Farhad Ahmed

Artificial Intelligence

Edward Wong

05/17/18

**AI Checkers Final Project**

Instructions:

The project contains 4 files:

AIPlayer.h

Player.h

Checkers.h

Main.cpp

Main (exe file)

To compile the program, I used the make command in my MacOS terminal after navigating to the correct location of the 4 files. Using the make command produces the Main executable file in the same directory as the files. Double clicking on the exe will run the program in a new terminal window.

High Level Description:

The checkerboard is stored as a vector of a vector of of strings. Any space with an ‘[A]’ denotes that an AI piece occupies that space, ‘[P]’ denotes the player, ‘[L]’ denotes an unoccupied Light Square, and ‘[D]’ denotes an unoccupied dark square. After intitializing the player, AI player, and checkerboard, the player chooses the difficulty and whether to go first or second. Both players are implemented to make their moves by keeping the checkerboard as a private data member and running its functions on the Board to determine where its pieces are and what jumps/moves are possible for each piece. Before moves are even considered, if there is any jump to be taken, the players are forced to considered jump options. After the player has decided its move, the player will alter the gameBoard to reflect those changes and return the board so that the the other player can take in the updatedGame and decide its moves on the new Board. The AI Player recursively calls its minmax function to get the scores of the direct children of the current state and chooses and returns the state with the max score. This exchange of moves and the board occurs consistently until one of the players does not have a valid move to make. In that case, the player that still is able to move will go again. If neither player can move, the game is over and the win is given to the side with more pieces. If the pieces are the same, then a draw is issued.

Terminal States:

States are classified as terminal states when there are no moves or jumps left for either player to make. The utility value of these terminal states are determined by whether the game is a win or loss for the AI player. The AI player wins when there are no Player Pieces left and only AI pieces left.

Evaluation Functions:

For the easiest difficulty, the evaluation function used simply subtracts the number of Player Pieces from the number of Ai Pieces trying to maximize the number of Ai Pieces at all times.

For the medium difficulty, a higher score is given to pieces in the second half of the board. This forces the AI to be aggressive and actively seek out interactions with Player Pieces. For the hardest difficulty, a higher score is given to pieces on the edges since those pieces cannot be jumped.