Project 1: Game AI

Due: 22 February 2022

v20220209-0245

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 modeling *all possible* valid states and actions,
- displays a reasonable view of the initial board state,
- reads actions from the keyboard,
- makes the correct state update, once, for at least one valid action, and
- displays the resulting board.

The model plan can be given in a readme if its final code implementation is not yet 100% complete at this point. It can also be given as code, and I will read the code, but your readme has to tell me where to look for it.

Note: Every game on the list either has a board more complicated than "just a 2D array of items" or a fairly complicated action/piece situation, so if you have a plan like "model is 2D array of int, view is just print it out in a rectangle" (and especially if you chose a particular game on that basis), you should consider further.

The checkpoint version is due at **4pm on Tuesday, 8 February**.

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 15 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 (15 points) and documentation (15 points), there remain 70 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 ICO solutions max out at 90/100 points, because its state space is so low-dimensional that you can in general search to goal states and not bother with a heuristic. (Also note that state representation is a bit painful for ICO.)

The game Reflecto is NOT available for this project, for reasons we'll discuss in class (and in homework).

The other 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
State model and view, and human player controller	
10 (40/D−)	Displays initial board, reads a human-player action, applies and displays result
5 (45/D)	Can load and display state of a game already-in-progress from a file (crucial for testing and demoing many of the later rubric items); format of this file can represent *any* valid game state and either the human or AI as "next player"
5 (50/D+)	Detects end-game state (and scores it, if appropriate) and reports the result
5 (55/C−)	Accepts all human-player actions in a usable format, and responds correctly to all valid ones (even ones not available from the initial state)
5 (60/C)	Rejects all invalid human-player actions, without rejecting any valid ones
AI player controller	
5 (65/C+)	Can identify at least one valid available AI action, and take it, in any state where the AI has a valid action
5 (70/B−)	Can play game from start to completion without crashing or hanging, alternating between human-player actions and AI-player actions, with the AI player always choosing *some* valid action (not necessarily a good one)
5 (75/B)	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 valid available AI actions (without any invalid ones) from any valid state
5 (80/B+)	Chooses action to immediately win/not immediately lose game in cases where AI action would end the game

Score	Description
5 (85/A−)	Uses minimax to make good choices at least in the cases where a win or loss is just a few actions away (i.e. without needing heuristics)
10 (95/A)	Implements and uses a reasonable heuristic evaluating board states, with very positive scores corresponding to better boards for one player and very negative scores for the other, so the AI player makes good choices even when a win is not imminent †
5 (100/A+)	The computer is capable of playing as either player (black/white, red/blue, whatever) based on command-line options.

† These points unavailable for ICO.

The final version is due at **4pm on Tuesday, 22 February**.

Handing in

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