Functional Programming: Minesweeper

Laurent Christophe, Wolfgang De Meuter

Programming Project: Assignment #2 (2014 - 2015)

1 Introduction

The final mark for the Functional Programming course is based 50% on a programming project (25% on the first programming assignment, 25% on a second programming assignment) and 50% on the oral exam. This document describes the second programming assignment.

Submission is done by sending your raw code files to the teaching assistant: Laurent Christophe (`lachrist@vub.ac.be`). The assignment is due 11 January 2015 (8AM). You will have to defend the complete project individually on a date that will be communicated later (see exam schedule of the faculty). Notice that the execution of the project is strictly individual and no plagiarism shall be tolerated!

The project will be marked according to how well you fulfil the functional requirements and according to how well you apply the concepts explained during the lectures and the lab sessions. If you encounter any problem or if you have a precise question, feel free to contact the assistant at `lachrist@vub.ac.be`.

2 Requirements

The goal of this programming assignment is to implement a graphical user interface for your text based Minesweeper of the first programming assignment. If the user left-clicks on a cell, its contents should be revealed. This corresponds to a click action in the first programming assignment. If the user right-clicks on a cell, a flag should be dropped on that particular cell. At the end of the game (independent of wether it is won or lost) the entire board should be revealed. Additionally, your graphical interface should include the following elements:

1. A standard "exit" button.
2. A "reinitialize" button to start a new game.
3. A timer that starts counting at the the time of the first user click.
4. A mine counter indicating the number of mines left (based on the number of flags and not the number mines actually spotted).

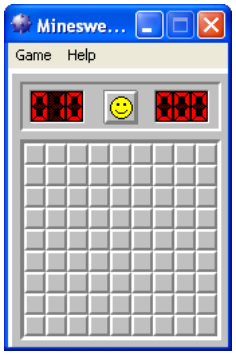

Figure 1: Minesweeper in WindowsXP.

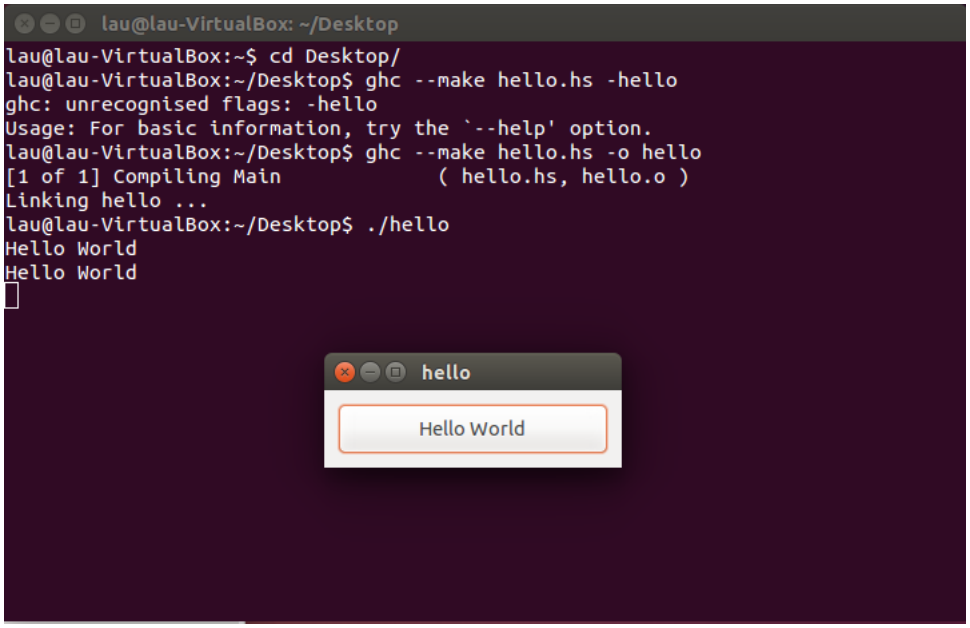

Figure 2: A "hello world" window created with Gtk2Hs.

3 Graphical Library

We advise you to use the Gtk2Hs graphical library to implement this second assignment. Gtk2Hs is a Haskell binding to Gtk+. It allows one to write Gtk+ based applications with GHC. Gtk2Hs is one of the most mature and robust graphical libraries for GHC. According to the installation page (`https://www.haskell.org/haskellwiki/Gtk2Hs/Installation`), Linux and Windows users will not need to struggle to get Gtk2Hs running. Running it on an OSX system is a little bit more involved (see further below). Once Gtk2Hs is installed you can test your setup with the below Haskell code:

```
import Graphics.UI.Gtk

main :: IO ()
main = do
  initGUI
  window <- windowNew
  button <- buttonNew
  set window [ containerBorderWidth := 10, containerChild := button ]
  set button [ buttonLabel := "Hello World" ]
  onClicked button (putStrLn "Hello World")
  onDestroy window mainQuit
  widgetShowAll window
  mainGUI
```

```
ghc --make Hello.hs -o hello
```

For further documentation, please refer to the project homepage: `http://projects.haskell.org/gtk2hs/`.

3.1 For Mac Users

As stated before, Mac users might experience some problems when trying to install Gtk2Hs. If you do not manage to get it installed on a particular OSX configuration, we advise you to use VirtualBox and install Ubuntu on it. The following steps will get Gtk2Hs running on your system:

1. Download and install VirtualBox (`https://www.virtualbox.org/wiki/Downloads`)
2. Download Ubuntu (`http://www.ubuntu.com/download`).
3. Create a new Linux/Ubuntu virtual machine and place the Ubuntu iso file in the virtual CD/DVD trailer.
4. Start the virtual machine and Install Ubuntu.
5. Run:

```
sudo apt-get install virtualbox-guest-dkms virtualbox-guest-utils virtualbox-guest-x11
sudo apt-get update
sudo apt-get install ghc
sudo apt-get install libghc-gtk-dev
```

6. Reboot the virtual machine.