Project Sheet

FritzLight Project - Controlling 30x12 RGB Pixels Individually

TUHH STS Software Technology Systems

Functional Programming - Winter 2016/2017 - January 9, 2017 - Schupp/Mattsen/Lehmann

Introduction

We offer the FritzLight project as an interactive way for you to get in contact with the input and output capabilities of Haskell, and to solidify your current Haskell knowledge in a more practical way. In this project, you will learn how to control the pixels of a screen individually in a functional way, based on user-given input. The FritzLight system consists of two parts: The emulator, which is fed with RGB data for each pixel to display it on a screen , and the user-defined program which handles input as well as the actual computation of pixel data. Your task is to write such a custom program in groups, creating a small game or other visual effects with interactive components (including input to the system).

How to compile my own project code for the FritzLight environment?

a) Install `stack`[1]

b) Download and extract `HSFritzLight.zip` from StudIP

c) Run `stack setup`

d) Run `stack build`

e) Use "`stack ghci`" instead of "`ghci`"

- `stack ghci`
- Then `:l YourFileName.hs` (`Main.hs` for the test project)

f) Use "`stack ghc --`" instead of "`ghc`"

- `stack ghc -- --make YourFileName.hs` (`Main.hs` for the test project)

What do we expect for the course of this project?

Generally spoken, we expect you to be motivated to actually *contribute* to your group project. Be prepared to explain your intermediate concept and implementations during the appointment session on Monday, January 16, as well as the final program on presentation day. Come up with your own ideas, try out whatever comes to your mind, be creative! Use the 360 pixels to create something new, exciting, astonishing (or boring, it is all up to you!).

[1] https://www.haskellstack.org/

How do I start with my own project?

The back-end (e.g., events creation, timing behaviour,...) is already implemented in the provided project library. To help you getting started with your own FritzLight project, we provide a sample program which allows the user to move a single pixel over the screen (output) by pressing the h, j, k, or l key (input).

```
1  module Main where
2  import Network.MateLight.Simple
3
4  import Data.Maybe
5  import qualified Network.Socket as Sock
6
7  move :: (Int, Int) -> String -> (Int, Int) -> (Int, Int)                    Main Functionality
8  move (xdim, ydim) "\"j\"" (x, y) = (x, (y + 1) `mod` ydim)
9  move (xdim, ydim) "\"k\"" (x, y) = (x, (y - 1) `mod` ydim)
10 move (xdim, ydim) "\"h\"" (x, y) = ((x - 1) `mod` xdim, y)
11 move (xdim, ydim) "\"l\"" (x, y) = ((x + 1) `mod` xdim, y)
12 move _ _ x = x
13
14 toFrame :: (Int, Int) -> (Int, Int) -> ListFrame
15 toFrame (xdim, ydim) (x', y') = ListFrame $ map (\y -> map (f y) [0 .. xdim - 1]) [0 .. ydim - 1]
16   where f y x = if x == x' && y == y' then Pixel 0xff 0xff 0xff else Pixel 0 0 0
17                 List of Keyboard Inputs   State          Frame          New State
18 eventTest :: [Event String] -> (Int, Int) -> (ListFrame, (Int, Int))
19 eventTest events pixel = (toFrame dim pixel', pixel')
20   where pixel' = foldl (\acc (Event m ev) -> if m == "KEYBOARD" then move dim ev acc else acc) pixel events
21
22 dim :: (Int, Int)
23 dim = (30, 12)
24
25 main :: IO ()
26 main = Sock.withSocketsDo $                  IP of pixelkino.haskell.de Port   Delay between frames in microseconds   Initial State
27   runMate (Config (fromJust $ parseAddress "134.28.70.172") 1337 dim (Just 500000) False []) eventTest (0, 0)
```

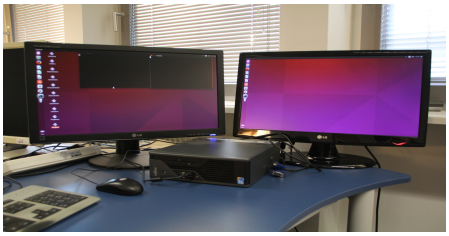
In the `main` function, four parts are of importance for your work: The IP address (which you will have to adapt according to the emulator device you want to communicate with[2], the delay between each sent frame (choose a value between 33000 (33ms) and 500000 (500ms)), the actual event function for your own processing of input and generation of output (repeatedly called after the desired delay), and the initial state of the system. The system state is not necessarily a two-element-tuple, but can be any data type that you want to use for data which is supposed to be persistent between events, such as the positions of dynamic pixels or letters (only make sure that the type corresponds to the one you are expecting in the contract of your event function).

The main functionality in this example is implemented via two functions: The eventTest function processes keyboard input (obtained from the list of input *events*), and moves the highlighted pixel accordingly. The toFrame function takes the information about the highlighted pixel and creates a complete assignment (`ListFrame`) of every single pixel to its desired RGB color data (in this case, the pixel to be highlighted is set to white (`0xff 0xff 0xff`), while the remaining ones are set to black (`0 0 0`).

If you need random data in any of your functions, replace the `runMate` function in `main` with `runMateRandom`, and add an additional first argument to your event function that will contain an infinite list of random `Int` values (the new contract would then be `eventTest :: [Int] -> [Event String] -> (Int, Int) -> (ListFrame, (Int, Int))`).

[2] online emulator: 134.28.70.172 | emulator in room E4.036 (via switch): 192.168.23.2 | FLight hardware in room E4.036 (via switch): 192.168.23.1

(a) A Switch



(b) A Computer

Fig. 0.1: The switch and the computer

Which possibilities do I have to output my pixel data graphically?

a) *Online* using the emulator at `http://pixelkino.haskell.de`
   - The IP address of the online emulator is `134.28.70.172`
   - Use a browser to view the available screens, click on your IP (to determine IP, see below)

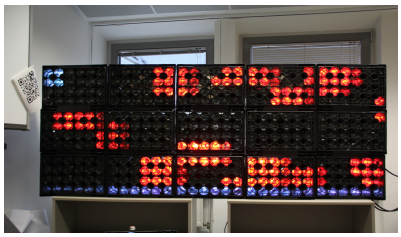
b) In room *E4.036*: On the SmartBoard (Fig. 0.2b), or the FritzLight hardware

   Connect (wired) to the *switch* (Fig. 0.1a, next to the FritzLight).
   - Use the emulator on the SmartBoard, available at `192.168.23.2` (PC with mouse is in grey shelf, use guest account, open *firefox*)
   - Control the FritzLight at `192.168.23.1`
   - You may also execute sample programs using the FritzLight PC (Fig. 0.1b, use guest account)

c) Install the emulator on your own system (advanced)

Useful links for the project

- Online MateLight emulator on `http://pixelkino.haskell.de`
- Determine your own IP address on `https://www.heise.de/netze/tools/meine-ip-adresse/`
- MateLight library on `https://github.com/bigmac2k/HSMateLight`, and MateLightEmu on `https://github.com/bigmac2k/MateLightEmu`



(a) The FritzLight running an Obstacles game

(b) The emulator running a Pong game

Fig. 0.2: Emulators in hard- and software

Task 1 - Find and Register a Group

Form a group of four (4) members, which you will work with for the complete duration of this project. Make sure that you distribute your contributions to the project equally, so that all group members are able to explain their parts both during the intermediate appointment session and the final presentation.

For the group registration, open the "Lecture: Functional Programming" group on StudIP, navigate to "Participants", and assign yourself to one of the groups there. In case you cannot find a group to work with, send us an email and we will see what we can do about that.

Task 2 - Demonstrate your setup - 0.5 Point

Compile the sample dot project, make sure that you can communicate with the online emulator, and show us that everything is working at the beginning of the intermediate appointment session (Monday, January 16).

Task 3 - Explain your concept during intermediate appointment - 1 Point

The intermediate appointment session is mandatory for all project participants. For that session, you are supposed to be able to explain both the overall *project concept* of your group, as well as the individual parts that you have already contributed or will contribute to the final result. Additionally, you are supposed to write down your concepts in a `Readme`-file (e.g., what does your program do? How is it working? How is the user supposed to interact?), which you will add to your final result submission.

Task 4 - Present your project on presentation day - 1.5 Point

The presentation session is mandatory for all project participants as well. You will demonstrate your final project result, and explain the core parts of your implementations. At the latest with the beginning of the presentation session, your project needs to be uploaded as `FritzLight_[YourGroupNumber].zip` to the provided project folder on StudIP, containing the project files, the readme, and a list containing the group number as well as the name of all group members.

DEADLINE: 13:15, February 02, 2017

2017-02-02T13:15:00+01:00