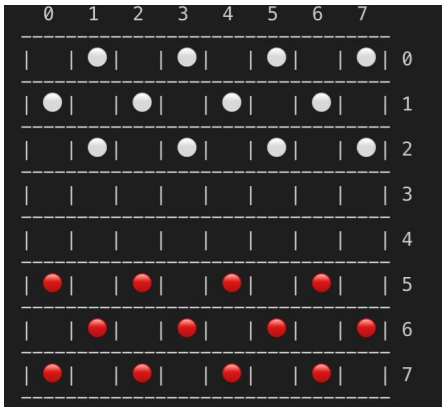
A6 Sprint 1 Progress Report: Checkers

Vision

We are building checkers platform that includes multiplayer and singleplayer modes. We are planning to implement an (AI) bot for the single player mode. The bot will always take a valid move and will always take (jump over) the opponent's piece when possible. If we have more time, we will implement an AI bot using game tree search (and alpha-beta pruning) aiming for a win rate higher than 50%.

Summary of progress

We have implemented a two-player checkers game system. The components of this system includes the main game logic, the board to keep track of game state, a pretty printer for the board, and a parser to handle user input. We also have a test suite for the main game logic functions. In the demo, we showed how the game works from the starting board and what is ouputted when one of the players win with board with two pieces.



Pretty printing of the Board

Activity breakdown

- Joo Hyun
    - Board.ml and Board.mli: worked on the functionality to use for the game like moving a piece, function to check if a move is valid, etc.

- Athena:
    - Board.ml and Test.ml: worked on testing and game logic functions such as checking for the win condition and checking whether a move is a valid jump.
- Henry:
    - Board.ml and main.ml: Worked on conceptualizing/"whiteboarding" and planning out the game module. Worked on general move functionality in board.ml and tweaking main.ml functions to print out proper messages.
- Timothy:
    - Main.ml and Parser.ml: worked on the functionality of running the game continually, such as game_input and game_loop. Implement the parser to translate the users' inputs.

Productivity analysis

We were productive in that we implemented all the features that we had planned to implement in the first sprint. Our estimates were accurate although we could have probably implemented more features such as mechanics for the mandatory jump.

Coding standards grades

- Documentation: Meets Expectations. We documented all the functions, helper functions inside board.ml, board.mli, which includes the major functions for game logic.
- Testing: Exceeds Expectations. We wrote tests for the main game functions such checking for the win condition and checking for valid moves.
- Comprehensibility: Meets Expectations. We used empty lines to separate definitions and used value, type names that are meaningful. We used type synonyms such as piece (color*rank), position (int*int) for comprehensibility.
- Formatting: Meets Expectations.We tried not to get over the 80 column limit. We used minimum parentheses and indented the nested if statements, pattern matching clearly.

Scope grade

Here is what we consider a satisfactory, good, and excellent solution:

- Satisfactory: the board.ml and all the rules are implemented
- Good: check the valid moves for the checker games
- Excellent: the main.ml is also implemented, the game works perfectly, and it passes all the test cases

We give our team “Excellent” for this sprint, because all of our source code including main.ml were implemented and passed all the test cases. Also, when we ran the game, it worked well with all of the cases of the user inputs such as valid move, invalid move, quit, etc. (except for the AI part which we will implement in the second sprint.)

Goals for next sprint

1. To Implement the mandatory moves
2. To implement the AI by calculating the probability
3. To add the King piece

Reference

http://www.cs.cornell.edu/courses/cs3110/2017fa/handouts/style.html