1 My Program - Tetris

1.1 What is it

I made tetris. (Hetris). If for some reason uninitiated, this is a game where you place pieces in the order that they spawn in from the top of the screen. Try to pieces blocks so that you fill up rows. Once you fill up a row, this will add 1 point to your score and remove the row from the game. If your pieces stack up to the top of the screen and past it, you lose and will be taken to the game over screen, where you can see your final score and have the option to play again.

1.2 How to play

To start or restart the game press the spacebar. Use the arrow keys to move pieces around, and the spacebar to rotate pieces.

2 Architectural choices

2.1 Efficiency

I initially stored each piece seperately in a list and rendered each of these every frame, but realised this was fairly inefficient, especially if the game was to be scaled up, as my design allows for. I instead used a vector of vectors method to create rows and blocks within those rows, and then could assign each block as 'filled' or 'empty', and then render the blocks onto the screen according to their status.

2.2 Dynamism

The main thing I had in mind throughout the project was to make sure that any of the dimensions could be changed, and the game would still work perfectly. So screen width, height, and block size can all be changed by changing their respective variables and everything will work perfectly. This, besides being cool, also meant I had to create sturdy helper functions (eg. to convert from co-ordinates to indices and back) which could work for any layouts, and this reduced the ability of bugs to crop up in my code.

3 Library used - Gloss

I used the gloss library for this coursework, as it was one of the only options with which to make a game with, and this seemed like a more interesting and motivating challenge to do in haskell. Gloss was nice as it has a simple function in which to interact with IO, and then leaves you to create the rest of the program in your own style. It is extremely flexible.

4 Personal experience

Tough. At first, coming to grips with using Gloss and interacting with a new library was quite hard. However after a day or two of understanding how the library functioned it was much easier to write idiomatically, both for the library and haskell. Time restrictions also meant that a lot of time was put toward debugging and figuring out the cleanest way to structure code (architectural choices), rather than expanding the game and adding even more features or a nicer UI. There were times when I decided to rewrite whole parts of the code, as describe in 2.1.

5 Resources read (clickable links)

Hake by Dixonary
Making a Glossy Game Tutorial
Record Wildcards
Hackage Gloss
Hoogle