INTERFACES CIENTÍFICAS

EXATAS E TECNOLÓGICAS

ISSN IMPRESSO - 2359-4934
ISSN ELETRÔNICO - 2359-4942
http://dx.doi.org/10.17564/2359-4942.2018v3n2

REVISING FRAMEWORKS FOR DEVELOPING MOBILE VIRTUAL REALITY

REVISANDO FRAMEWORKS PARA DESENVOLVIMENTO DE REALIDADE VIRTUAL MÓVEL

REVISIÓN DE LOS ENTORNOS PARA EL DESARROLLO DE REALIDAD VIRTUAL MÓVIL

Guillermo Horacio Rodriguez[1] Fabio Gomes Rocha[2]

ABSTRACT

The development of mobile virtual environments has been enabled by recent advances in hardware and software for mobile computing. This new trend has resulted from the convergence of wear able computing, wireless networking and mobile virtual reality interfaces. This work provides a survey of different mobile technologies that are useful to build virtual reality applications running through mobile devices. Our aim is to place those technologies into different categories so that it becomes easier to understand the state of art and to help identify new directions of research. A comparison of attributes of each technology is also summarized.

KEYWORDS

Framework. Developing Mobile. Virtual Reality.

RESUMO

O desenvolvimento de ambientes virtuais móveis foi possibilitado pelos recentes avanços em hardware e software para computação móvel. Essa nova tendência resultou da convergência de interfaces de computação portátil, redes sem fio e realidade virtual móvel. Este trabalho fornece uma pesquisa de diferentes tecnologias móveis úteis para criar aplicativos de realidade virtual executados em dispositivos móveis. Nosso objetivo é colocar essas tecnologias em diferentes categorias para que seja mais fácil entender o estado da arte e ajudar a identificar novos rumos da pesquisa. Uma comparação de atributos de cada tecnologia também é resumida.

PALAVRA-CHAVE

Framework. Realidade Virtual. Desenvolvimento Mobile.

RESUMEN

El desarrollo de entornos virtuales móviles ha sido habilitado por los recientes avances en **hardware** y **software** para computación móvil. Esta nueva tendencia se debe a la convergencia de la computación portátil, las redes inalámbricas y las interfaces de realidad virtual móvil. Este trabajo proporciona una investigación de diferentes tecnologías móviles que son útiles para construir aplicaciones de realidad virtual que se ejecutan a través de dispositivos móviles. Nuestro objetivo es colocar esas tecnologías en diferentes categorías para que sea más fácil comprender el estado del arte y ayudar a identificar nuevas direcciones de investigación. También se resume una comparación de los atributos de cada tecnología.

PALABRA CLAVE

Framework, Realidad Virtual, Desarrollo móvil

1 INTRODUCTION

Today's mobile phones are being used as computers for multi-purpose functions. Not only the number of mobile Internet users is growing rapidly, but also the mobile market is growing (MILOSEVIC; DEHGHANTANHA; KIM-KWANG, 2017). However, at the same time the market of the mobile platforms is fragmented, there are many mobile operating systems and programming languages. Thus, mobile developers need to use platform specific tools and APIs in order to write mobile applications in different programming languages on different platforms (ALLEN; GRAUSPERA; LUNDIRGAN, 2010).

Reprogramming basically the same application into multiple mobile platforms means an increase in development costs and longer development times or decrease in the number of supported mobile platforms and a transfer of focus only on specific devices. The demand for shorter mobile application development process has driven the need for cross-platform solutions. The idea of cross-platform mobile programming is that with the same base-code with little or no modifications a mobile application could be published into multiple mobile phone platforms.

Mobile devices have become an integral part of everyday lives. Particularly, virtual reality, like 3D games, has been highly used in education and entertainment because everybody is able to access to mobile devices. A new approach that is currently taking place in education is the use of virtual worlds and virtual-reality-based games. These applications allow teachers to load courses and interact with students through the virtual world. This approach is known as m-Learning. For example, some approaches have illustrated the suitability of the framework and authoring tool for supporting users without programming skills in developing their own apps (MOTA *et al.*, 2018). In this light, the aim of this work is to research the state of the art in cross-platforms and their development environments.

The remainder of the paper is organized as follows. Section 2 gives an overview of different mobile cross-platforms. Section 3 describes frameworks for developing mobile virtual reality. Section 4 described 3D game development tools which integrate with mobile platforms. Section 5 compares different frameworks analyzed. Finally, Section 6 concludes this research and identifies future lines of work.

2 VIRTUAL REALITY ON MOBILE PLATFORM

The emergence of the Mobile Computing (MC) paradigm has been led by the evolutions in portable computer devices technology, and the advances in wireless communication (DIAS JUNIOR, 2010). Constant research in this area has permitted the development of a great variety of portable devices that allow users to work together regardless their physical location, leading to greater freedom among users and computational systems.

Different types of equipment such as mobile cell phones with Wireless Application Protocol (WAP), notebooks, palmtops, among others, have resulted in an increased demand for new services and applications in the most diverse areas. The evolution of MC has promoted advances in the development of new hardware and software technologies, opening a range of new applications based on wireless communication resources. Among those, applications for Virtual Reality (VR) and/or Mixed Reality (MR) in the most diverse areas such as education, entertainment, visualization etc., are in special evidence (PAPAGIANNAKIS; SINGH; MAGNENAT-THALMANN, 2009). The portability provided by mobile devices allows the creation of a Mobile Virtual Environment to exchange experiences, information, and images, among people, in a more attractive and motivating way for the user.

However, there are some problems specifically related to mobile devices limitations, which continue to represent a `bottleneck' for the creation of VR mobile applications. The main obstacle to be transposed in the development of applications for mobile devices is their heterogeneity, characterized by limitations in terms of processing, memory,

and battery capacity, communication bandwidth, as well as the great variety of existing software (Windows, Symbian, Palm OS etc.). Due to those different features, it would be necessary to develop specific VR content versions for these diverse environments, in accordance with the characteristics of each device.

However, the creation of different versions might not be viable and might not reach the human condition, because of the number of platforms and devices available. In this context, the development of VR applications for mobile devices becomes a challenge, especially for heterogeneous devices. Subsection 3.10 shows an approach to tackle the problem of heterogeneous platforms and frameworks.

3 FRAMEWORKS TO DEVELOP MOBILE VIRTUAL REALITY

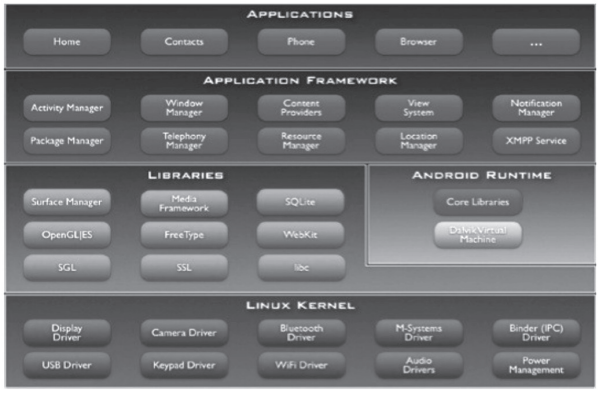
Particularly, this work focuses on analyzing cross-platform native frameworks. A cross-platform native framework is software that allows a common development approach across platforms but that build to an application that is indistinguishable by a user from one built with native code (ALLEN; GRAUSPERA; LUNDIRGAN, 2010). Cross-platform native frameworks let a developer to create a native, installable and mobile application distribution platform compatible with multiple platforms using cross-platform APIs and usually HTML/CSS/JavaScript frameworks.

3.1 ANDROID

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel. Its API (Application Programming Interface) uses the Java language and consists of a set of Java libraries written by Google (CHALANT, 2010). Android applications are organized by activities. An activity is like a window or screen, and contains layouts which contain widgets such as buttons, or labels. Widgets are created when the activity starts, and have their own event listeners. An activity can start another, but communication between activities is usually done by passing parameters.

Only one activity can have focus at a time and activities that are idle for a long time are killed by the Activity Manager, therefore Android applications never exit upon request from the user, they are just put in background and killed if they remain inactive for some time. Figure 1 describes the components of Android architecture based on a Layer design pattern.

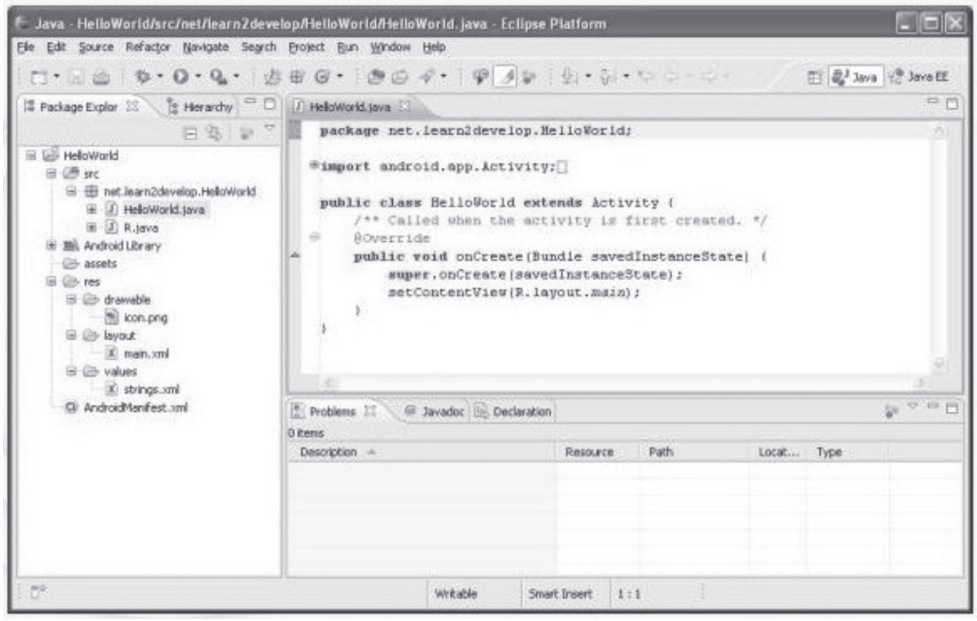
Figure 1 – Android components



Fonte: MOTA, 2018

The emulator is essentially meant to test your Android applications. It resides under the `tools/' directory. The emulator also ships with a plethora of applications like a browser, a cooler-than-traditional phonebook and a map application, among other features. So, even if users have not yet written an Android application, the emulator is a must checkout. Figure 2 shows an example built in the android SDK, a plugin of Eclipse IDE.

Figure 2 – Android Eclipse



Fonte: Elaborado pelos autores

Figure 3 – Android Manifesto.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.HelloWorld">

    <application android:icon="@drawable/icon"
        android:label="@string/app_name">

        <activity android:name=".HelloWorld"
            android:label="@string/app_name">

            <intent-filter>
                <action
                    android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>

</manifest>
```

Fonte: Elaborado pelos autores

Figure 3 shows an example of AndroidManifest. xml. Every application must have an AndroidManifest.xml le (with precisely that name) in its root directory. The manifest presents essential information about the application to the Android system, information that system must have before it can run any of the application's code.

3.2 I-PHONE

The iPhone is a line of smart-phones designed by Apple and released in 2007. The user interface is entirely based on the multi-touch screen and uses the built-in sensors. The iPhone API is called Cocoa touch and it inherits from Cocoa, the Mac OS X API. It is written entirely in Objective-C, a reflective, object-oriented programming language which adds Smalltalk-style messaging to the C programming language. Cocoa's design is a strict application of Model-View-Controller principles: all the UI widgets are views which are themselves contained in other views. Each iPhone application typically has one single window which contains several sub-views. These views can be brought to front whenever it is necessary (CHALANT, 2010).

Xcode is the complete toolset for building Mac OS X and iOS applications. With Xcode 4, the tools have been redesigned to be faster, easier to use, and more helpful than ever before. The Xcode IDE understands a project's every detail, identifies mistakes in both syntax and logic, and will even fix code. Quite simply, Xcode 4 will help users write better code. Xcode 4 has a brand new user interface; built upon proven technologies that Apple itself uses to build Mac OS X and iOS, and that have produced over a quarter million Mac OS X and iOS apps.

3.3 MOSYNC

MoSync is a SDK that allows developing applications for all major mobile platforms using a single environment and C/C++ code base on Windows or OS X. MoSync produces real native applications, packaged and ready for distribution in each platform's native installation format. In MoSync API there is support for most of the device features such as location and contacts. The framework supports also OpenGL (MOSYNC). The developing is done using standard C/C++ in an Eclipse-based environment pre-configured with a set of MoSync-specific plugins. Since MoSync uses standard C/C++, many libraries are readily available to developers. Examples of libraries used with MoSync are SDL, yajl, STLPort and SQLite. MoSync supports the iOS, Android, Windows Mobile, Symbian, JavaME and Moblin platforms. MoSync is dual licensed: open source GPL v2 for open source projects and a commercial license for closed source applications.

MoSync is an ideal cross-platform framework for enterprise and customer applications. With MoSync, users develop using standard C/C++ in an Eclipse-based environment per-configured with a set of MoSync-specific plugins to create a fluid experience from development and debugging to testing and deployment. The same compiler, GCC 4.0 with a custom MoSync backend, is used for all platforms, ensuring consistent application behavior everywhere.

3.4 PHONEGAP

Phonegap is an open source framework for building native mobile applications using HTML 5, CSS and JavaScript. Nitobi Software introduced Phonegap in 2008, and is free to use under an MIT license. With Phonegap it is possible to develop applications for iPhone, Android, BlackBerry, webOS and Symbian WRT (ALLEN; GRAUSPERA; LUNDIRGAN, 2010). Phonegap is at its best for transforming a mobile web application into native application. The principle of Phonegap is that it provides a client-side JavaScript APIs with a method for hosting a web application within a native mobile application that an end user may install. Basically a Phonegap application is native application with a full-screen browser.

Developing with Phonegap starts by writing a mobile web application using HTML, CSS and JavaScript. The content of the application does not have to be in any particular structure, developer has much choice how to form the mobile web application layout and structure.

Similar to any other cross-platform frameworks that use the browser for UI, Phonegap is not well suited for applications that require intense math calculations, 3D animations or data-driven applications, like most enterprise applications that must work o offline and synchronize local data. There is no support for database on Phonegap, instead it relies on HTML5 database APIs that is unavailable for some devices.

In order to initialize the project development, the configuration showed by Figure 4 is also required. Figure 5 shows an example of a functionality named get contacts coded in JavaScript.

Figure 4 – Phonegap init

```
<meta name="viewport" content="initial-scale=1.0, user-scalable=no,
width=device-width" />
```

Fonte: Elaborado pelos autores

Figure 5 – Get Contacts

```
function getContacts(selectedstring) {
        navigator.service.contacts.find(['photos', 'name',
        'addresses.streetAddress', 'addresses.postalCode',
        'addresses.locality', 'phoneNumbers', 'emails'],
             successContactFindCallback,
             generalErrorCB,
            {filter: selectedstring}
        );
}
```

Fonte: Elaborado pelos autores

3.5 TITANIUM MOBILE

Titanium Mobile is a commercially supported and open-source framework for creating native cross--platform applications using web technologies. Appcelerator Inc. introduced the framework in December 2008. Titanium Mobile SDK provides the necessary tools, compilers and APIs for building for the target platform, and a visual environment tool called Titanium Developer for creating, running and packaging Titanium applications. However, the Titanium Platform does not contain an IDE or a text editor, but this is subject to change as Appcelerator acquired IDE provider Aptana on January 2011. The new Appcelerator solution will bring features like debugging, code completion, integrated document, and editing tools.

The Titanium SDK allows creating, running and packaging native mobile applications for iOS, Android and BlackBerry (beta) devices using cross--platform JavaScript API. The applications are run against a standalone JavaScript engine that invokes native APIs. The Titanium Mobile SDK uses the native platform's SDK to combine the JavaScript source code via a JavaScript interpreter, and static assets of the application into an application binary.

Titanium Developer is a desktop application for creating, running, managing and packaging Titanium Mobile or Desktop application projects. It also keeps Mobile and Desktop SDKs up to date.

3.5.1 GETTING STARTED

In Titanium Developer, choosing the New Project from the top navigation opens a new project screen. The Project type needs to be Mobile for creating iPhone and Android project. In the new project window the application name, application id, project directory, company URL and Titanium SDK version used are also defined. In the next screen more additional information for the project is asked, such as application version control, description of the project, publisher's name and publisher's URL. The Titanium Mobile Project directory structure Programming the Contacts application begins with editing app.js as in all applications made with Titanium (FIGURE 6).

Since the application is meant to be a simple contact picker application that has access to smartphone's contacts, app.js contains only basic UI settings and a window creation that points to external JavaScript le, contacts.js, where the actual application functionality. The example sets to the contact info label text to show the selected person's full name. Some information in contacts can be multi--value, for example a person can have more than one phone number. If all phone numbers of the person would need to be retrieved, looping through all the data could do it as shown in Figure 7. Figure 8 shows the contact manager application in two different platforms: iOS and Android.

Figure 6 – JavaScript code

```
Titanium.UI.setBackgroundColor('#fff');

var win = Titanium.UI.createWindow({
    title:'Contacts',
    exitOnClose: true,
    url:'contacts.js'
});
win.open();
```

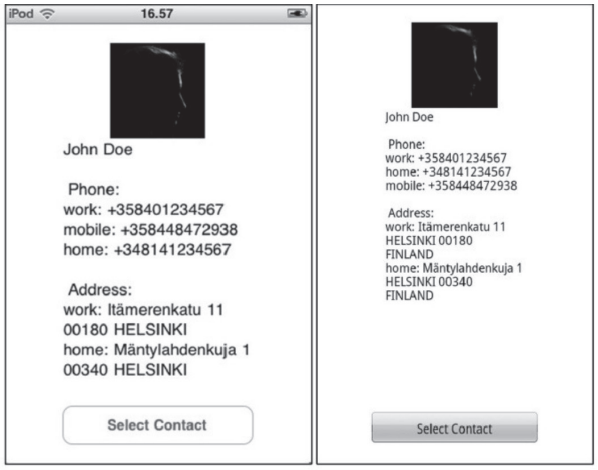
Fonte: Elaborado pelos autores

Figure 7 – Titanium looping

```
for (var label in e.person.phone) {
    var phones = e.person.phone[label];
    for (var i = 0; i < phones.length; i++) {
        contactinfoLabel.text += label+': '+ phones[i] + '\n';
    }
}
```

Fonte: Elaborado pelos autores

Figure 8 – Contacts application running on iOS device (left) and Android (right)



Fonte: Elaborado pelos autores

3.6 RHODES

Rhodes is a commercially supported open source cross-platform smartphone application framework by Rhomobile. It was released in December 2008 (ALLEN; GRAUSPERA; LUNDIRGAN, 2010). Rhodes is available for BlackBerry, Windows Mobile (up to 6.1), and Android and iOS platforms. The Rhodes applications are created using HTML, CSS, JavaScript and Ruby programming languages. The Rhomobile tools and frameworks can be used across Mac, Windows and Linux, and require the target platforms SDKs to be installed. Rhodes is targeted at enterprise applications and is the most suitable for applications that present a series of screens that include standard UI widgets, including common phone UIs. Rhodes leverages Model-View-Controller (MVC) approach in application framework.

Rhodes follows the Model-View-Controller (MVC) pattern. In controller, methods that define actions that map to HTTP requests are implemented. Controller action will fetch data from model that is implemented in the Rhodes ORM Layer, Rhom, and will render a view, which is implemented in HTML ERB. In view the user interface of the application is defined in HTML and CSS. At runtime the HTML and CSS is rendered in a native browser UI control that is embedded in the application by the Rhodes framework.

JavaScript may be used for some interaction control, but using embedded Ruby (ERB) application logic can be added to views. ERB is similar to PHP, in that sense that in the both code can be mixed with markup to create dynamic HTML. Rhodes requires Ruby, Ruby's library packaging system, RubyGems, and GNU installed. When user selects a contact from previously generated contact list, a single contact is viewed. Rhodes Contacts running on iOS.

3.7 ADOBE AIR

Adobe Integrated Runtime (AIR) is a cross-operating system runtime that enables to use ActionScript or HTML/Java-Script development skills and tools to build web applications that run as standalone client applications (CORLAN, 2011). In AIR version 2.6, support for Android, BlackBerry Tablet OS, and iOS mobile operating systems is available. The AIR client runtime is a combination of Flash Player, an embedded SQLite database engine, and the WebKit browser engine. AIR runs on Windows, Mac OS X and Linux. AIR software can be developed with multiple tools, such as Adobe Dreamweaver CS5, Flash Builder 4, Flash Catalyst CS5, Flash Professional CS5, Aptana Studio with AIR plug-in installed or any text editor with AIR SDK.

Development can be done using alternatively ActionScript 3 with Adobe Flash, ActionScript 3 and MXML markup for presentation logic with Flash Builder, or HTML, JavaScript and AJAX with any text editor. For the mobile AIR applications only ActionScript or Flex development is possible, while for desktop it is also possible to use HTML, JavaScript and CSS. Adobe's development tools are commercial products, but AIR SDK is free and includes the tools necessary to build and deploy AIR applications. Adobe Flex SDK is also open source, but Flash Builder is not. Adobe Flex SDK can also be used with an open source IDE, such as Eclipse, instead of Flash Builder.

3.8 OPENPLUG ELIPS STUDIO

Elips Studio is a cross-platform mobile application development toolset by Alcatel- Lucent (former Open Plug) (CORLAN, 2011). The framework was originally introduced in 2009. It allows developing native mobile applications and deploying them on Android, iOS, Windows Mobile and Symbian platforms.

Elips Studio is available as a plug-in for Adobe Flash Builder versions 3 or 4 or as a stand-alone Eclipse based IDE. Mobile applications are developed using ActionScript, the application UI and visual elements using MXML. In addition, the SDK supports web services, network APIs, and device APIs such as GPS, Photos and SMS. The product offers native UI controls mapped directly from each device's OS into code, including lists and screen transitions. It is also possible to manage graphical assets and UI styles through CSS for each variant of an app on each class of device platform and form-factor.

The Elips compiler cross-compiles the application code into C++ code, which allows direct access to native APIs. Finally, an application is packaged into native, installable mobile application (ELIPS, 2011). Elips Studio is not open source. It is free for evaluation only for corporations, for individual developers the framework is free, but ad-supported, which means that an advertisement banner is automatically added to all applications.

A new Elips project can be created in Adobe Flash Builder 4 either by selecting File New Other Elips Project or by clicking the Create a new Elips Project button in Elips toolbar. The Elips targeted devices can be chosen from Project Properties Elips Targeted devices. Multiple devices can be selected to the project, such as iPhone, Generic Android devices with multiple screen resolutions, S60 Devices of different editions, Windows Mobile devices, and Symbian.

3.9 CORONA

Corona SDK is a mobile development framework for creating applications and games for the iPhone, iPad, and Android. The developing language is a scripting language called Lua. With the framework it is possible to create native, installable and distributable applications on the iTunes App Store and Android Market.

The Corona SDK is graphic and game oriented framework, and it offers an integrated game engine and support for the graphics acceleration hardware of the device (OpenGL/ES). The framework is not open source: the subscription for one year is 199 $ / platform or 349 $ / year for iOS and Android. Since the framework is game oriented and a closed source, it is not applicable to the Client's needs. PHP, PERL also embed HTML with their business logic. This embedded HTML is passed to browser for parsing which generates User Interface at client side. With that in mind, the same architecture will provide a solution for cross platform mobile development.

The application page is designed similarly as it is done in HTML but here it would be done in custom defined PhoneXML (NAGESH; CAICEDO, 2012). This XML would be parsed at every mobile platform to generate User Interface of phone application. The argument is based on the fact that almost all the smart phones support internet connectivity and third-party applications. The framework will work in form of an application that will render the PhoneXML returned by the application server. The developer then only needs to learn PhoneXML to develop application for any mobile platform. This is one time task and that same application could be used for different business logic and user interface, which will be provided by server.

The PhoneXML is nothing but simple XML tags which holds the information about the User Interface, as the application is thin client all the business logic and data storage is done at server side. This XML could be static and dynamic at application runtime. Developers (user of the framework) will have to integrate the server side logic with XML being parsed in order to present his/her intended GUI. Platform independency will be achieved after the development of client side application for every Mobile platform.

4 GAME DEVELOPMENT TOOLS

Game development for mobile devices has received important attention since games are required not only for entertainment, but also for educational contexts. Compared to traditional game development, programming mobile games is less complex (KURKOVSKY, 2009), which enables users with limited programming experience to create playable mobile games. In this section two game development tools are described: JMonkey and Unity 3D.

4.1 JME (JMONKEY ENGINE)

It's an interesting time in the gaming world as devices other than PC's and Consoles are gaining the horsepower required to run real 3D games. These powerful little devices are bringing a new aspect to the physicality of gameplay with their many integrated features. With that in mind, a member of the JMonkey Engine team made an announcement in the forums that he was instituting Android support for the third installment of the engine.

The adoption of Android abilities into jME3 means big things for game developers using the engine. The ability to develop for many platforms with expertise in one engine will empower JMonkey Engine users like never before. Games, scientific applications, and just about anything else you can dream up are all coming to an Android phone near you. One of the most portable, and open source, engines around just got more accessible. Android has made a splash in the mobile industry not only as the first operating system to be put forth by Google, but by being the first real competitor to Apple's iPhone OS. The operating system is currently available on a number of phones including the HTC Dream (marketed as the G1), HTC Hero (marketed as the G2), Motorola Droid, and the Nexus One. More phones are forthcoming meaning that Android is likely to keep getting into new hands.

4.2 UNITY 3D

Unity is a proven game development tool that has been designed to let you focus on creating amazing games for iPhone, iPad and Android OS. Unity3D is a powerful graphic engine that has emerged recently as the main software for creating 3G content (KATZ; COOK; SMART, 2011).

The main features of Unity3D are its visual work philosophy with the resources, regardless of complicated interfaces and menu systems, its multiplatform game feature for Mac OS, Windows, and Apple's mobile devices (iPhone / iPhone) and its price as it has a version Indie free, ideal for independent developers and lovers of these applications, and a Pro one for commercial use. On the side of programming, Unity uses MonoDevelop, an open source implementation of .NET framework which supports JavaScript, C# and Boo. These languages are interpreted, powerful with an outstanding performance and easy for programmers. Unity includes a tool to edit scripts named Uniscite based on Scintililla and SciTE. Also, Unity3D allows developers to integrate to Visual Studio.

Other features of Unity3D are:

Management of advanced resources which import automatically resources from project hierarchy;

Implementation of OpenGL for representing graphics;

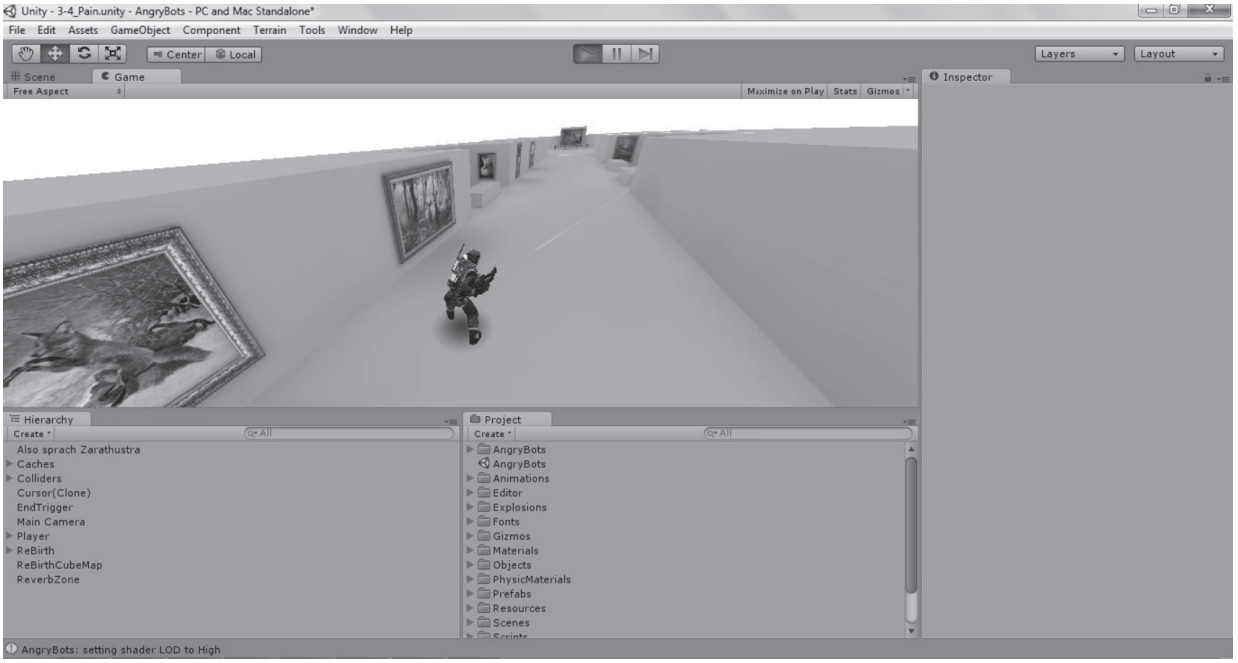
Support for shaders and implementation of ShaderLab to create and configure shaders;

Export to Web content which is rendered by a proprietary plugin;

Implementation of physics engine called PhysX from Nvidia.

Unity3D has an adapted version of Javascript, based on the ECMAScript specification informally called UnityScript25. Formally, within Unity3D, Mono will be responsible for the collection and interpretation of the code written in this language. Thus, intermediate code will be the same as the one used in C# interpreting, i. e. MSIL. Figure 9 shows an example of a Unity scene rendered in the Unity tool kit.

Figure 9 – Unity 3 Dscene



Fonte: Elaborado pelos autores

5 COMPARISON OF FRAMEWORKS

Cross-platform development is a good choice, if a developer has no native programming skills, the schedule is tight and there is not a great amount of financial resources available for the development. Cross-platform is also a good choice if the application uses much web-based data or shares resources with a website, or it is a quite lightweight application that does not require much of the smartphone's hardware resources.

The comparison method was done by tables with features present in almost most frameworks. Figure 10 shows a comparative table in which main features of cross-platform framework are summarized. Figure 11 evaluates and compares two 3D game development tools described in section 4. It aims to serve as a reference point and guide for developers and practitioners in choosing a mobile platform for development on information appliances.

Many virtual worlds are accessed via a rich client interface that must be downloaded and installed into the user's environment. For many users, especially enterprise users, this large download and install represents a significant obstacle to virtual world acceptance (KATZ; COOK; SMART, 2011). Using Unity3D provides the possibility of accessing virtual worlds through a Web browser.

Figure 10 – Comparison of frameworks

	Android	Mosync	PhoneGap	Titanium	Rhodes	Corona	AIR	Open Plug	I-Phone
License	Open Source	Free for private use. Subscription for commercial use.	Open Source: MIT	Open Source: Apache 2.0	Dual License: Open Source: MIT; Closed source for commercial projects	Privative	Closed source, Flex SDK mostly Open Source	Closed source, free for individuals (adsupported)	Free and Open Source Software
Interpreting	Compiled using Eclipse	MoSync IL using GCC	Rendered in WebView control	JavaScript mapped to native code	Runs a Ruby bytecode through bundled RubyVM interpreter	Xcode to compile	Adobe Air Runtime	AS3 compiled to C++, which compiled to standalone native application installer	Cross-compiler runs on Mac OS and buils exe's for iPhone ARM.
Development language	C, C++	C, C++, Java	Java, JavaScript, HTML, CSS	JavaScript, HTML, CSS	Ruby, JavaScript, HTML, CSS	Lua scripting language	ActionScript3 (MXML)	ActionScript3 (MXML)	Objective-C
Native UI Support	yes	yes	3rd party libraries required	yes	yes	no	3rd party libraries required	no	yes
Native Code Support	Yes, using NDK C++	yes	yes	yes	yes	yes	no	yes	Yes, using iPhoeWax
Platform	Windows, Mac, Linux	Java ME, Android, Symbian, iOS	Android, iOS, BlackBerry Phone, Symbian, Palm	Android, iOS	Android, iOS, BlackBerry Phone	Android, iOS	Android	Android, iOS, Symbian, Palm	iOS, Mac, Widnows
SDK	API for Eclipse IDE with emulator	MoSync SDK, MoSync IDE	Eclipse, Android SDK, Ant, Ruby	Titanium Mobile SDK	Rhomobile SDK	Corona SDK	Air SDK	Flex SDK	IPhone Apple SDk
Programmability	Java and Emulator built-in	Easy combination between C++ and Java	Eclipse, environmental variable settings, javascript	IDE with several perspectives. XML, HTML and JavaScript.	Open source SDK. It uses Ruby on Rails.	fast and easy development tool for iPhone, iPad and Android games and applications	AIR SDK with Java JDK. XML, HTML, JavaScript, console commands.	Flex development to build flash apps on mobile devices	Xcode IDE with Ojective-C

Fonte: Elaborado pelos autores

Figure 11 – JME vs. Unity 3D

	Unity 3D	JMonkey
Development Language	C# and JavaScript	Java
License	Unity is free. Unity Pro is privative	Open source
Graphic Engine	Direct3D and OpenGL	Java OpenGL Engine
SDK	Authoring tool to create 3D games	No authoring tool
Browser Rendering	Yes	No
Mobile Framework Integration	I-Phone, Android	Android
Auto-generated Code	yes	No
Platform	Windows, Mac OS	Windows, Mac OS, Linux, Solaris

Fonte: Elaborado pelos autores

6 CONCLUSION

The items discussed in Section 3 and 4 are only a subset of the possible research topics in mobile technologies for virtual reality, but serve to indicate the breadth of research needs and opportunities in this emerging field. The trend towards more virtual reality on mobile devices is accelerating. There is an explosion of new mobile frameworks that allows users to develop complex virtual reality representations. Indeed, current advances in mobility are providing new ways to look at virtual reality helping users to create and develop non-immersive virtual reality.

Particularly, this work focused on the use of mobile virtual reality applications like virtual worlds or other games for education and learning. Thinking about education through mobile devices fails to be a utopian idea since resources are available. As future work, we are planning to conduct substantial research in order to bridge the gap between virtual reality representation and diversity of mobile technologies.

REFERENCES

ALLEN, Sarah; GRAUSPERA, Vidal. Graupera; LUNDIRGAN, Lee. Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution. **Apress**, 2010.

CHALANT, Rue. **Abstraction layer tool for designing user interfaces on mobile devices**. PhD thesis, Ecole Nationale Superieure de l'Electronique et de ses Applications, Cergy-Pontoise (Paris), France, 2010.

CORLAN, Mihai. Developing for mobile devices with the adobe flash platform. Technical report. **Adobe Developer Connection**, 2011.

DIAS JÚNIOR, José Barbosa *et al*. Software architecture for adapting virtual reality content to mobile devices. In Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology, CIT '10, pages 2039 2045, Washington, DC, USA, 2010. **IEEE Computer Society**. 2010.

ELIPS Studio. Elips studio: cross-platform native mobile application development. Technical report. **OpenPlug Developer**, 2011.

KATZ, Neil; COOK, Thomas; SMART, Robert. Extending web browsers with a unity 3d-based virtual worlds viewer. **IEEE Internet Computing**, n.15, p.15-21, 2011.

KURKOVSKY, Stan. Engaging students through mobile game development. In Proceedings of the 40th ACM technical symposium on Computer science education, **SIGCSE '09**, New York, NY, USA, p.44-48, 2009. ACM.

MILOSEVIC, Nikola; DEHGHANTANHA, Ali; KIM-KWANG, Raymond Choo. Machine learning aided android malware classification. **Computers & Electrical Engineering**, n.61, p.266-274, 2017

MOSYNC. What is mosync. Available in: <http://www.mosync.com/what-is-mosync>. Access in: 08/01/2018.

MOTA, José Miguel *et al*. Augmented reality mobile app development for all. **Computers & Electrical Engineering**, n.65, p.250-260, 2018.

NAGESH, Anirudh; CAICEDO, Carlos. Cross-platform mobile application development. **ITERA 2012 Conference**, Indianapolis, 2012.

PAPAGIANNAKIS, George; SINGH, Gurminder; MAGNENAT-THALMANN, Nadia. A survey of mobile and wireless technologies for augmented reality systems. **Computer Animation and Virtual Worlds**, v.19, p.3-22, 2008.

Recebido em: 15 de Agosto de 2018
Avaliado em: 22 de Agosto de 2018
Aceito em: 1 de Setembro de 2018

1 Doutor em Ciencias de la Computatión, Pesquisador na Faculdad de Ciencias Exactas de Universidad National del Centro de la Provincia de Buenos Aires (UNCPBA-Argentina/ ISISTAN (CONICET-UNICEN) Research Institute) – email: guillermo.rodriguez@isistan.unicen.edu.ar.

2 Doutorando em educação – Unit, Mestre em Ciências da Computação – UFS, Bacharel em sistemas de informação, Professor – Unit e pesquisador ITP. Email: gomesrocha@gmail.com.