AppRC: A framework for integrating mobile communication to virtual reality applications

Bernardo F. V. Pedras
Tecgraf/PUC-Rio
bpedras@tecgraf.puc-rio.br

Alberto Raposo
Tecgraf/PUC-Rio
abraposo@tecgraf.puc-rio.br

Ismael H. F. Santos
Petrobras Research Center
ismaelh@petrobras.com.br

Abstract

The popularity of smartphones and tablets has grown immensely in the last years and many different kinds of applications can take advantage of mobile functionalities. In special, applications that run on engineering virtual environments stand to benefit the most of a second screen for input and output. Such applications can use tablets or smartphones to help navigate the virtual world or even to access and easily view lots of data in the form of tables or graphs without disturbing the virtual world representation on the main screen. Another big advantage that mobile devices bring to the engineering applications is the capability of accessing the data on remote locations, like on oil platforms or refineries, and so allowing the field engineer to check the data or even change it on the fly. The problem is that most engineering applications are very complex and there is no easy way to implement the communication without a lot of programing effort on the application side. We propose a framework that is easily integrated with any application and adds external communication channels, adding functionalities like: collaboration with many devices, concurrency checks and handling of multiple instances of the application. As a first use of the framework, we discuss the implementation of EnvironRC, a communication channel for the 3D engineering visualization tool Environ.

Keywords: Mobile-Communication, Mobile-Integration, Collaboration, 3D-Interaction

1 Introduction

The integration of mobile functionality is a new demand for many different kinds of applications and services. Nowadays it is almost expected of an application or a service to have some kind of mobile integration. The first kinds of services to utilize mobile platforms were, for the most part, related to mobile phones, social networks or media playback. But with the computational power of current smart devices, a lot of other kinds of applications can utilize the smart devices to enhance its functionalities.

There are a lot of benefits that a mobile device can bring when integrated to an application. The first one is the obvious gain in mobility. No longer does the user need to be on a desktop inside an office to access or manipulate the data. A second benefit are the included sensors present in most smart devices. Sensors like GPS-Location can, for instance, add important usability information to the application so that it can respond according to the user's location. Most smart devices also have sensors like accelerometers and multi-touch screens that can help interact with the application, helping to navigate in a 3D-world, or to see large amounts of data, for example.

We are especially interested in the use of mobile devices to aid the visualization of engineering data and simulations. Many applications and simulators have been developed over the years for traditional desktops and/or clusters. It is still impossible to run such simulations or applications on a mobile device as a standalone application because of performance issues, and even if it were possible to run entire simulations on smart devices, it wouldn't be the best way to do it because all the generated data must be stored and centralized in databases or servers, and it wouldn't be practical to transfer such amounts of data for each simulation. So, mobile devices should be used to add its benefits to an existing application or simulator, creating a collaborative environment between the main application (running all the heavy work) and different connected mobile devices. And exactly at this point comes the main problem we try to address with our proposal. How to build a collaborative environment between an existing application and mobile devices with minimum reworking of the existing application?

We present a framework, callled AppRC, that is easily integrated with any application and adds external communication channels, adding functionalities like: collaboration with many devices, concurrency checks and handling of multiple instances of the application. As a first use of the framework, we discuss the implementation of EnvironRC, a communication channel for the 3D engineering visualization tool Environ [Raposo et al. 2009]. Although our focus is on its use for immersive virtual reality applications, AppRC is flexible enough to be used in other kind of applications, needing only a command passing interface.

This paper is organized as follows. The following section presents related works. In Section 3 we present a brief overview of the Environ application and its use. Section 4 discusses the structure of our framework and how it works. Section 5 explains in detail the implementation work that needs to be done in order to use the framework (extension points). Section 6 presents the use-case example of utilizing the framework with Environ. Finally, conclusion and future work follow in Section 7.

2 Related Work

The AppRC framework started as a specialization of the much bigger and generic collaboration enviroment called CEE (*Collaborative Engineering Enviroment*) [Santos et al. 2008]. The focus of CEE was to create a collaboration framework where many different users can work at the same time across different platforms. However, CEE requires a network and server infrastructure, which is not suited for the integration of mobile devices.

There are many applications that utilize mobile devices to enhance the VR experience. Most of them focus on implementing efficient ways to navigate or to interact with the 3D world. Medeiros et al. [Medeiros et al. 2013] presents an efficient way to navigate in an immersive environment with the use of a tablet. Further discussions and concepts for navigation with mobile devices can be found in [Noronha et al. 2012] and [Guimarães et al. 2004].

VRCAI 2013, November 17 – 19, 2013, Hong Kong.

ACM 978-1-4503-2590-5/13/11

Most implementations of navigation functionalities on VR systems rely on the VRPN *(Virtual-Reality Peripheral Network)* [Taylor et al. 2001], which is a device-independent and network-transparent system for accessing virtual reality peripherals in VR applications. VRPN provides a communication standard for developers who want to utilize different sensors as inputs to their application. Although the VRPN framework is widely adopted as the communication channel for navigation input information, it does not provide a good way to handle other kinds of information, like symbolic input, large amount of data or more complex data structures like photos or videos.

LVRL *(Lightweight Virtual Reality Libraries)* [Teixeira et al. 2012] is a way to port an application to a VR system. It also defines an input communication channel that is based on the VRPN implementation, leaving some room for the developer to implement custom channels. Since the main goal of LVRL is to port the entire application, requiring reworking of the application to use the LVRL, there is no concern in keeping the communication channel generic.

3 Environ

Environ [Raposo et al. 2009] is a tool designed to allow visualization of massive CAD models and engineering simulations both in desktop and VR immersive environments. It is a system composed of a 3D environment for real-time visualization and plug-ins to import models from other applications, allowing users to view and interact with different types of 3D data, such as refineries, oil platforms, risers, pipelines and terrain data (Figure 1). It enables the user to view the simulation in real-time as a 3D environment and enables, at the same time, the user to view all the simulation data and engineering information.

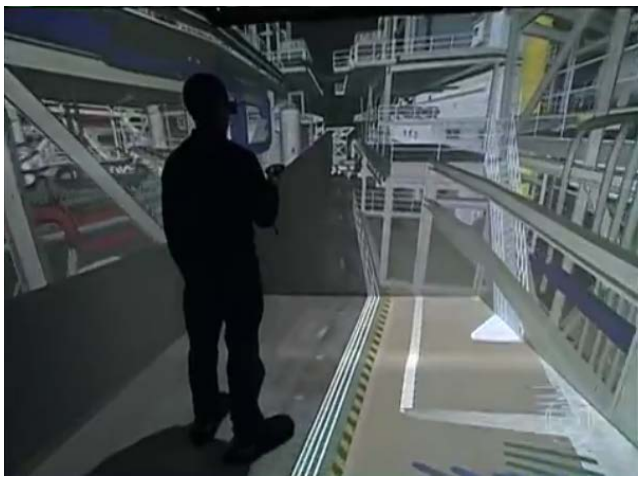

Figure 1: *Environ running on a CAVE.*

Interface resources and beautifully rendered scenes are important for helping manipulate objects and viewing large amounts of data; however in some cases more flexible and programmable resources are required. So, Environ provides a full script interface using the LUA language [Ierusalimschy 2006]. With this script language, it is possible to execute many complex operations inside Environ programmatically. At the same time, it provides a perfect way to send commands from an external application to Environ.

To achieve the integration needed by our framework very few changes were needed on the application side, since Environ already met the most important prerequisite to start integrating an application, which is having a basic exported command passing interface. In Section 5 we will discuss different ways that an application can export commands.

4 Architecture

The framework AppRC was developed in Java as a Netbeans Platform application using Mule ESB [Mule ESB framework 2013]. It is designed to have a very simple User Interface for basic monitoring and controlling different communication channels. It is also able to run on the background as not to disturb the main application. As can be seen on the basic architecture overview (Figure 2) , we defined 4 different kinds of possible channels for the input message: Basic text file, Socket, WebService and Mobile communication.

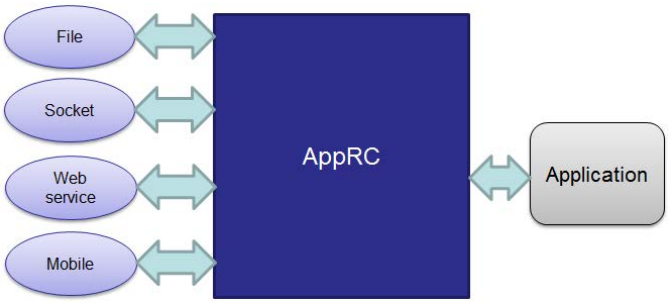

Figure 2: *Basic high level overview.*

4.1 Internal architecture with Mule ESB

The internal architecture of the framework uses the concept of Service Oriented Architectures (SOA) in the form of Mule ESB implementation. We use the benefits of SOA to build a communication bus inside our framework so that the input services (the 4 input channels) can exchange message with the base application service. This section describes the concept of an EAI (Enterprise Application Integration) and the Mule ESB implementation.

EAI is the use of software and computer systems architectural principles to integrate a set of enterprise computer applications. The idea behind EAI is to provide an integration framework where multiple applications and systems can cooperate across the enterprise. The problem is that each sub-system or application inside the enterprise could be running on different operating systems, using a different database or be built on different computer languages. EAI provides 3 main benefits: data integration, vendor independence and a common facade (when working as an front-end for a cluster of applications)

There are many projects and implementations of EAI, like Mule ESB [Mule ESB framework 2013], Apache Camel [Apache Camel Enterprise application Integration 2013] or Spring Integration [Spring Integration 2013]. For our framework we used Mule ESB because it is an open-source lightweight enterprise service bus (ESB) implementation.

4.2 AppRC

Using Mule ESB implementation, we defined two Mule flows. One to handle mobile devices incoming messages (called mobile flow), and another to handle all other incoming messages (called app flow) — Figure 3.

So, with Mule ESB, we were able to easily implement the four command input channels. On top of the four input channels, an adapter

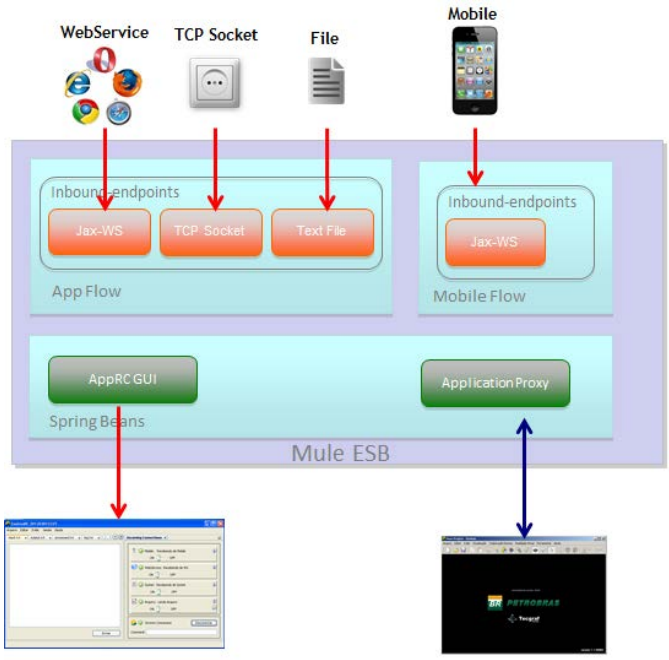

Figure 3: *AppRC with Mule ESB.*

had to be implemented to handle the communication between the ESB and the application (Application Proxy).

Another important aspect of the framework is its support for managing multiple instances of the application running at the same time. This functionality is very important when running an application on very large displays for example, as it allows the user to manipulate from a single source, multiple instances of the application, providing an efficient way to compare simulation results or data. To achieve this goal, we adopted the concept of Sessions. Each session can run a single instance of the application and the user can create and manage multiple sessions.

5 Extension Points

We aimed to make the framework as generic as possible as to enable its use on many different kinds of applications. To achieve this goal we had to create 2 simple extension points (Figure 4) that the user of the framework has to implement to start integrating his application with a mobile device. The first extension point is actually optional and is called Mobile Application Service and it is used to create (if needed) a layer of abstraction for the mobile commands; the second one is called Application Proxy and it handles the communication between the framework and the user's application.

Extension points are actually Java classes from the framework that need to be extended in order to provide specific functionalities of a given application. So, the only work that needs to be done to integrate our framework to an application is actually to implement two (or possibly only one since one of them is not required) simple Java classes.

6 EnvironRC

EnvironRC is the first use of the AppRC framework to integrate mobile functionalities to an existing Virtual Reality application. We added many functionalities to the mobile App to enhance the VR

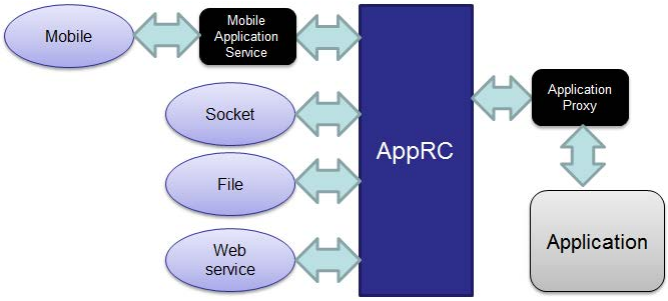

Figure 4: *Extension points that should be implemented.*

experience by making it easier for the user to operate a complex application like Environ without a mouse or keyboard. Furthermore, we implemented ways to help the user navigate the 3D world using the multi-touch display of the Mobile device as well as the build-in accelerometer sensors.

Environ has capabilities to create 3D annotations. An annotation is any textual information that users want to add to their projects to enrich the content or just for documentation purposes. We used AppRC framework to enable the mobile user to create, delete and edit 3D annotations.

Finally, we also implemented a screenshot capture feature to enable remote users (outside of the VR environment) to check different images of the main simulation (Figure 5). In this section we discuss how we display and use the information of the application on the mobile device. The mobile App was developed for IOS devices with support for both IPad and IPhone.

Figure 5: *Environ taking a screenshot of the simulation and sending to a mobile device.*

6.1 Extension points implementation

For EnvironRC we implemented both extension points. In this section we discuss how we used the extension points to suit the needs of a Virtual Reality application.

6.1.1 Mobile Environ Service

To help on the development process of the mobile App for Environ, we created a custom Mobile Application Service to handle high level commands coming from the mobile device. The mobile

service is also able to do some logic depending on the command. A good example of the benefits of using this extension point is the handling of active selection on Environ. We display on the mobile device interface a list of selected objects. With many users using the system at once (many mobile devices at the same time for instance) it can be a problem to maintain the current selected objects list updated on all devices. So, we implemented as part of the Extension Point, a layer of business logic that is in charge of keeping the selection consistent across all connections. When there is a request from a Mobile Device to get the selected elements, the custom mobile service will not pass the command direct to the application, instead it process the command internally and responds accordingly.

6.1.2 Environ Proxy

The implementation of communication between the AppRC and Environ was implemented using a simple local socket connection. The Environ side of the communication was already built to be used as part of the communication channel of the Collaborative Engineering Enviroment CEE [Santos et al. 2008], and we could use it without changing a single line of code on Environ application itself. We only had to implement the framework Extension Point to send the LUA commands to Environ using the same socket structure.

7 Conclusion and Future Work

The AppRC is part of an initiative to streamline the creation and integration of mobile functionalities to complex applications like simulators and 3D immersive environments. In this context, AppRC was designed to be very lightweight, easy to use, extensible, as well as being flexible enough to accommodate the different kinds of needs of any kind of application. AppRC requires very little development work in order to achieve the integration. By implementing only two basic classes, and with minimum change to the base application, it is possible for any application to easily export its functionalities, not only to a mobile device, but to any other application using one of the four available input channels.

AppRC is already in use by PETROBRAS at CENPES as a way to enhance the experience of Environ on many different kinds of virtual reality setups like a CAVE and an L-shape display. AppRC is also in use on an Ultra-HD display (20x HD) that, because of the very high resolution, enables the use of multiple instance of Environ on the same display with many users at the same time. AppRC enables the user of Environ to navigate the 3D world, review engineering data, create 3D annotations or even take snapshot pictures of the simulations from a mobile device.

One problem we would like to address in the future is to also streamline the Mobile App development process. As it is right now, the user of our framework needs to develop the mobile App from scratch. Moreover, if the user wants the mobile App on different platforms (IOS and Android, for example), the work would be doubled. We are beginning to study ways to define a protocol to automatically extract the commands from an application and programmatically build a Mobile app interface based on HTML5 for the application that will work on any mobile platform.

Acknowledgements

Tecgraf is a laboratory mainly supported by Petrobras. Alberto Raposo thanks also CNPq for the support granted to this research (470009/2011-0).

References

APACHE CAMEL ENTERPRISE APPLICATION INTEGRATION, 2013. http://camel.apache.org/.

GUIMARÃES, M. P., GNECCO, B. B., AND ZUFFO, M. K. 2004. Graphical interaction devices for distributed virtual reality systems. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, ACM, New York, NY, USA, VRCAI '04, 363–367.

IERUSALIMSCHY, R. 2006. *Programming in Lua, Second Edition*. Lua.Org.

MEDEIROS, D., CARVALHO, F., RAPOSO, A., AND SANTOS, I. H. 2013. An interaction tool for immersive environments using mobile devices. In *XV Symposium on Virtual and Augmented Reality*, SVR '13, 90–96.

MULE ESB FRAMEWORK, 2013. http://www.mulesoft.org/documentation/display/current/Home.

NORONHA, H., CAMPOS, P., JORGE, J., ARAÚJO, B., SOARES, L., AND RAPOSO, A. 2012. Designing a mobile collaborative system for navigating and reviewing oil industry CAD models. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense through Design - Industrial Experiences*, NordChi 2012.

RAPOSO, A., SANTOS, I., SOARES, L., WAGNER, G., CORSEUIL, E., AND GATTASS, M. 2009. Environ: Integrating VR and CAD in engineering projects. *Computer Graphics and Applications, IEEE 29*, 6, 91–95.

SANTOS, P., RAPOSO, A., AND GATTASS, M. 2008. A software architecture for an engineering collaborative problem solving environment. In *Software Engineering Workshop, 2008. SEW '08. 32nd Annual IEEE*, 43–51.

SPRING INTEGRATION, 2013. http://www.springsource.org/spring-integration.

TAYLOR, II, R. M., HUDSON, T. C., SEEGER, A., WEBER, H., JULIANO, J., AND HELSER, A. T. 2001. VRPN: a device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, New York, NY, USA, VRST '01, 55–61.

TEIXEIRA, L., TRINDADE, D., LOAIZA, M., CARVALHO, F. G. D., RAPOSO, A., AND SANTOS, I. 2012. A VR framework for desktop applications. In *Proceedings of the 2012 14th Symposium on Virtual and Augmented Reality*, IEEE Computer Society, Washington, DC, USA, SVR '12, 10–17.