Checkers AI Documentation

Rules of the game

This checkers program implements all the rules of checkers: pieces can move diagonally one square, or jump an over an enemy piece (destroying it in the process) and follow this up with subsequent jumps until no more offensive jumps can be made. If a jump move can be made, it must be taken. If a piece reaches the opposite end of the board from where it started, it is turned into a 'king' piece and can move and jump backwards as well as forwards, as otherwise, they can only move in one direction. Pieces can turn into kings mid-jump sequence. A player loses the game if they have no more pieces.

Controls

Left and right arrow keys – cycle through available moves

Space – apply the move.

WASD, control, shift – move the camera.

AI

The AI uses a modified Monte Carlo search tree algorithm to decide each move.

The Monte Carlo method is an algorithm used by AIs to make decisions with a tree of 'decision' nodes. For every initial legal move, a node is added, and through that node the rest of the game is simulated many times in order to apply weighting to that node. When the randomised game ends in a success, it is a positive result, whereas failure is negative, and each is typically added to the weighting score by adding or taking 1. This can be expanded out to a larger tree by creating nodes for each following legal move. Once all these moves are simulated to the end of the game, and each node has a score, the node with the best score is chosen, as most end results after that move ended in success.

Firstly, the AI takes the board state and a list of all possible moves it can make that turn. For each possible action, it will create a cloned board state, apply the move, then will play out a randomised game (including randomising any subsequent jump actions) to a specified number.

The randomised game will return a value depending on whether or not the AI player 'won' after taking that move. On top of the standard randomised games the Monte Carlo algorithm runs, several additions have been made. After 5 and 20 moves, if the AI's side has more pieces than the other, then the value is incremented. If the opposite is true, it is

decremented. The same check is applied at 200 moves, which is the cap on moves so that the game does not play indefinitely. The score is also reduced if the move immediately puts their piece into ‘check’.

Once all the games have been played and each action has an attached ‘score’, the one with the highest is applied. The entire process is then repeated if the move was a jump move and there are possible sequential jump moves to be made.