

# Heat — An Interactive Development Environment for Learning & Teaching Haskell

Olaf Chitil



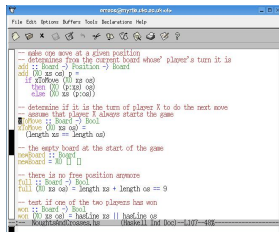
11<sup>th</sup> September 2008



# Emacs and IDEs

## Emacs, gvim etc. with Haskell mode

- not fully integrated (evaluate wrt. old program)
- too many confusing features
- GUI not nice enough



```
-- make one move at a given position
-- determines from the current board whose player's turn it is
add :: Board -> Board -> Board
add (0 0 0) p =
  if XMove (0 0 0)
  then (0 1 0) 0
  else (0 0 1) 0

-- determine if it is the turn of player X to do the next move
-- assume that player X always starts the game
XMove :: Board -> Bool
XMove (0 0 0) =
  (length xs == length os)

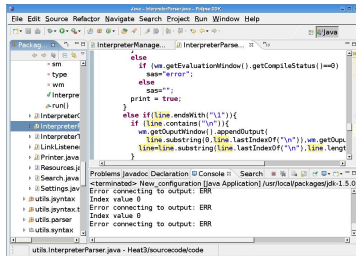
-- the empty board at the start of the game
newBoard :: Board
newBoard = 0 0 0

-- there is no free position anymore
full :: Board -> Bool
full (0 0 0) = length xs + length os == 9

-- test if one of the two players has won
won :: Board -> Bool
won (0 0 0) = hasLine xs || hasLine os
hasLine :: [Int] -> Bool -> Bool
```

## Professional IDE: Eclipse, Visual Studio

- too complex
- Haskell modes not mature
- hard to install



```
else
  if (vs.getEvaluationWindow().getCompileStatus() != 0)
    ssw = "error";
  else
    ssw = "";
    print = true;
}
else if (line.endsWith("\n")) {
  if (line.contains("\n")) {
    ws.getOutputWindow().appendOutput(
      line.substring(0, line.lastIndexOf("\n")).getBytes(),
      line.substring(line.lastIndexOf("\n"), line.length)
    );
  }
}
```

Problems | javadoc | Declaration | Console | Search | [Java Application] /usr/local/packages/jdk-1.5.0  
<-terminated> New\_configuration [Java Application] /usr/local/packages/jdk-1.5.0  
Error connecting to output: ERR  
Index value 0  
Error connecting to output: ERR  
Index value 0  
Error connecting to output: ERR

# Requirements

- ① integrates editor and interpreter console within a single user interface

# Requirements

- ① integrates editor and interpreter console within a single user interface
- ② provides a graphical user interface similar to professional IDEs

# Requirements

- ① integrates editor and interpreter console within a single user interface
- ② provides a graphical user interface similar to professional IDEs
- ③ simple and fool-proof:  
only features that support students beginning to learn Haskell

# Requirements

- ① integrates editor and interpreter console within a single user interface
- ② provides a graphical user interface similar to professional IDEs
- ③ simple and fool-proof:  
only features that support students beginning to learn Haskell
- ④ reliable

# Requirements

- ① integrates editor and interpreter console within a single user interface
- ② provides a graphical user interface similar to professional IDEs
- ③ simple and fool-proof:  
only features that support students beginning to learn Haskell
- ④ reliable
- ⑤ runs on all major platforms used by our students and staff  
(Windows, OS X, Linux, Solaris)

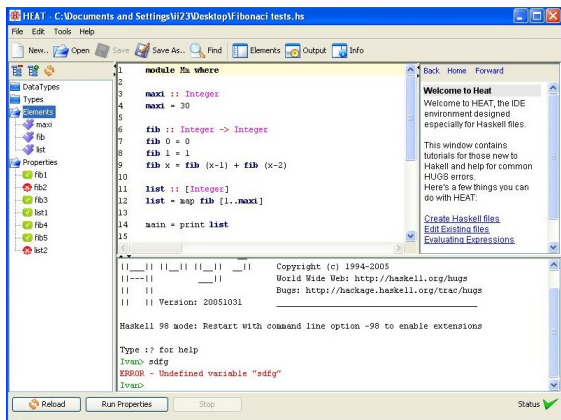
# Requirements

- ① integrates editor and interpreter console within a single user interface
- ② provides a graphical user interface similar to professional IDEs
- ③ simple and fool-proof:  
only features that support students beginning to learn Haskell
- ④ reliable
- ⑤ runs on all major platforms used by our students and staff  
(Windows, OS X, Linux, Solaris)
- ⑥ easy to install

# Requirements

- ① integrates editor and interpreter console within a single user interface
- ② provides a graphical user interface similar to professional IDEs
- ③ simple and fool-proof:  
only features that support students beginning to learn Haskell
- ④ reliable
- ⑤ runs on all major platforms used by our students and staff  
(Windows, OS X, Linux, Solaris)
- ⑥ easy to install
- ⑦ small source code and easy to maintain

# The Solution: Heat



## Editor

- for a single module
- syntax-highlighting
- matching brackets

Status: ? × ✓  
Interpreter console

- highlight prompt & errors
- error: source line & explanation

## Overview

- defined types, functions, ...

# Checking Properties

## Design recipe

(*How To Design Programs*)

- 1 purpose in comment
- 2 type declaration
- 3 example properties
- 4 actual definition
- 5 test

```
-- yield square of given number  
square :: Float -> Float  
square x = x * x
```

```
prop_square1 = square 2 == 4  
prop_square2 = square 0 == 0  
prop_square3 = square (-4) == 16
```

Only Boolean unit tests, general QuickCheck later.

Properties separate in overview pane and automatic testing.

# Composing Pictures

Simon Thompson *The Craft of Functional Programming*

A small library for making pictures compositionally:

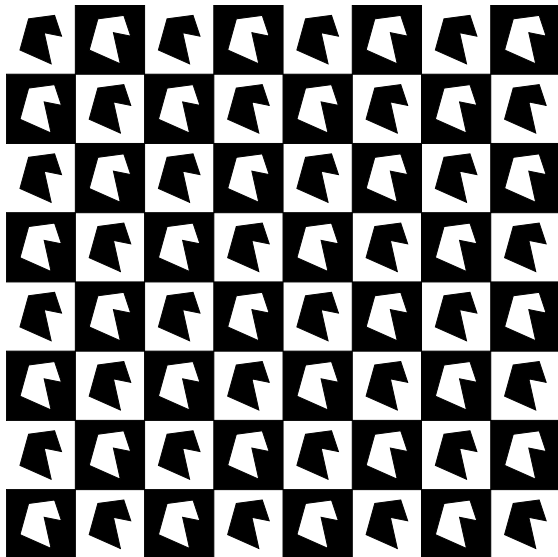
```
data Picture -- abstract

horse :: Picture
rotate90, flipV, flipH, invertColour
  :: Picture -> Picture
above , sideBySide, superimpose
  :: Picture -> Picture -> Picture
```

```
.....##...
.....##..#..
...##.....#.
..#.....#.
..#...#...#.
..#...###.#.
.#....#...##.
..#...#.....
...#...#....
....#..#....
.....#.#....
.....##....
```

ASCII art does not impress students!

- generate a PDF file
- automatically start PDF viewer
- separate from Heat



# Implementation

Written by final year project students.

- 15.000 lines of Java
- jEdit syntax package
  - lexing for syntax-highlighting
- Hugs as separate process
  - communication only via textual input/output stream
  - no interrupt signal: restart Hugs process
- new basic Haskell parser
  - Hugs' `:browse` and `:info` only works for valid Haskell
- graphics in PDF
  - incremental graphics model of PDF poor fit

*It's a great idea and much easier than using Hugs itself. However it was inconsistent on the Mac and a total disaster on Vista which with the number of both on campus (CS specifically) is not very useful. Preferences saving between sessions didn't work very well and the pop-up help did it's own thing. It has issues with line breaks too.*

- Reliability (spurious bug) main issue

*Syntax error in expression (unexpected ';', possibly due to bad layout) is all that needs to be said.*

- Fool-proof?

Enter full path of Hugs...

- Expect standard GUI features

shortcut keys, OS X shortcuts, automatic indentation, ...

- Missing or awkward features not a problem

no interrupt; expression input box

# Why Not Use the Glasgow Haskell Compiler?

- GHC's error messages more detailed but
  - many messages, no stop after first
  - mention language extensions: `fst :: forall a b. (a,b) -> a`
  - use substantial indentation
  - error location confusing, if in input expression
- GHC is harder to install
- GHC does not have built-in `observe` (future debugging)

Still, Heat should support GHC in future.

# Big Role Model: DrScheme



Unfortunately relies on close integration of IDE with compiler and runtime system.

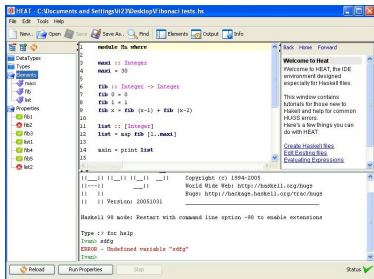
Different language levels with different error messages highly desirable, especially language with limited classes and without partial application.

Not many!

- include library documentation
  - Haddock: extensible, standard
  - handwritten: Prelude for novices (several levels?)
- spelling help
- support full QuickCheck
- debugging via observing functions

# Conclusions

- small but effective wrapper IDE possible
- reliability most important, not features



The screenshot shows the Heat IDE interface. The main editor displays a Haskell program named 'fib.hs' with the following code:

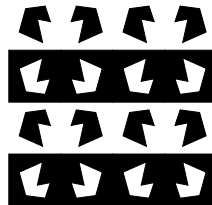
```
1 module Fib where
2
3 main = do
4   main <- 30
5
6   fib <- [Integer -> Integer
7         fib 0 = 0
8         fib 1 = 1
9         fib x = fib (x-1) + fib (x-2)]
10
11 list <- [Integer]
12 list = map fib [1..main]
13
14 main = print list
15
```

The output window at the bottom shows the execution results:

```
Copyright (c) 1994-2003
World Wide Web: http://haskell.org/hugs
Bug: http://backlog.haskell.org/rcc/sugs
Version: 20031031

Haskell 98 mode: Restart with command line option -98 to enable extensions

Type :> for help
[eval]: aStrg
ERR00 - Undefined variable "aStrg"
[eval]
```



## Availability

- Heat 1.1  
<http://www.cs.kent.ac.uk/teaching/resources/haskell/heat.html>
- Heat 3.0  
real soon ... watch Haskell mailing list