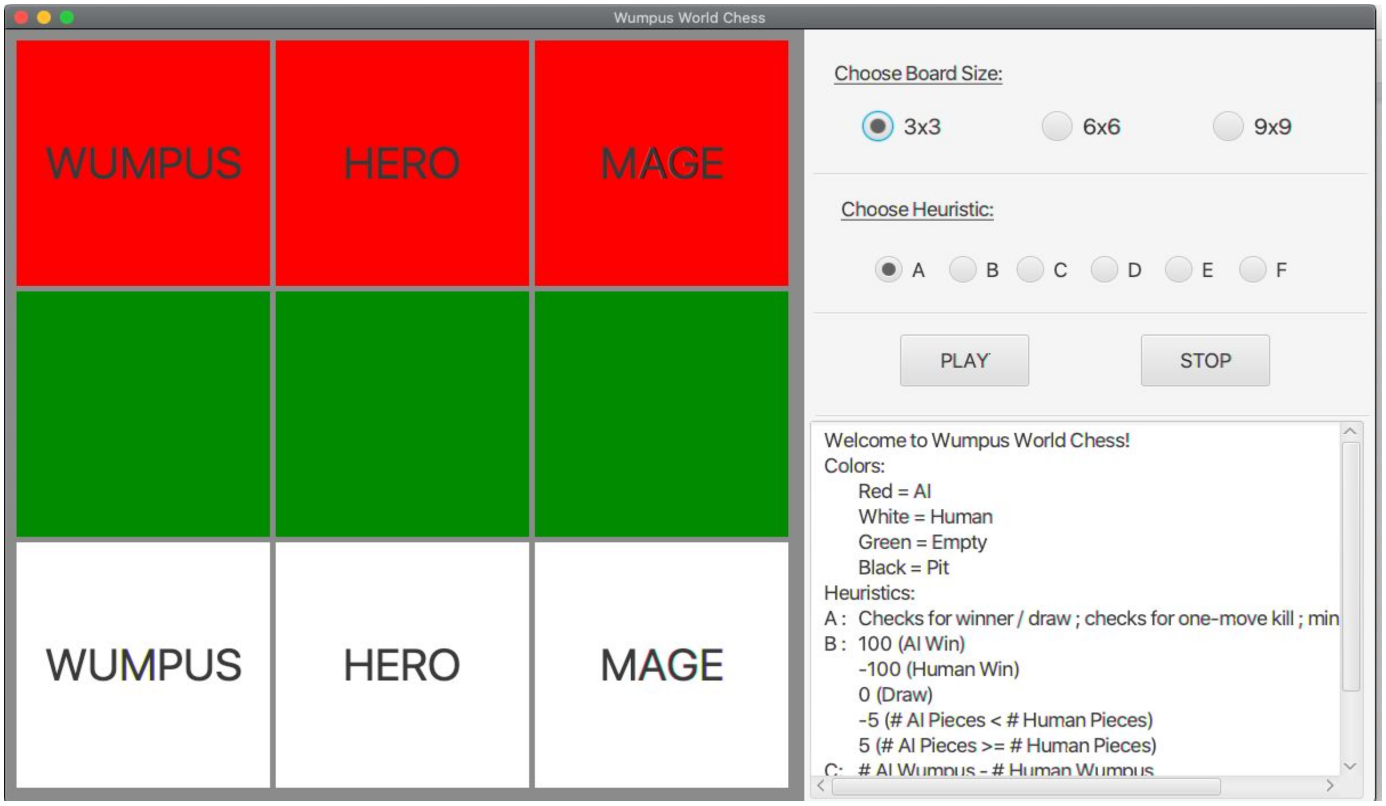
Chris Zachariah | cvz2
Jason Chou | jc2280
Myles Chou | mc1998
Tyler Hong | th517
CS440: Introduction to Artificial Intelligence
Project 2: Adversarial Search

Report

1. Graphical User Interface Overview:

Our Graphical User Interface, that we designed and built, displays a grid and all the functions that the user needs in order to play the game against the AI player. The GUI allows users to choose a board size (between 3, 6 and 9) and displays the game board in a clear to use and easy way.

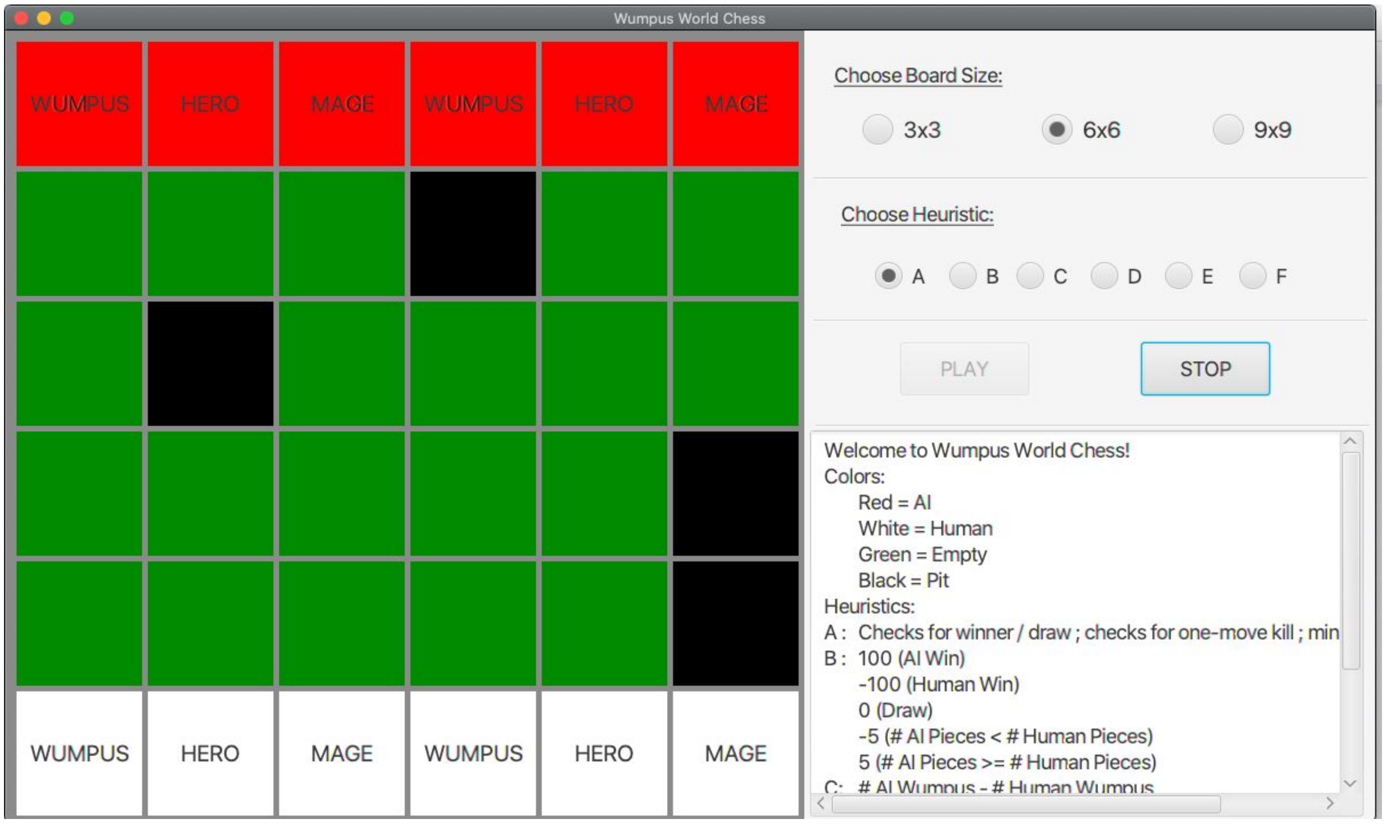
The user can choose between the six different heuristics (metrics) displayed by the letters A through F. There is a box underneath the options which briefly describes the size of the board, heuristic chosen and the moves of both the human player and the AI player as the game progresses.



Choose Board Size: [✓ 3x3] 6x6 9x9

Choose Heuristic: [✓ A] B C D E F

Welcome to Wumpus World Chess!
Colors:
    Red = AI
    White = Human
    Green = Empty
    Black = Pit
Heuristics:
A : Checks for winner / draw ; checks for one-move kill ; min
B : 100 (AI Win)
    -100 (Human Win)
    0 (Draw)
    -5 (# AI Pieces < # Human Pieces)
    5 (# AI Pieces >= # Human Pieces)
C: # AI Wumpus - # Human Wumpus



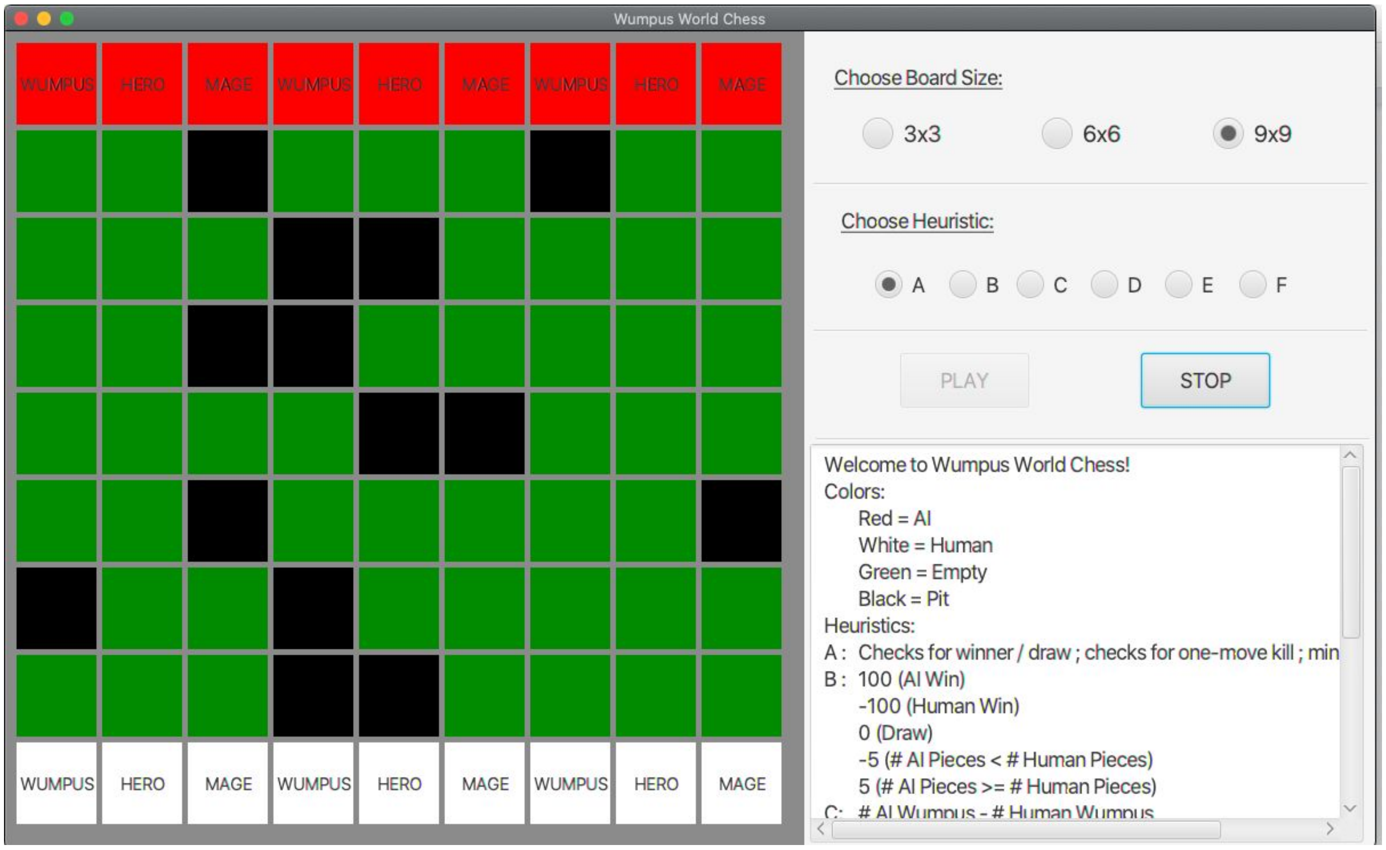
Choose Board Size:

3x3 [✓ 6x6] 9x9

Choose Heuristic:

[✓ A] B C D E F

```
Welcome to Wumpus World Chess!
Colors:
    Red = AI
    White = Human
    Green = Empty
    Black = Pit
Heuristics:
A : Checks for winner / draw ; checks for one-move kill ; min
B : 100 (AI Win)
    -100 (Human Win)
    0 (Draw)
    -5 (# AI Pieces < # Human Pieces)
    5 (# AI Pieces >= # Human Pieces)
C:  # AI Wumpus - # Human Wumpus
```



Choose Board Size:

3x3 6x6 [✓ 9x9]

Choose Heuristic:

[✓ A] B C D E F

```
Welcome to Wumpus World Chess!
Colors:
    Red = AI
    White = Human
    Green = Empty
    Black = Pit
Heuristics:
A : Checks for winner / draw ; checks for one-move kill ; min
B : 100 (AI Win)
    -100 (Human Win)
    0 (Draw)
    -5 (# AI Pieces < # Human Pieces)
    5 (# AI Pieces >= # Human Pieces)
C:  # AI Wumpus - # Human Wumpus
```

(Examples of the different graphical user interfaces for each size game board)

2. **Minimax Search w/ Alpha-Beta Pruning Description:**

   The Minimax Algorithm works through recursive calls on the given board state. At each different recursive stack, depending on whether it is the human's turn to play or AI's, it will either pick the minimum value or maximum value, respectively. This way, the AI will always be at its maximum board state even if the human plays the best move.

   Because of its recursion, the algorithm is restricted to how many levels to search. To increase the efficiency of the algorithm, we implemented a pruning feature that kills search depths where the final value of the board state is out of range of the most effective board state value possible. To keep track of this, we add parameters alpha and beta to the minimax search function. Alpha and beta will keep track of the best or worst values of the board depending on whose turn it is. Therefore, if a tree is searched and it is out of range of a previously calculated value, the search will terminate on that tree to improve runtime.

3. **Optimizations:**

   We used similar optimization techniques to the ones we utilized in Project One. For any sort of searching and storing of board, node and moves we used arrays and max heaps in order to keep the cost O(1) or constant time.

   We used max heaps to store the heuristic values for every board so that the algorithm can pick the best board (best next move) to use in every iteration. Given the time constraint on Project Two, we used what we learned in Project One to allow us to optimize our code better and more efficiently.

4. **Heuristics (Metrics) Used:**

   Heuristic A: First checks to see if either the user or AI has won the game or if a draw has occurred. Then it checks all the potential moves to find either immediate one-move kill or draw. Lastly, it goes through all the AI pieces and checks the distance between the pieces it can kill or draw against and increases the points for pieces which are closer.

   Heuristic B: Checks to see if user or AI has won or if a draw has occurred. Heuristic then increases the points if there are more AI pieces than user pieces remaining on the board.

Heuristic C: This heuristic targets a specific type of game piece. It returns the difference between the number of AI Wumpuses to user Wumpuses.

Heuristic D: Prioritizes the Hero game pieces for both user and AI. Returns the difference between the number of AI Heroes and user Heroes.

Heuristic E: Lastly, this heuristic finds the difference between the user number of Mages and the AI number of Mages on the board.

Heuristic F: Heuristic F will run the previous heuristics to find the highest value possible value to return and then returns it.

Among these different heuristics/metrics used, we believe that Heuristic A is the most admissible heuristic. This is because it is the only heuristic that incorporates the use of distance on the board in order to come up with a reasonable score that the AI could use in order to differentiate between good and bad moves. The other heuristics focus more on the overall number of pieces or specific type of pieces, which based on our testing does not help the AI play the game in the most practical manner.

With Heuristic A, the AI seems to be able to understand the idea that getting the piece closer to another piece that it has a chance of killing would increase the likelihood of it winnering drastically.

Given the time constraint for Project 2, we believe that Heuristic A is a good admissible heuristic that definitely works well for this game.

5. **Behavior of AI:**

Our AI is rather simple in nature as time constraints made it difficult to further polish the algorithm. However, depending on the heuristic chosen by the user, the AI performs to varying degrees of success.

Our complex Heuristic A allows the AI to make competitive moves against the user while some other inadmissible heuristics such as Heuristic C, D, and E will not be as consistent in the aid it will provide the game playing AI.

6. **Better Heuristic Proposal:**

If we were given some more time for the project, a particular heuristic that could be very beneficial for the AI to use would be one that incorporates proper distance and proper (and proportionate) weights that would help determine the best next move to make.

This better heuristic would first go through all the possible moves on the board that it could make and see if any of those moves will help get rid of the opponent's pieces. This is important since the main objective of the game is to kill off all the opponent's pieces without losing your own. Each opportunity to immediately kill off, draw against or be killed by one of the opponent's pieces will be taken into consideration. These would be a point system that will calculate which decision has the best outcome. The points will get added or subtracted depending on which outcome ends up having the higher chance of occurring.

Once that is taken into consideration, then the next point to look at is the distance between potential kills, draws and defeats. Each of the AI's pieces will store the closest distance to a piece that it can kill, a piece that it can draw against and a piece that can kill it. Ideally, the AI's piece is closer to the opponent's piece that it can kill rather than going for a draw or defeat. A point system will be assigned to this as well.

Once all the points have been assigned, the move with the highest number of points will be chosen as the best move since it will be the best chance for the AI to obtain victory in the game.

If there was more time given, this would be a possible heuristic that we would definitely have looked into. Heuristic A is the closest to this heuristic in the ones that have been implemented so far.