Chess App on Android that implements Accounts, Records of games, Statistics, ELO, GUI, & LAN Multiplayer

Specification

Android app starts with user log in.

Users can login with their username and password, play as a guest or sign up.

After the user is logged in

Once Logged in users can view records and statistics based on previous games they have played. Users can also start a 1 v 1 games of Chess to be played locally on the device or over a LAN connection.

When the user starts a game, another player can log in with their username and password or play as a guest.

Chess

Chess will follow the standard rule set. Turn-based 1v1 game between white and black pieces, on a 8x8 board, with 16 pieces per player. Each turn, a player can select and move 1 piece. There are 6 different pieces (Pawn, Bishop, Knight, Rook, Queen, King) which all move in different ways. A player can capture an opposing piece by moving into their tile, with exceptions for the pawn and the king.

- Bishop: Can move diagonally
- Knight: Can move 2 tiles in one direction + 1 tile in a perpendicular direction
- Rook: Can move in Cardinal directions (NSEW)
- Queen: Can move in all directions (NSEW + Diagonally)
- Pawn: Can move 1 tile forward. Players can choose to move 2 tiles forward if it's the first time the pawn is moving. Can only take pieces forward that are one tile away diagonally.
- King: Can move 1 tile in any direction (NSEW + diagonally). Cannot move to tile if it would leave the king in check (a position in which it can be taken by an opposing piece) right after the player's turn.

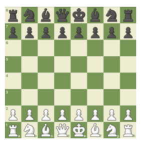
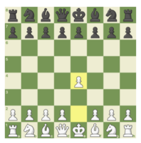
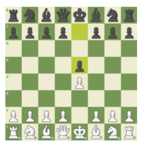
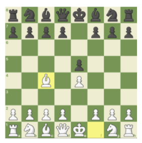
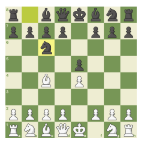
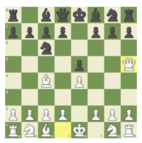
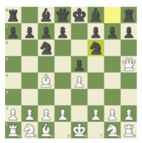
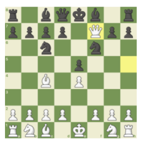
The game is won by putting the opposing King in a position where no matter what the opponent does on their next turn, the King can be taken. This is referred to as checkmate.

CRC Model

Our CRC Model contains the classes necessary to create a functional chess game. It includes the ChessGame Main Controller, Chessboard,

Scenario Walk-Through

White begins the game by moving the king's pawn forward two squares from E2 to E4. Black moves the king's pawn forward two squares from E7 to E5. White moves the king's bishop out 3 squares diagonally from F1 to C4. Black moves the knight from B8 to C6. Then White moves the Queen diagonally from D1 to H5. Black moves another knight on G8 to F6. White takes out Black pawn by moving Queen from H5 to F7. Black is now in checkmate, so the game ends.

GameStart	W:E2-E4	B:E7-E5	W:F1-C4	B:B8-C6	W:D1-H5	B:G8-F6	W:H5-F7
							

Skeleton Program

For our skeleton program the following classes were implemented:

- ChessGame (Main Controller)
  - Player (keeps track of each player's info)
- Chessboard (Chessboard made up of array of Tiles)
  - Tile (Keeps track of which piece is on itself)
- Piece (Abstract)
  - King, Knight, Pawn, Bishop, Queen, Rook, EmptyPiece (Child class of Piece)

To demonstrate functionality moves are made to depict Scholar's Mate, a commonly known outcome of a game of Chess. These moves can be seen above in our Scenario Walk-Through

What's Worked Well So Far With Our Design?

We tried to divide up the class as much as possible, which came out to be a very well-organized CRC model. It allows us to have a low coupling and high cohesion program.

Since it is the beginning of a project, everyone worked together instead of distributing the work because we thought that it is important to share each other's ideas for constructing a domain, specifications, scenario walk-through, CRC Cards, etc. For the skeleton program, Aidan worked on establishing the foundations for the logic behind the chess app, enough so to have a rough demonstration of intent. The skeleton program is lacking many of the major features we want from our app (online functionality, moves like castling and En passant, statistics tracking); however, due to the design of our app a lot of these things are secondary to the core purpose of the app and are highly modular in their implementation. With our current design, these things can be added on quite simply as long as the foundational logic of the chess game is strong.

Despite this, there are definitely still major improvements to be made to the code base; revisions to the movement system to account for different board sizes is a must, as well as simplifying some of the monstrous chunks of logic inside certain classes that not only make understanding the program hard but also can cause errors that have yet to unearth themselves in our *very* scarce testing.

What's Next?

The next big step in terms of the game itself is likely to smooth out and generalize the movement system to allow for different board sizes and for certain edge cases. We can't even capture targets with pawns yet! As mentioned previously, many of the features we want added on to the program are quite modular in their implementation, so development on the front end, on account logins and statistics tracking, and multiplayer systems can occur simultaneously.

Moving forward our plan is to have Aidan focus on statistics tracking and analysis, Kole focus on account logins and credentials, Matthew focus on front end on Android, Jong Eun investigate multiplayer systems and Giwon and Ang complete the chess game functionality.

Questions for the TA?

Should we build a text-based console chess game first then work on GUI?