CS440: AI Project 2 Wumpus World
Lab Report

Christopher Yong (cy287), Fares Elkhouli (fae15),
Shlomo Benyaminov (ssb165),
Dennis Galinsky (Dbg65)

October 28, 2020

General Information

How To Play?

Setup

1. Choose game dimension 3,6,9 and hit start.
2. Choose AI search depth and hit select.
3. Choose AI heuristic from heuristic list.

After Setup

1. Click a unit you would like to move
2. Click a valid position on the grid to move into.
3. Hit Next Turn.
4. Repeat steps 1-3 until the game status is no longer "In Progress".

NOTE: Some of the heuristics work poorly alone, meaning the AI might suicide its pieces or perform bad choices. For the best heuristic, please select the fifth one **"Weighted Heuristic 1-5"**.

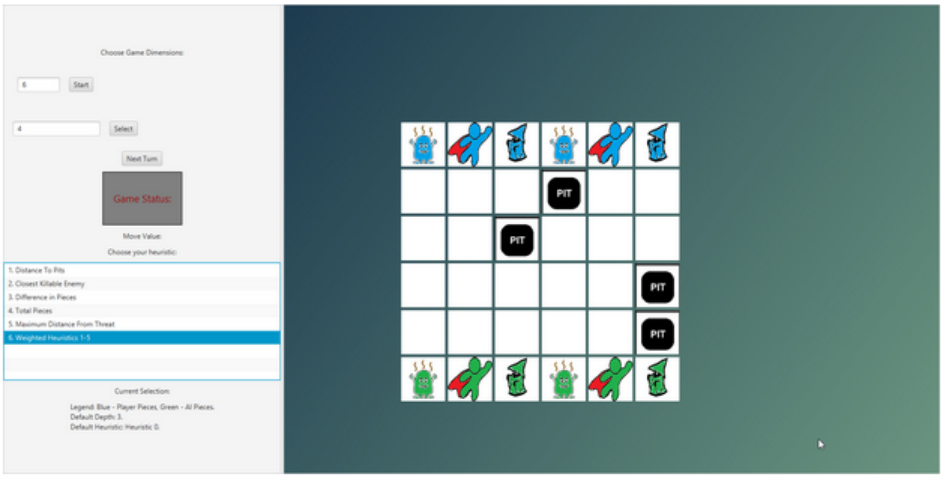
Dependencies

- JavaFX

Question 1

Implement the interface for the game. The interface should display the board with all of the pieces, it should allow the adversary (human player) to input moves and update the state of the board based on the players move (including any captures). If should then indicate the agent's move and update the state of the board based on the agent's move.

Answer:
The game was implemented using Java and JavaFX



Question 2

In this problem it is clearly infeasible to search out the entire depth of the search tree and thus we will be bounding our search by the depth of d. Propose a metric for evaluating the bottom-level nodes of the tree in such a way that a higher metric value indicates a better state for the agent and a lower metric value indicates a better state for the adversary.

Answer:
A good metric to evaluate a state of the board is a combination of things. We want to maximize the AI pieces distance to pits, minimize the average distance of AI pieces to player pieces that the AI pieces can kill, calculate the difference between AI and player pieces, and calculate the total pieces the AI has versus the player. A simple way to use a combination like this is to keep the weighted-value a positive number for the AI and simply negate the value for the player since the AI will choose maximized numbers, and the player will choose minimized values.

Question 3

Implement minimax search. This algorithm should take as input a max depth and conduct the minimax search out to that depth. When evaluating states bottom-level states during minimax you should use the metric you proposed for question.

Answer:
Implemented in source code

Question 4

Implement alpha-beta pruning for your mini-max search.

Answer:
Implemented in source code

Question 5

Specify a set of 5 potential heuristics that can be applied to this game and discuss the benefits of each of the heuristics. Note that the heuristics do not need to be admissible.

Answer:

1) **Maximize the distance between AI pieces and pits**. We want to maximize the distance to pits so that AI pieces aren't eventually trapped and have more possible moves to attack or play defense.

2) **Minimize the distance to a killable enemy**. For instance, if the AI detects a close mage, it will want to minimize the distance between the enemy mage and its wumpus because a wumpus kills a mage. The same is true for hero/wumpus and mage/hero pairs.

3) **Maximize the difference in pieces**. The AI wants to maximize the difference in pieces it has over the user. This will ensure that the AI thinks twice about sacrificing itself if it is not beneficial to the overall game, it always wants to have more pieces than the user, not just equal pieces.

4) **Maximize total pieces**. This heuristic deals with maximizing the total pieces the AI has, rather than the difference in pieces it has (like the previous heuristic). The reason this heuristic is important is because the AI would want to prioritize keeping atleast 1 of each piece (wumpus, hero, mage) because once it has 0 pieces of a piece, it is at a disadvantage since it cannot kill an opposing enemy piece without sacrificing its own now.

5) **Maximize distance from threats**. The AI wants to create as much space between each piece and any threat on the board that can kill that piece. For example, the AI will maximize the distance between its hero and the player's hero and/or mage. It will do this by either running away or eliminating the nearest threat.

6) **Linear combination of the heuristics 1-5**. Heuristic $= 0.05 * H_1 + 0.15 * H_2 + 0.5 * H_3 + 0.2 * H_4 + 0.10 * H_5$. The individual heuristics by themselves only specialize in maximizing or minimizing one area of the game. By themselves, the heuristics are useless since they do not account for any other variations. This is why to build an intelligent AI that can compete and beat a player, we use a combination of all previous heuristics. This version of the AI can now detect many things at once.

Question 6

Implement the minimax algorithm with heuristics described in section 2 and apply it to this problem.

Answer:
The player can select which heuristic it wants the AI to perform on the UI