Reactive Checkers

Members: Kaleb Dutson
Michael Hoffmann

Overview

The goal for this project is to create an agent that can play checkers competently against a human player. The agent will be reactive in that it does not plan moves for future states, nor does it remember previous states. The agent will move its pieces based on a set of prioritized rules. The project will be created using pygame.

Desired Outcome

We want to demonstrate that a reactive agent can be designed with rules sufficient to allow it to sometimes beat a novice human checkers player. The agent will be purely reactive, using only the current state of the board to make decisions. It will not rely on a history of moves or a prediction of possible moves beyond one turn ahead. This means that the agent will not use planning or learn from past mistakes. These limitations will demonstrate the feasibility of creating simple input-utility-output systems that can solve complex challenges.

Sensors and Actuators

Our initial approach will be for the reactive agent to contain a tree filled with sub-agents, one for each checkers piece. The agent's only sensor will be an array of 64 game squares, each containing information about the piece or absence of a piece on the board. At its turn the agent will poll each of its pieces and they will report their legal moves. The agent will then compare those and prioritize moves based on the state of the game. The agent's only actuator will be to move a piece, which will be handled as a function of the environment to check that the agent is not attempting an illegal move.

Anticipated Challenges

- Creating the reactive rules the agent will follow will be one of the biggest challenges since the rules will completely determine how well the agent plays checkers.
- We will need to create a test board environment to be used when testing. This will allow us to test the agent's reactions without needing to play an actual checkers game to get to a certain state.

User Interface

The player will have an 8x8 checkerboard where they can play against the reactive agent. The human player will always go first. The human will be able to click a piece, then click a square that the piece should move to. If that move ends the player's turn then the agent will move, otherwise they'll be prompted to move again (as in the case of a jump). After the player's turn ends the agent will take its turn and control will return to the human. This will repeat until one player has run out of pieces, ending the game.

Roles

Kaleb:

- Create checkerboard
- Create basic setup for pieces
- Define reactive rules
- Create the testing environment

Michael:

- Make functions for the game pieces to poll an environment for its legal moves
- Make a function in the environment that accepts a move from the agent or the player, checks whether it is legal, and performs it to update the game state and the user interface.
- Create classes for the agent, the game pieces, and the player's controller.