Reversi Revisited **12/10/18**

Team Members
Brian Petersen
Alex Wilson

Algorithm Details
The algorithm is written in Java. When the client is asked to play a move, it enumerates all legal moves that can be played. From this set a move is chosen. The move is chosen by running each of the derived states in a minimax algorithm that uses alpha beta pruning. A depth of 7 is used by default. Shallower depths are used depending on the time remaining for the AI player. Once the target depth is reached in the minimax algorithm, a heuristic algorithm is called to evaluate the board state. The state of the board is represented as a *double* — the larger the value, the better the chances are the AI player has of winning, according to the heuristic algorithm.

The heuristic algorithm is quite simple. It uses a total of three well known Reversi strategies:

- *Position values (**V**):* the value of a owning a specific position of the board. Values were chosen from research papers. For example, corners as worth more than center positions. A player's ***V*** is determined by summing the values of all the positions they own.
- *Player mobility (**M**):* the number of legal moves that can be made by a specific player. Higher mobility ensures that more spaces can be explored. The higher the better.
- *Player piece count (**C**):* the number of pieces a player owns minus the number of pieces of the opponent. The higher the better.

Different phases of the game use a different linear combination of these heuristic factors. In the early phase of the game (i.e. less than 40 positions are claimed in total), the evaluation function is ***10V + 20M + 0C***. In the end phase, it is ***10V + 0M + 20C***. In summary, the algorithm focuses primarily on mobility early on in the game. Later on it focuses on gaining as many pieces as possible to win. In all phases of the game, it focuses on claiming pieces that are valued highly because of ***V***.

Qualification for Tournament
The algorithm described above was put to the test using the provided AI implementation. Our algorithm beat the provided algorithm 6 of 6 times on *Easy*. Ours beat the provided algorithm 12 of 15 times on *Easy, Medium 1,* and *Medium 2*. As such, our algorithm qualifies for the tournament.

CS 356

- user centered iterterative design: observe → prototype → think/observe

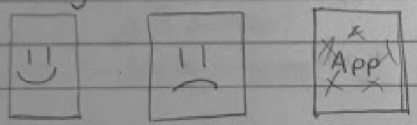
Fluidity - easy to change
Fidelity - true/resembles final solution
} through process moves from h-fl+l-fi to l-fl + h-fi

Fidelity	Fluidity: low	Fluidity: high
low	fail	paper
high	functional fidelity	holy grail

Problem statement - specific, detailed, no solution

- storyboard

setting | conflict | resolution



- solution statements

analogy - solutions inspired by, patterned after
utopia - unrealistic (optimal) perfect solution
laser focus - focus on one aspect exclusively

- critiquing - describe objective | analyze criteria | judge success or failure?

- wireframing - used to engage people in conversation
focuses on layout/flow (important content) (not settings, account, etc)

- reading
  - develop empathy for users
  - responsive design is the future
  - flat: buries affordance
  - learnable, effective, effecient

affordance: visual characteristics that make clear how to use something
- delight: idea that's hard to do w/ clever technical solution