OCaml Checkers with AI

Mindy Lee Ryan Hill Stephanie Hellman Will Wang

December 23, 2020

1 Vision

The vision for this project was to implement a terminal based version of checkers to play anywhere and everywhere (with a laptop). The game has two game modes: a 1v1 showdown with two human players battling in a race to see who can capture all of the other player's pieces the fastest, or a riveting battle of the brains between the player and an AI with three difficulties (easy, medium, and hard). The game is incredibly user friendly for beginner checkers players, with an option for the player to take a hint and have the game tell them what is the best move from the current layout of their and their opponent's positions on the board. We implemented all the functionality we planned to, so our vision is essentially identical to the last sprint.

2 Summary of Functionality

We implemented a CPU for the single player game mode and hint functionality. There are three different CPUs: an easy AI that always moves the first available piece with its first available move (unless there is an available capture), a medium AI that chooses random moves available to it, and a hard AI that uses a Minimax game tree to find the best possible move, evaluated with a fine-tuned point system. The best move is calculated by looking at all possible moves available to the AI (depth in tree = 1) and then looking at all the possible moves the opponent can make in response to the AI's move (depth in tree = 2). See Figure 1.

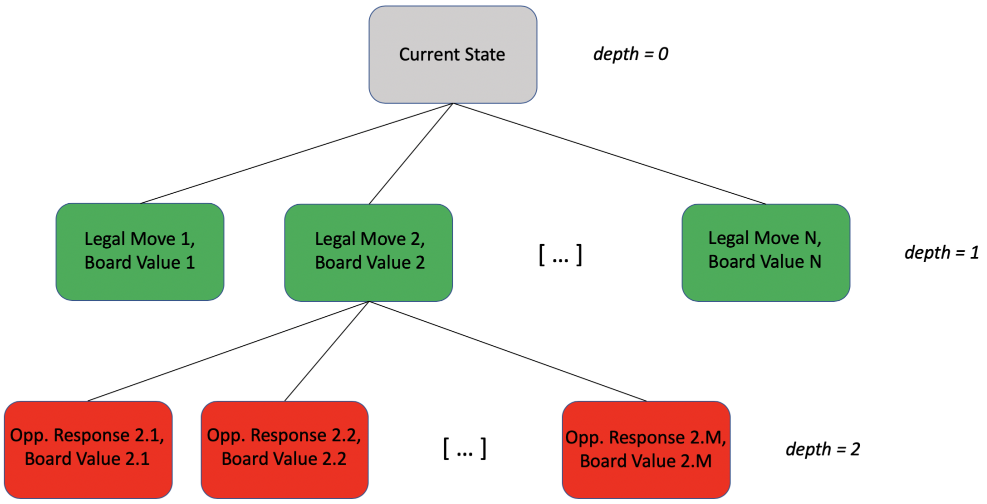
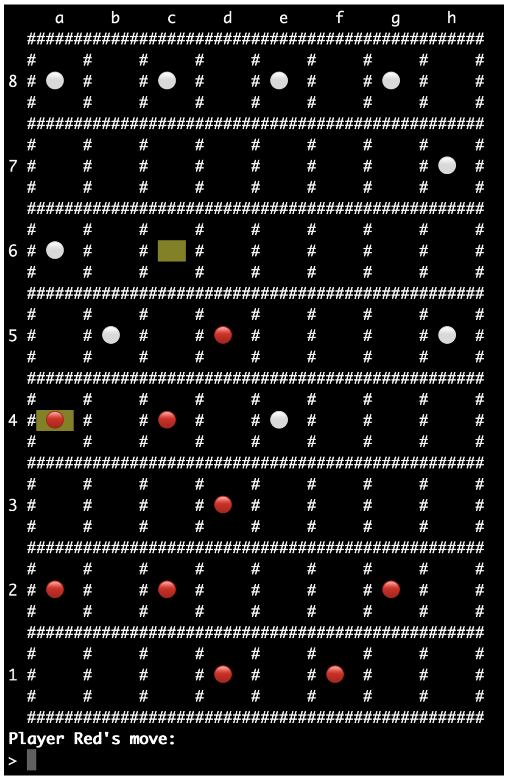

Current State
depth = 0
Legal Move 1, Board Value 1
Legal Move 2, Board Value 2
[...]
Legal Move N, Board Value N
depth = 1
Opp. Response 2.1, Board Value 2.1
Opp. Response 2.2, Board Value 2.2
[...]
Opp. Response 2.M, Board Value 2.M
depth = 2

Figure 1: Game Tree Visualized

The resulting positions of the AI pieces after each potential opponent response to a move the AI can make are then evaluated with a point system. The point system weighs king pieces greater than pawns, and assigns more weight to kings and pawns with stronger positions that are also dependent on the stage of the game. It also has different criteria for the position strength of kings and pawns. The criteria for a stronger position are as follows: moving pawns to the opposite end of the board, to the edges of the board, keeping pawns in the back row (to defend against the opponent kinging pawns), kinging pawns, making captures while avoiding getting captured, moving kings towards enemy pieces, and moving kings in tandem with other friendly kings, and weighing edges and back rows less in the late stages. Each of these move scenarios were weighted differently according to the order of optimal game-play strategy (e.g. capturing a piece is preferred to moving a pawn to an edge).

All levels of the AI follows all rules of the game, including executing a capture if it is available, and multi-capturing if they are able to do so. None of the AI raise errors by making an invalid move. The Hint command uses the best possible move algorithm used by the hard AI to highlight the best moves the player can make given their current board positioning.



```
Player Red's move:
>
```

Figure 2: Game Board with Hint Invoked

3 Activity Breakdown

- *Mindy*: Implemented easy and medium AI, integrated hint in the UI.
- *Ryan*: Implemented hard AI.
- *Stephanie*: Created a working UI for choosing the difficulty of AI, main.ml functionality for AI, implemented hint.
- *Will*: Implemented hard AI.