Project 2 - Minesweeper-o-Matic

The final project for CS4012 requires you to build an auto-playing implementation of the game *minesweeper*; use Threepenny to build the user interface and provide as complete an automatic player as you can.

Rules of Minesweeper:

Minesweeper is a single-player logic puzzle; the basic rules are simple (wikipedia has a good description as well).

Phase 1

- Model the game of Minesweeper in Haskell. With no UI or interaction model this part of the project is simple enough.
- Build a UI for the program from phase 1 using Threepenny GUI. You can specify the details of how the user interacts to play the game (i.e. what the game looks like and exactly what controls are used to select locations and clear or flag them). It's probably best to keep this simple.

For phase one the following are necessary:

- The program must distribute mines randomly around the grid
- Allow the user to uncover and flag mines (interactively)
- Detect the endgame condition

Phase 2

- Add a 'play move' button to the game from phase 1. Pressing this button will *attempt* to play a good, safe, move. I will note that Minesweeper is known to be NP-complete, which means that a well written and efficient solver is, well, hard! That link contains some very good commentaries and links on solving minesweeper. Check out his PDF presentation linked there.

For this part it is *necessary* that

- the program play a move automatically if there is an unambiguously safe move available

It is *desirable* that

- If a safe move is not available the program will play the least dangerous move available (see the links below for some discussion of the probabilities).

Documentation

You need to document your program clearly:

- Include suitable comments wherever there is a part of the program that deserves greater explanation

- Include a short document (text or PDF), about 2-3 pages will suffice, outlining the high level design choices you made, indicating how many of the project deliverables were satisfactorily completed. This should also include also a short (one or two paragraphs paragraph) reflection on the process of designing the program. Some questions you might consider here include: how suitable was Haskell as a language for tackling each phase of the project, what was your experience of the software development process (including things like testing and debugging), and what features of the language proved useful.

Grading

- The project is worth 35% overall
- 15% of the marks will be given for the documentation
- The basic implementation (phase 1) is worth up to 50% of the final marks. Your program needs to be
    - Well designed and commented
    - Accompanied by suitable documentation (this might be in the form of literate source files, or it may feature additional documents).
- The auto-player is worth 35% of the marks. At a minimum your player should be able to make simple and obvious moves (i.e. spot a few of the basic patterns as described in the Minesweeper wiki). The more sophisticated the player the better! Remember that you don't need to solve the whole game here (that's extremely hard), but try to implement at least *one* advanced tactic.

Submission:

Submit a zip file containing your project directory, ideally as a Stack project ready to be built and run. The documentation may be included in that directory, or submitted as an additional file.

The deadline for this assignment has been set as late as it possibly can be, and *no extensions are possible. If your assignment is not in by the end of the day on the 13th then I will not have it marked in time for the exam board.*

Resources

- Richard Kaye has the definitive discussion of NP-Completeness in minesweeper (of course).
- Here is a description (with a link to some Java sources) of one project (that got a writeup at hackaday.com)
- Some good strategic discussion at the Minesweeper Wiki
- Simon Thompson has some Haskell minesweeper code (reading someone elses design can make it hard to see your own creative solution, so I would counsel against reading too much of this at first)
- Sean Barrett has a nice discussion of calculating probabilities in minesweeper.
- Raphaël Collet wrote a paper about solving Minesweeper with constraints (Raphaël's put a copy on the web here, and a cached copy of the paper is linked from citeseer).
- Minesweeper even has a package in the Haskell hackage!