Functional Reactive Augmented Reality: Proof of concept using an extended augmented desktop with swipe interaction

Conference Paper · October 2016

CITATIONS
0

READS
26

3 authors:

J. Adam Jones
University of Mississippi
39 PUBLICATIONS **496** CITATIONS
SEE PROFILE

Conrad Cunningham
University of Mississippi
65 PUBLICATIONS **332** CITATIONS
SEE PROFILE

João Paulo Oliveira Marum
University of Mississippi
7 PUBLICATIONS **4** CITATIONS
SEE PROFILE

EUROGRAPHICS '0x / D. Reiners, D. Iwai, and F. Steinicke
(Editors)

Volume 0 (2016), Number 0

Functional Reactive Augmented Reality: Proof of concept using an extended augmented desktop with swipe interaction

Joao Paulo Oliveira Marum, J. Adam Jones, and H. Conrad Cunningham

Department of Computer and Information Science
University of Mississippi
University MS USA

Abstract

Functional Reactive Programing (FRP) is a new paradigm that has been evolving and used in many different fields, i.e.: robotics, user interface and others. FRP was originally developed as a programming paradigm to handle animation and it has since been used as a replacement for the observer pattern in systems that require order and accuracy when handling large numbers of callbacks. Inaccuracy and ordering issues related to callbacks can negatively affect a user's experience when using a virtual environment (VE). We propose a framework for virtual environments, that we believe will improve the design of them through the use of functional reactive capabilities. As a proof of the concept, we are currently developing a FRP-based augmented desktop system that utilizes multiple virtual and real workspaces and gesture interactions for switching between desktop contexts.

1. Introduction

Functional Reactive Programming (FRP) can be described as programming with asynchronous data streams. This gives us the opportunity to remove callbacks, and consequently "callback hell", and replace the typical observer pattern. A major advantage of FRP is that it enables propagation of change and has a toolbox of functions to combine, create and filter any data. A great deal of FRP efforts have been directed toward computer graphics, but only a few of these involved VEs. Of these, none have examined current virtual environment technologies e.g. Kinect, Oculus and Leap Motion. The paradigm used for building software with callbacks and the observer pattern entails many problems. As stated by Blackheath and Jones(2016) [BJ16] the callback plagues us with "unpredictable order in which events are received, difficult to guarantee that you have registered your listeners before you send the first event. messy state machine style, threading issues, difficult to guarantee that no more callbacks will be received after unregistering a listener, leaking callbacks and accidental recursion". These problems have the potential to affect the experience of VE users, causing inaccuracy , event ordering issues and rendering problems. In our research, we are aiming to determine if VE design can be improved through the use of FRP, and if it is possible to integrate existing FRP tools into common VE programming environments. We believe this to be the case as FRP has shown to be beneficial for dynamic environments such as user interfaces and robotics. For instance, in user interface developers want elements of their software to change or update as the user interacts with it. In robotics, developers want that their autonomous devices to respond to a dynamic and sometimes unpredictable world. Virtual environments have many of these same needs. The world itself becomes a spatial interface and must also respond to the user's actions, movements and behaviours. VEs must respond to the sometimes unpredictable actions of the user. These key elements make FRP beneficial for interfaces and robotics, therefore it seems reasonable to speculate that VEs can benefit from FRP as well. Our goal is to improve software engineering procedures for VEs while also improving their responsiveness. We are currently testing the integration of existing FRP tools, like Sodium and Elm, into existing VE tools, such as Unity3d. We are also developing a FRP proof-of-concept to explore and demonstrate whether or not this is a feasible approach. This proof-of-concept uses an augmented environment to impose the multiple, low resolution virtual desktops around a high resolution, physical monitor containing the real desktop. Reactive gesture detection enables the user to select a virtual desktop and transfer it into the real monitor, where it can be seen and interacted with in high resolution. This is conceptually similar to the focus-plus-context screens described by Baudisch et al(2001) [BGS01].

2. Related Work

Much work has examined both FRP and VEs separately, but very little work has looked at their combination. For the FRP side, Czaplicki and Chong(2013) [CC13], and Blackheath and Jones(2016) [BJ16], these papers examine the development of Elm

and Sodium, the two possible candidates examined in this paper. There are also works that focus on the development and design of virtual environments, such as Furness and Barfield(1995) [FB95]. We also examined similar works that mixed FRP and VE. Blom and Beckhaus(2007) [BB07] examines the development and practical results of the development of a Functional Reactive Virtual Reality systems. However, this work is no longer contemporary and may not lend itself well to modern VE development environments. Kraeutmann and Kindermann(2015) [KK15] describe the use of reactive programming on the development of a "pong"-like game with no virtual environment components. Kawai(2014) [Kaw14] creates a Reactive Extension for Unity, which extends the Unity3D object set, but it is motivated by solving web connections issues related. Xie et al(2014) [XHC*14] develops a game extending the Model-View-Controller (MVC) pattern, by reactively respond to inputs from user, but it only examines inputs coming from mouse. The next ones are non FRP related VE applications that inspired the choice of our conceptual tool. Baudisch et al(2001) [BGS01] examines a Focus plus context which has a low resolution projection of the desktop in the wall, and a high resolution monitor that works as a magnifier, enhancing the visualization of a particular area while the rest remains visible, DiVerdi et al(2003) [DNH03] which presents a 3D augmented reality desktop, and Jones et al(2013) [JBOW13] examines a tool that augments the physical environment surrounding a television to enhance interactive experiences.

3. Method

Among the many frameworks and tools available for the development of virtual environments, we chose Unity3D, which relies on the tight correspondence between the graphical workspace and the script code written in UnityScript (proprietary language based on JScript.Net) or C Sharp. Every element in the workspace exists as an object with attributes that can be read or modified, allowing for sensors and input devices to affect the scene. As the FRP component in our project, we have selected candidate frameworks based on their maturity, reliability, extendability, and whether or nor their native language is supported by Unity3D. Based on these criteria, we have selected two candidate frameworks: Elm and Sodium. Sodium is a functional reactive programming library for multiple languages. It has versions for C++, C Sharp, Java, Kotlin, Scala, and Typescript/Javascript. It is based on Flapjax, Yampa, scala.React, and a number of other Functional Reactive Programming efforts. Sodium is also based on the idea of data cells and streams. A Stream is represented by a list of time/value pairs describing the events within stream. A Cell is represented by a pair (initial value, steps): the initial value pertains to all times before the first step, and the time/value pairs give the discrete steps in the cell's value These objects can be anything predefined within the language or user-defined types even types coming from other libraries. Elm is a concurrent FRP language focused on easily creating responsive GUIs. Elm has two major features: (1) purely functional graphical layout and (2) support for Concurrent FRP. Purely functional graphical layout is a high level framework for working with complex visual components. It makes it quick and easy to create and combine text, images, and video into rich multimedia displays. Concurrent FRP solves some of FRP's long-standing efficiency problems: global delays and needless recomputation. Together, Elm's two major features simplify the complicated task of creating responsive and usable graphical user interfaces.

4. Future Work

We are currently testing the candidate frameworks' compatibility with Unity3D. Meanwhile, we are developing the prototype of the gesture detection in Unity and expanding this prototype to work with the virtual desktops soon. As soon as we have a fully functional augmented system prototype, it will be modified in a reactive way, replacing the callbacks and synchronous calls for reactive asynchronous data streams. The FRP-based augmented desktop system will have its performance compared with the original augmented system to verify whether the benefits in the user's and developer's experience stated were achieved or not.

5. Acknowledgement

This work has been done with support from CAPES, Coordination for Enhancement of Academic Level Individuals - Brazil"

References

[BB07] BLOM K. J., BECKHAUS S.: Supporting the creation of dynamic, interactive virtual environments. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology* (2007), ACM, pp. 51–54. 2

[BGS01] BAUDISCH P., GOOD N., STEWART P.: Focus plus context screens: Combining display technology with visualization techniques. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2001), UIST '01, ACM, pp. 31–40. URL: http://doi.acm.org/10.1145/502348.502354, doi:10.1145/502348.502354. 1, 2

[BJ16] BLACKHEATH S., JONES A.: *Functional Reactive Programming*. Manning Publications Co., New York, USA, 2016. 1

[CC13] CZAPLICKI E., CHONG S.: Asynchronous functional reactive programming for guis. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation* (New York, NY, USA, 2013), PLDI '13, ACM, pp. 411–422. URL: http://doi.acm.org/10.1145/2491956.2462161, doi:10.1145/2491956.2462161. 1

[DNH03] DIVERDI S., NURMI D., HOLLERER T.: Arwin-a desktop augmented reality window manager. In *Proceedings of the 2Nd IEEE/ACM International Symposium on Mixed and Augmented Reality* (Washington, DC, USA, 2003), ISMAR '03, IEEE Computer Society, pp. 298–. URL: http://dl.acm.org/citation.cfm?id=946248.946839. 2

[FB95] FURNESS T. A., BARFIELD W.: *Virtual Environments and Advanced Interface Design*. Oxford Publications, Oxford, UK, 1995. 2

[JBOW13] JONES B. R., BENKO H., OFEK E., WILSON A. D.: Illumiroom: Peripheral projected illusions for interactive experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, ACM, pp. 869–878. URL: http://doi.acm.org/10.1145/2470654.2466112, doi:10.1145/2470654.2466112. 2

[Kaw14] KAWAI Y.: Reactive extensions for unity. https://github.com/neuecc/UniRx, 2014. 2

[KK15] KRAEUTMANN D., KINDERMANN P.: Functional reactive programming and its application in functional game programming. 2

[XHC*14] XIE Y., HOFMANN H., CHENG X., ET AL.: Reactive programming for interactive graphics. *Statistical Science 29*, 2 (2014), 201–213. 2