A Web Player for Ogre3D

Maylson Gonçalves Leonardo de Aguiar Werson Araújo Jairo Oliveira

Waveit Technology, Belém, Brazil

Abstract

This paper presents the architecture of an open-source framework, created for the development of browser-based applications that use Ogre3D as rendering engine. Currently, many games and 3D interactive applications have been distributed on the Internet to run via a browser through plugins like Unity Web Player. This work allows Ogre3D-based applications to run in the browser without requiring the user to install any Ogre3D exclusive plugin, using Java plugins through Java Native Interface to communicate directly with the native code of Ogre3D. At the end, two case studies developed with the framework are shown.

Keywords: Web Player, Web Games, Ogre3D

Authors' contact:
`{maylson,leonardo}@waveit.com.br`
`{werson,jairo}@waveit.com.br`

1. Introduction

Currently, many games and 3D applications are distributed on the Internet as webgames or browser-based games. Several game engines and authoring tools have provided web players to execute their content in various browsers. We can highlight the 3D applications and games developed with Unity3D, and deployed on the web through Unity Web Player [Unity 2011]. Flash games are another example of browser-based games [Flash 2011].

The Ogre3D graphics engine has been widely used in the development of electronic games, virtual reality applications and computer graphics in general [Ogre 2008], but it lacks any kind of tool that enables these applications to run in the browser. Thus, it is important to develop tools that enable web deployment of Ogre3D-based applications.

In this context, this paper presents an open-source framework for developing browser-based applications that use Ogre3D as rendering engine. This proposed framework uses Java Plugin technology for the deployment of the application on the web, taking advantage of Java Native Interface (JNI), which allows a Java application to access native code of another application. The great advantage of using Java Plugins is the fact that the Java Runtime Environment (JRE) has great penetration in today's computers [Adobe 2011].

This paper is divided as follows: Section II discusses the related work. Section III presents a brief summary of tools used in developing the proposed framework. Section IV shows the definitions of the proposed architecture. Section V illustrates how the application is deployed on the web. Section VI shows some case studies and, finally, conclusions and perspectives for future work are made.

2. Related Work

Regarding Ogre3D for web, this work divided them in two most relevant categories: "own plugins" and "Java-based plugins." Most solutions are proprietary code, not allowing other developers to use them. For the item "own plugins" the work developed by NeoAxis Group to the NeoAxis Engine, an Ogre3D-based engine, can be exemplified. NeoAxis based applications can be deployed to web browsers by means of additional closed-source plugin called NeoAxis Web Player. This web player supports all major web browsers including FireFox, Internet Explorer, Google Chrome, Opera and Safari. It is necessary to install the plugin and restart the browser to use the Web Player [NeoAxis 2011].

A similar web player was developed by I-Maginer for the project OpenSpace3D. Both [OpenSpace3D 2011] and [NeoAxis 2011] require that the application is developed in their respective tools, which makes them impractical for applications developed directly with the Ogre3D.

For the item "Java-based plugins", we can highlight the strategy online game Dynastica, released by Straycat Studios. According to reports of the developer in [Dynastica 2010], the solution adopted by Straycat Studios is the closest to the solution proposed in this paper, however, it is closed-source.

The framework presented in this paper differs from related work because it is open- source and uses Java, a widely known plugin and with a large user base, instead of using an own plugin. Furthermore, this work allows that any application based on Ogre3D can be ported to the Web Player, not being restricted to third-party tools.

3. Tools

A. Ogre3D

The Ogre3D (Object-Oriented Graphics Rendering Engine) is a 3D rendering engine developed in C++, object-oriented, cross-platform and open source, widely used in the development of electronic games. The graphics capabilities of Ogre3D include support for OpenGL, OpenGL ES and DirectX [Junker 2006].

B. Java Plugins

Java Plug-in technology, included as part of the Java Runtime Environment Standard Edition, establishes the connection between the browser and the Java platform. This connection allows Java Applets to run on the desktop, with or without the browser. [Oracle 2007]

C. Java Native Interfaces

JNI is a powerful framework developed by Sun that dipped to meet the need of re-use of native code written in other programming languages (C, C + +), allowing them to be invoked by Java Virtual Machine [Liang 1999].

In order to circumvent the restrictions imposed by Java Virtual Machine, one can write classes in Java with JNI methods that access these libraries from other languages and native applications [Liang 1999].

Initially, the JNI design took into account only C/C++ programming languages, but researchers interested in this tool facilitated communication with other programming languages such as Assembler and Delphi, for example [JNI 2005].

Figure 1 represents the layers of communication applications with Java libraries, JNI and functions from other programming languages.

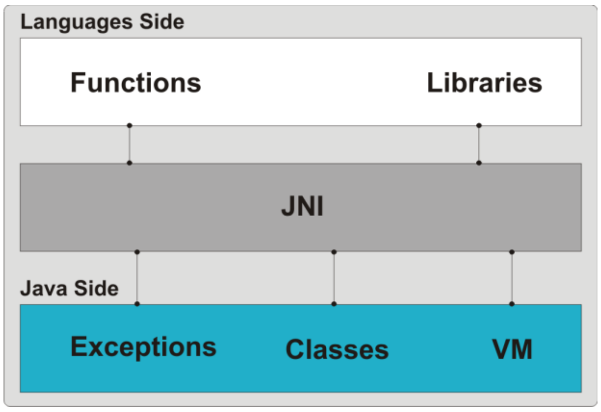

Figure 1: JNI communication layers

4. Framework Architecture

The framework's architecture aims at simplicity and modularity, and its understanding is quite simple. It is very familiar to the Ogre users, because it is based on the architecture of the Ogre3D sample applications. Each module of the framework has a specific and well defined function that helps the programmer to develop the application quickly.

Figure 2 shows the architecture of the framework.

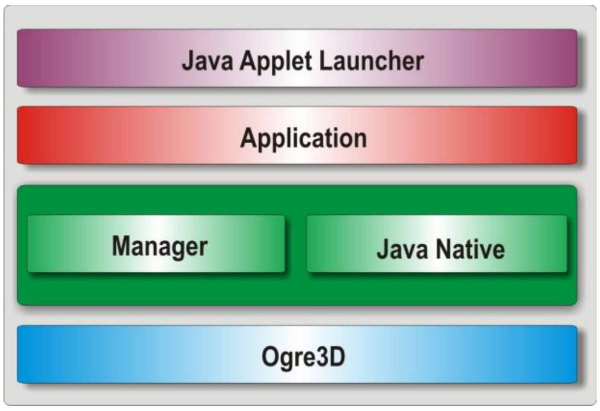

Figure 2: Framework Architecture

The modules of the framework are:

A. Manager

Manager is one of the most important parts of the framework, although it should be hidden from the developer. This module is responsible for managing the entire life cycle of the application, from its startup to the control of the game loop. Also, it sends to the Application class the received input from mouse and keyboard through Java.

B. Application

This class is the interface between the Manager and the user application. The Application class has the basic methods for creating and controlling a scene, such as loading of media files, game loop and the controls of mouse and keyboard, camera control, among others. The developer can also add his own methods.

C. Java Native

Java Native module is responsible for implementing the interface of communication with Java, exposing some methods of Manager, allowing the application to be initialized, updated on each frame and closed through Java. In addition, this module also implements some methods responsible for injecting input received from Java to the Manager.

D. Java Applet Launcher

Once the application has been developed, the code generated from application is accessed by the Launcher, which is a simple Java Applet responsible for sending the input commands to the framework and receive the rendering done by this one on each frame.

5. Deploying the Web Application

Deploy the application on the Internet becomes quite simple. It is need to put the JAR packages on the web server, digitally signed, set some parameters in JNLP file and put the small JavaScript code on the website, responsible for loading the Java Applet [Oracle n.d.]. This entire process is described below.

A. *JAR packages*

Java Plugin technology requires that all needed files to run the application are packaged in JAR format. Thus, it is necessary to compress all media of application and all required binaries.

For example, if a particular application uses the files "Models.zip" and "Textures.zip" as a source of media, just put them in the same folder, open a terminal and run the command:

```
jar.exe cfv media.jar Models.zip Textures.zip
```

This command generates the "media.jar" file, which contains all the media necessary for correct execution of application. Similarly, to generate the JAR file that will contain the binaries of application, it is necessary to put the binary of Launcher (Launcher.class) and the DLL generated by the framework (framework.dll) in the same folder. The following command generates the “bin.jar” file.

```
jar.exe cfv bin.jar framework.dll Launcher.class
```

The next step is to sign the JAR files.

B. *Digital signature of JAR files*

Because using JNI, the Launcher loses secure and managed environment of Java and break, at first, the portability of the application, since the native code can only be executed on the platform for which it was compiled. Thus, in order to impose greater security for Applets, it is need to digitally sign the JAR files, so that the user will be alerted to the execution of the application and will be asked if desires to run it in his browser.

It is possible to create a digital signature to the JAR files using tools provided by Java itself, like *keytool* and *jarsigner*.

C. *JNLP file*

Applications developed using Java Plugin are initialized using the Java Network Launch Protocol (JNLP), which consists in simple XML text file containing instructions for Java to download the required JAR files and launch the application, besides passing other useful parameters to the Launcher, like the dimensions of the Applet, for example.

D. *Website*

To publish the application on a website, it is necessary to place the JAR files on the server and load the Applet using JavaScript, like the example below.

```
<script
   src="http://www.java.com/js/deployJava.js">
</script>

<script>
   var attributes = {code: 'Launcher.class',
                     width: 1024,
                     height: 480,
                     archive: 'bin.jar'};

   var params = {jnlp_href: aplicativo.jnlp',
                 separate_jvm: 'true'} ;

   deployJava.runApplet(attributes, params, '1.6');
</script>
```

6. Case Studies

In order to evaluate the potential of this framework, some 3D applications were developed and made available on the web. The developed examples include techniques developed as skeleton animation, hardware skinning and character controller, based on examples of Ogre3D.

When testing these applications, they proved to be very efficient, running flawlessly on Windows XP, Vista and Seven, in browsers Internet Explorer, Firefox, Opera, Safari and Chrome.

Figure 3 shows a demo of Ogre3D running in the browser. This demo uses Skeletal Animation and Hardware Skinning.

This demo can be accessed at: www.waveit.com.br/webplayer.

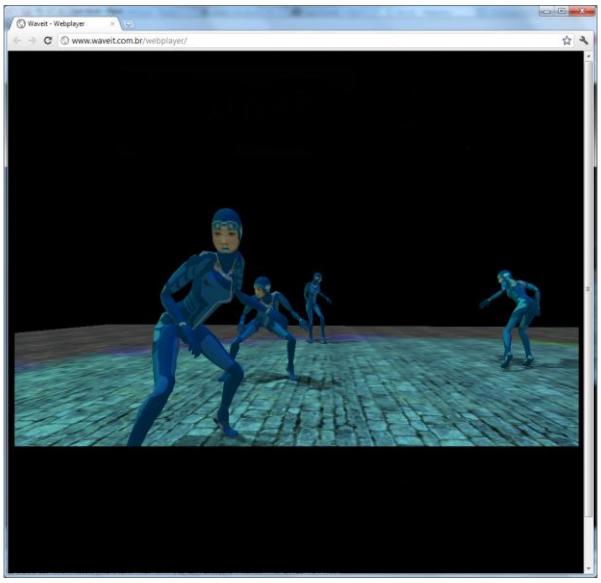

Figure 3. Skeletal Animations e Hardware Skinning demo

Figure 4 shows another Ogre3D demo running in the browser. In this application, the user can control the avatar using the arrow keys to move, space to jump, and the left mouse button to look around. Additionally, some sound effects were included using the OgreOggSound library [OgreOggSound 2011].

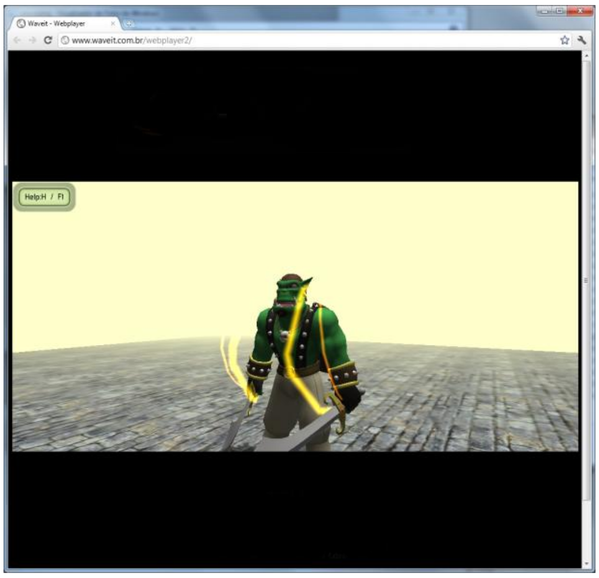

Figure 4. Character Controller demo

This demo is also available at: www.waveit.com.br/webplayer2.

3. Conclusion

The applications developed for the case studies were very efficient without loss of performance or problems that prevent their execution. The tests were performed on a Pentium IV 3.0 GHz with 1 GB of RAM and a generic onboard video card. We found some problems with video cards with generic drivers, which prevents the proper execution of the application in the browser.

This work allows that 3D applications like games and simulators, for example, are deployed on the Internet in a simple way and can use all the Ogre3D graphics capability, like access to the GPU, shaders, Skeletal Animations, Scene Graph etc.

As future work, we intend to use the Ogre3D plugins structure, facilitating the integration of Web Player with future and ongoing Ogre3D-based projects. Another objective is to expand the framework by adding support for streaming of media files. In current version, applications developed with the framework still need to download all JAR packages that contain the media files and binaries, before starting the application. Furthermore, the launcher needs to be improved to be completely independent of the application that's being developed, so that the necessary attributes to be passed as parameters in JNLP file, avoiding the need to change them in code. Also, the framework needs to be ported to other platforms such as Linux and Mac, because the use of Java with JNI depends on the target platform.

The framework implementation is public and can be found in: https://github.com/maylson/Waveit-Web-Player.

References

UNITY. *Unity3D*. [Online]. Available from: http://www.unity3d.com/ [Accessed 25 July 2011].

FLASH. *Adobe Flash Player*. [Online]. Available from: http://www.adobe.com/products/flashplayer/ [Accessed 25 July 2011].

OGRE. *OGRE User Survey 2008 Results*. [Online]. Available from: http://www.ogre3d.org/downloads/OGREUserSurvey2008_Results.pdf [Accessed 25 July 2011].

ADOBE. *Flash Player Penetration*. [Online]. Available from: http://www.adobe.com/products/player_census/flashplayer [Accessed 25 July 2011].

NEOAXIS. *NeoAxis Engine*. [Online]. Available from: http://www.neoaxis.com/ [Accessed 25 July 2011].

OPENSPACE3D. *OpenSpace3D*. [Online]. Available from: http://www.openspace3d.com/ [Accessed 25 July 2011].

DYNASTICA. *Dynastica*. [Online]. Available from: http://www.dynastica.com/ [Accessed 25 July 2011].

DYNASTICA. *Dynastica – an online strategy game*. [Online]. Available from: http://www.ogre3d.org/forums/viewtopic.php?f=11&t=60857 [Accessed 25 July 2011].

JUNKER, G., 2006. Pro Ogre 3D Programming. Apress.

ORACLE. *Java Plugin Technology*. [Online]. Available from: http://www.oracle.com/technetwork/java/index-jsp-141438.html [Accessed 25 July 2011].

LIANG, S., 1999. Java Native Interface: Programmer's Guide and Specification. Prentice Hall.

SEMAAN, G., *Interação Java e outras linguagens*. [Online]. Available from: http://www.linhadecodigo.com.br/Artigo.aspx?id=747 [Accessed 25 July 2011].

ORACLE. *Deploying a Java Web Start Application*. [Online]. Available from: http://download.oracle.com/javase/tutorial/deployment/webstart/deploying.html [Accessed 25 July 2011].

OGREOGGSOUND. *OgreOggSound Project*. [Online]. Available from: http://sourceforge.net/projects/ogreoggsound/ [Accessed 25 July 2011].