Methods for scientific research

Assignement 2 - Research setup and proposal

Functional Reactive Programming and its speed contribution to GUIs

Keywords : functional programming, reactive programming, desktop graphical user interfaces, efficiency

Bruno Rocha Pereira - 0529512

Wordcount : 701

1 Introduction

Functional Reactive Programming is a programming framework that stands on two main ideas : *behaviours* and *events*. Reactive programs need to answer to internal and external stimuli - *events* - with a defined behaviour, which is exactly what a user interface does. [Wan and Hudak, 2000] One of the most advantageous side of FRP is the introduction of datatypes that can observe values changing over time and thus react as soon as the change happens. [Kraeutmann and Kindermann, 2015]

Previous work in the FRP research field has shown that using it was a great improvement to create GUIs. However, this improvement has not really been quantified. The aim of this research is to define the speed difference as well as propose an improvement of the speed, implementing a programming language with a natively supported GUI library. Comparisons between OO programming and Functional reactive programming have already been made [Salvaneschi et al., 2014]. However, this research will only use pure functional programming and will thus avoid object-oriented programming.

2 Research Questions

2.1 What is the speed and memory usage difference between reactive and non reactive programming in the case of GUI and 3D?

Reactive programming brings a whole new vision of data that is more suitable for event-driven softwares and show greater performances spped-wise in such cases. [Bainomugisha et al., 2013] The point of this study is to quantify the speed difference that reactive programming brings to a software with a GUI. This will also give a good knowledge of the current level of FPS(Frames Per Second) reached by reactive and non reactive applications in specific cases.

2.2 Could a programming language be implemented with a built-in reactive 3D library?

The programming language Elm has already been designed for GUI purpose on a FRP basis. However, the Elm compiler produces css, html and javascript, thus aimed at web pages. [Czaplicki, 2012]. The aim of this reseach is to implement a new functional reactive programming language aimed at creating GUIs for a desktop application - and more particularly games - purpose without any external library. The implemented language will have to handle reactive OpenGL bindings for the 3D part.

3 Methods

3.1 Implementing a software with a GUI

The goal of this method is to implement a small game with a GUI using reactive programming and non reactive programming. These softwares have to be implemented using the same language - Haskell - to avoid bias due to the language itself. The two libraries that will be used for the 2D GUI are *wxHaskell* for the non reactive programming part, and *reactive banana* for the FRP one [Apfelmus, 2011]. The 3D part will be implemented using OpenGL. Haskell has two libraries that bring OpenGL Utility Toolkit bindings to it.The first one is the simple implementation of GLUT in haskell [Panne, 2014] while the other one combines glut with reactive [Elliott, 2008]. The observation of the FPS variation gives a good indication of the performances difference between the two implementations. The last step of this research is to implement a basic FRP language with a built-in reactive GUI. The new created FRP language will then be compared to existing functional reactive framework such as *reactive banana* for Haskell. To allow the comparison, the same desktop application will be implemented. The exact same operations will then be applied manually on the user interface - several times to avoid bias due to the user reaction time. Haskell provides many tools for profiling [O'Sullivan et al., 2008]. Correct profiling tools must then also be implemented in the newly created

language in order to compare in the most precise way the 3 implementations.

Several 3D scenes will be generated in order to identify the pros and cons of each implementation. Each scene will have different complex elements to render. The different FPS values and the memory usage will be extracted and saved for a comparison. This will bring several points of comparison and allow to determine whether an implementation is generally more efficient than the other or if each of them has a more specific usage.

4 Outcome

A chart presenting precise FPS values and memory usage of the different softwares in several situations will be realized. The real contribution of the FRP in 3D and GUIs will be thus meticulously quantified. However, a functional programming language fully implemented for a 3D animation purpose is the main goal to achieve. The implemented language will compute 3D environment in a reactive way using only functional programming. The computation speed will overtake haskell's one since it will avoid any feature not GUI or 3D related. This will allow more interactive and more reactive 3D softwares such as video games, medical visualization tools and engineering tools.

References

[Apfelmus, 2011] Apfelmus, H. (2011). reactive-banana: Library for functional reactive programming (frp). `http://hackage.haskell.org/package/reactive-banana-0.1.0.0`. Accessed: 01-11-2015.

[Bainomugisha et al., 2013] Bainomugisha, E., Carreton, A. L., Cutsem, T. v., Mostinckx, S., and Meuter, W. d. (2013). A survey on reactive programming. *ACM Computing Surveys (CSUR)*, 45(4):52.

[Czaplicki, 2012] Czaplicki, E. (2012). Elm: Concurrent frp for functional guis. *Master thesis, Harvard University.*

[Elliott, 2008] Elliott, C. (2008). Connects reactive and glut. `http://hackage.haskell.org/package/reactive-glut`. Accessed: 05-11-2015.

[Kraeutmann and Kindermann, 2015] Kraeutmann, D. and Kindermann, P. (2015). Functional reactive programming and its application in functional game programming. `http://verify.rwth-aachen.de/proseminar/PK15/ausarbeitungen/reactive_programming.pdf`.

[O'Sullivan et al., 2008] O'Sullivan, B., Goerzen, J., and Stewart, D. B. (2008). *Real world haskell: Code you can believe in*. O'Reilly Media, Inc.

[Panne, 2014] Panne, S. (2014). Glut: A binding for the opengl utility toolkit. `https://hackage.haskell.org/package/GLUT`. Accessed: 03-11-2015.

[Salvaneschi et al., 2014] Salvaneschi, G., Amann, S., Proksch, S., and Mezini, M. (2014). An empirical study on program comprehension with reactive programming. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 564–575. ACM.

[Wan and Hudak, 2000] Wan, Z. and Hudak, P. (2000). Functional reactive programming from first principles. In *ACM SIGPLAN Notices*, volume 35, pages 242–252. ACM.