Minesweeper Documentation - Functional Programming

Ernests Kuznecovs

17332791

31/01/2021

Overview

The aspect of Haskell that interests me the most is discovering the best and most natural programming constructs a program can take on. With this interest I didn't let any opportunity of learning the best way to do something slip away. Unfortuately by doing this, I didn't leave much time for creating a complete solution for the program, and therefore missed out on some extra opportunities to see other programming constructs, although the small bits that I did spend time on I working through.

Deliverables

- Most of phase 1
  - Threepenny UI with minesweeper game logic.
  - Distributes mines randomly around the grid.
  - No flag placement.
  - No endgame detection.
- None of phase 2

Project Structure

The FRP constructs (Event a) and (Behaviour a) had the role of creating events and firing the actions that changed the state, and providing UI components with a sink from where to display values.

The FRP events triggered functions that used the state monad to handle logic and gave back the state without the monad back, and in turn the state was used as the value for the Behaviour.

The separation of concerns was very nice to have, the guaranteed state threading of the Behaviour events made it so that it was only necessary to think about the actual logic of the game.

Programming Experience

Cell Datastructure

It was not immediately apparent as to what data structure to use for the grid. I used a map but there was also an array, I think an array might've been more effecient to use and would be identical programatically.

FRP

Without using the great calculator example, which took quite some effort to mostly understand, I'd imagine it would've been very difficuly to be able to write programs effectively using threepenny.

State monad

At first my function signatures looked like :: MyState -> MyState, so then I took the time to think about how to incorparate the state monad into the program. It took a while to realise how to do it without contaminating the FRP part of the program, which I achieved by using runState in a function that is called by the Events that trigger.

Ugly Bits

The part I didn't like was using the record sysntax along with if then else statements. Instead of if then else, it seems like the alternative would be to use guards, but I found it difficult to see how it would make the syntax clearer.

Overall

Overall the biggest challenge about the programming was the actual understanding of the solution mathematically which means the programming language wasn't in the way. I can see the safety and elegance of Haskell being extremely useful for industrial back end engineering. Although I'm not sure if its the best for game programming as sometimes it seems its desireable to do very unpure things for the sake of performance.