Project 1: Game AI

Due: 1 March 2016

In this project, you'll choose one of the games we've been working on in class, and implement it in a program that lets a human play against the computer.

The exact details of your program's interface are part of your design task. Your program will need to be able to distinguish and display all valid board states, and accept user input permitting all valid actions.

Checkpoint

For the checkpoint (next Tuesday), you should have a program that:

- has a plan for representing all valid states and actions,
- displays the initial board state,
- reads actions from the keyboard,
- makes the correct state update for at least one valid action, and
- displays the resulting board.

The representation plan can be given in a readme if its final code implementation is not yet 100% complete at this point.

The checkpoint version is due at **4pm on Tuesday, 16 March**.

Final version

A full-credit final version will be a complete, non-buggy, working implementation of one of the games specified above, TOGETHER WITH convincing proof that it is correct. The "proof" should consist of test cases (in whatever format is convenient to you) to illustrate various situations, including both input and expected results.

Remember that there need to be clear instructions on how to run it in general as well as how to run each/all of the tests and quickly verify that they ran correctly (and which rubric items each one corresponds to); and don't forget to explain how to enter actions and interpret the display! Having complete and correct documentation is an easy 25 points, but if your documentation omits important info or tells me the wrong thing, you'll get less than full credit there.

After checkpoint work (25 points) and documentation (25 points), there remain 100 points in the rubric, which will be awarded according to the table below. Under each score, I show (for your convenience) the total cumulative points if you get that item plus *all* the previous points, and the letter grade this corresponds to. The order is less significant than usual here; different rubric points are easier or harder depending on which game you chose. You can in general get points for the later ones (if they work) without getting the earlier ones.

Note that Ricochet Robot solutions max out at a C+, because it's not a turn-taking game and so several of the points won't be available. It should be, at most, your emergency backup plan.

Chakra has available a web API that takes the visualisation of the board and the human-player side of things entirely off your hands (and gives you the direct ability to play your AIs against each other, which is always fun). I've provided a C++ starter kit so you don't have to worry about the web API stuff (much), and a Javascript version is available and I think I can get Python to work quickly. This means that some of the rubric points are almost free (but not quite completely free); I've noted this in the rubric.

The other four games all have various pros and cons, but all of the rubric points will be available for them.

NOTE: if your code doesn't compile, or immediately crashes when it's run, you will get zero of these points. Don't let this happen to you!

Score	Description
10 (60/D−)	Displays initial board, reads a human-player move, applies and displays result ○
10 (70/D)	Plays game for several moves, alternating between human-player moves and valid AI-player moves †
10 (80/D+)	Detects end-game state (and scores it, if appropriate) and reports the result *
10 (90/C)	Accepts and responds correctly to all valid human-player moves (even ones not available from the initial state) * ○

Score	Description
10 (100/C+)	Can clearly enumerate (perhaps in a debugging statement to `cerr`, possibly triggered by a command-line option or in-game command) a complete list of available AI actions from any valid state *
10 (110/B−)	Can play game from start to completion without crashing or hanging, with the AI player always choosing *some* valid move (not necessarily a good one) ††
10 (120/B+)	AI makes good choices at least in the cases where a win or loss is just a few moves away * †
10 (130/A−)	Implements a reasonable heuristic evaluating board states, with very positive scores corresponding to better boards for one player and very negative scores for the other * †
5 (135/A)	AI uses heuristic effectively, making good choices even when a win is not imminent * †
5 (140/A)	The computer is capable of playing as either player (black/white, red/blue, whatever) based on command-line options. ◦◦ †
10 (150/A+)	Some other good AI work. This is open-ended but possibilities include particularly good heuristic functions, alpha-beta pruning, or clever data structures that permit better caching or increased lookahead. †

* It might be helpful to have some mechanism to load a game already-in-progress to test and/or demonstrate some of these.

◦ For Chakra, the "applies a human move" points are *almost* free (since the UI does the heavy lifting there), but you do need to apply it to some sort of internal state, not just print out the Action as is done in my starter kit.

◦◦ For Chakra, the "either player" point is *almost* free, as long as you don't break it when you add an internal representation.

† These points unavailable for Ricochet Robot.

†† For Ricochet Robot, the "play to completion" point should let the human enter a full path and then report the length and contents of the optimal path.

The final version is due at **4pm on Tuesday, 1 March**.

Handing in

Hand it in as `proj1` using the handin script.