Product Specification

Group 5

1 Description

The chess application will allow the user to play chess against another person locally, or against an AI which will have three different difficulties; easy, medium and hard. The application will also be able the track the results of the games and save the skill rating (using the Elo system) of the users and place them on a scoreboard. The AIs will be at a static rating for each difficulty level. The chess application also allow the user to play different game modes than the regular one. Horde, Light brigade and chess tutor. Horde chess is a variant where the white side has 36 Pawns and the black side with regular pieces needs to destroy the Horde to win. Light brigade chess is a game - apart from the usual king and pawns, one side has three queens and the other has seven knights. Chess tutor is a mode where the player is guided on how to win the game in a fast manner.

2 System Requirements

2.1 Functional:

- Users shall be able to play against another human opponent (multiplayer).
- The system shall appoint rankings to the players of the game, which indicates that the winner of a game is awarded points which will have to be calculated.
- The system keeps track of the results for each game played.
  - This enables the system to provide an overview of the ranking of the human participants, based on on how many games they have won.
- Users shall be able to play against an AI, which will have 3 levels of difficulty:
  - Easy
  - Medium
  - Hard
- Winning against a more intelligent or higher ranked AI should award more points.

2.2 Non-functional:

- The system shall be made with 2D graphics.
  - It must feature a board layout.
  - It must feature player statistics.
- The system shall be implemented on top of a open source graphics engine.
  - The game shall be implemented in JavaFX.
- The system shall be programmed in such a way that the ruleset (board size etc.) can easily be changed.
- System shall be programmed in Java.

3 User stories

USER STORY 1: As a Chess player I want to play against good chess AIs so I can become better at chess.

USER STORY 2: As a pair of friends, we want to play chess together so we can spend more time with each other.

USER STORY 3: As a user of the chess game I want my available moves to be visualized when I pick a given chess-piece.

USER STORY 4: As a beginner level chess player I want to learn to play the game/basic rules of the game so I can play the game without help.

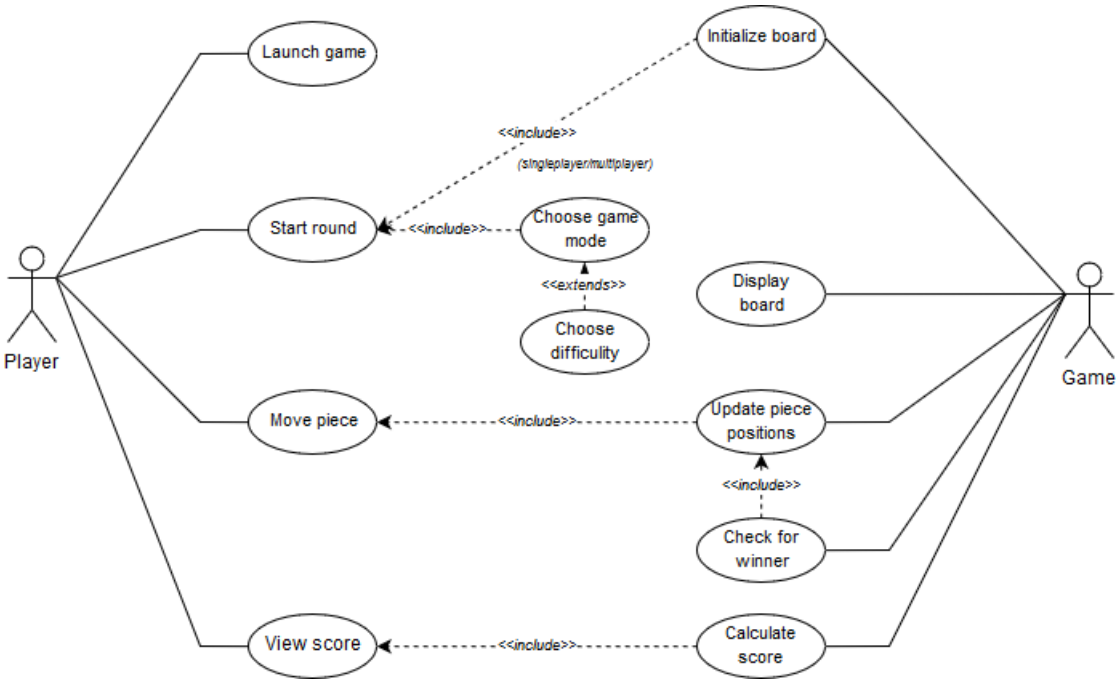
USER STORY 5: As a user I want the moves I make to be viewable through the course of a game.

USER STORY 6: As a frequent user of the software i want my MMR to be saved/updated after every match so i can show it to my friends.

USER STORY 7: As a frequent user of the software, i want to try other variants of the chess game

4 Use case diagram

Diagram showcasing the major user operations of the chess application.



5 Use cases

5.1 Use case 1 (fully dressed)

A start menu allows a user to start the game and choose single player to play against a machine AI. The user will be prompted to write their player name so the high score can be saved in an external file. In the next step the player has to choose between three difficulties. After the AI difficulty is chosen, the game will start. The player must progress by moving their pieces according to the game rules. The player can move pieces by clicking on their piece (if it is their turn and a legal move) and pick the square that the piece should be moved to. The machine will respond according to the difficulty chosen by the player earlier.

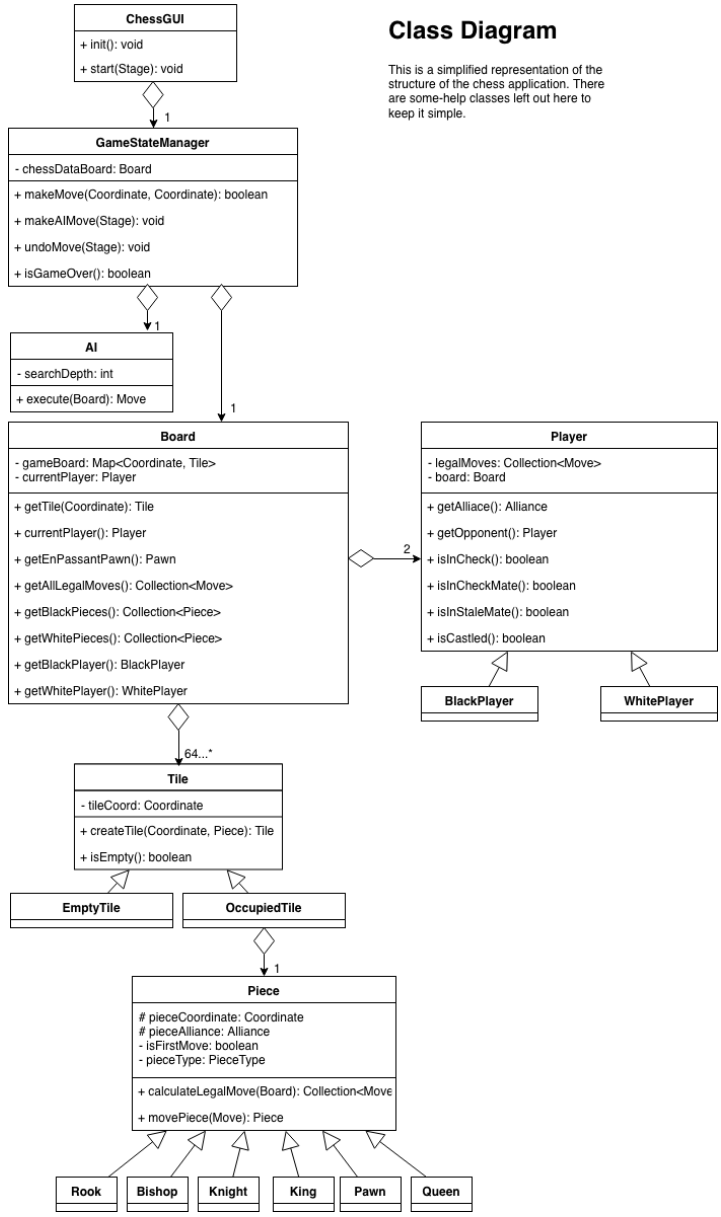
Name	Play chess with AI
Scope	Chess
Primary actor:	Player 1
Level	Summary
Stakeholders and interests	Player: wants to win and proceed in game machine: wants to win
Precondition	The player knows chess
Minimal guarantees	Player gets to play chess
Success guarantees	Either Player or machine wins
Trigger	Starts game in game menu
Main success scenario	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds the player will put the opponent in check 4) Checkmate, which results in a win for the player
Extensions	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds the machine will put the player in check 4) Checkmate, which results in a win for the machine

5.2 Use case 2 (fully dressed)

A start menu allows a user to start a game and choose multiplayer with another local player. After the player picks the multiplayer option it will be prompted to write name of player 1, then name of player 2, so the high score can be saved in an external file. After the name dialog, the players will be taken to the chess game and the possibility to play against each other. The game will be started and either player 1 or 2 starts. The players must progress with moving their pieces according to the game rules. The player can move pieces by clicking on their piece (if it is their turn and a legal move) and pick the square that the piece should be moved to.

Name	Play multiplayer chess
Scope	Chess
Primary actor:	Player 1
Level	Summary
Stakeholders and interests	Player 1: wants to win Player 2: wants to win
Precondition	Both players know chess
Minimal guarantees	Player 1 and player 2 gets to play chess
Success guarantees	Either player 1 or player 2 wins a game of chess
Trigger	Starts game in game menu
Main success scenario	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds player 1 will put the opponent in check 4) Checkmate, which results in a win for player 1
Extensions	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds player 2 will put the opponent in check 4) Checkmate, which results in a win for player 2

6 Domain model (class diagram)



6.1 Short explanation of the project structure

The ChessGUI class is in charge of showing the board to the player and the GameStateManager cooperates with the GUI to update the board according to

user input. The GameStateManager holds a Board object that we display to the user with the help of the javafx library in ChessGUI. We get user input that we can use to change the board, or alternately we can get input from the AI. Whenever we change the board we let the GUI redraw it for the user to see. The main game logic is taken care of in the Board class, with the GameStateManager supplementing this class to keep track of state information. The Board class keeps track of the two players as well as all the tiles that can contain a piece. The Tile class is a way of keeping track of all the pieces in a way that makes is easy to draw in the GUI. The pieces contain all the logic of how they can move,so we can simply call that piece to get its possible moves. The Player contain information necessary to analyze the board for certain events. It is also capable of making a move, which will return a new board object with the move executed. Remember all of these are reachable from Board and thus GameStateManager and ChessGUI. There are several helper classes, factories and enums left out of this graph to make it readable, but this is the overall structure off the entire project.

6.2 Additional class overview

Just a full class overview for good measure: