EMBEDDING MULTIMEDIA CONTENT WITHIN VIRTUAL ENVIRONMENTS
The OGRE Approach

Paulo N. M. Sampaio[1,2], Roberto Ivo C. de Freitas[2] and Gonçalo Nuno P. Cardoso[2]
[1]Centre for Informatics and Systems of the University of Coimbra (CISUC)
[2]Laboratory of Distributed Systems and Networks (Lab-SDR)
University of Madeira (UMA), Campus da Penteada 9000-390, Funchal, Madeira, Portugal

Keywords: Virtual Reality, OGRE, Multimedia Documents.

Abstract: Most of the tools and languages for modeling Virtual Reality environments, such as VRML, X3D, Java3D, etc. do not provide means of describing the synchronized presentation of multimedia content inside these environments. Multimedia has demonstrated its capabilities of motivating users and capturing their attention, which is an interesting characteristic when we want to provide a higher degree of immersion inside Virtual Reality applications. This paper presents a robust and generic solution for the integrated presentation of different kinds of media objects inside virtual environments based on the Graphical Engine OGRE.

1 INTRODUCTION

The integration of multimedia content inside Virtual Environments (VEs) is a promising and interesting trend in the development of Virtual Reality (VR) applications since interaction can be enhanced, and through the addition of audio and video, the user´s immersion inside the VE can be improved. Indeed, we understand the integration of multimedia content as the temporal and logical synchronization of different media objects (with at least one audio or video) rendered inside a VE.

Many VR systems have been proposed in the literature developed within different application domains: e-learning (Chee, 2001), (McArdle et al., 2004), (Halvorsrud et al., 2004), collaboration among workgroups (Bochenek and Ragusa, 2003), augmented collaborative spaces (Pingali and Sukaviriya, 2003), Multimodal VR applications (Carrozino et al., 2005), among others. Indeed, the rapid prototyping, modeling and authoring of VEs has been a major concern to many authors, as presented in (Rodrigues and Oliveira, 2005), (Osawa et al., 2002), (Ficheman et al., 2005) and (Garcia et al., 2002). Although, most of the systems propose the development of VEs, few of them (or none) explore the presentation of integrated multimedia content inside VEs (Walczak et al., 2006).

This paper presents a robust solution to provide the integration of multimedia content inside a VE based on the Graphical Engine OGRE (http://www.ogre3d.org/). The API implemented is called OGRE-Multimedia, and can be applied to any VR application to allow their customization with multimedia content.

OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, flexible 3D engine written in C++ designed to make it easier and more intuitive for developers to produce applications using hardware-accelerated 3D graphics. The class library abstracts all the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other intuitive classes. When comparing OGRE with other different languages for describing virtual worlds (such as VRML (http://www.w3.org/MarkUp/VRML), X3D (Extensible 3D (X3D)) or Java3D (http://java.sun.com)), the choice was not hard to make, specially when it comes to favoring design quality, flexibility and clear documentation.

This paper is organized as follows: Section 2 presents a solution to customize a VE with multimedia presentations; Section 3 presents the main architecture of the API developed; Section 4 illustrates a Multimedia and Virtual Reality application, and; Finally, Section 5 presents some conclusions.

N. M. Sampaio P., Ivo C. de Freitas R. and Nuno P. Cardoso G. (2008).
EMBEDDING MULTIMEDIA CONTENT WITHIN VIRTUAL ENVIRONMENTS - The OGRE Approach.
In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications*, pages 524-531
DOI: 10.5220/0001098905240531
Copyright © SciTePress

2 CUSTOMIZING MULTIMEDIA PRESENTATIONS WITH XML

The main goal of this project is to propose and develop an API for providing the presentation of multimedia content within an OGRE´s virtual environment (VE). The multimedia content is related to the presentation of any multimedia player (such as RealPlayer (http//www.real.com/player), GRiNs (www.oratrix.com/GRiNS/SMIL2.0), etc.), Flash (Adobe – Flash Player) or a web-browser. The idea is to customize and render the multimedia presentation as textures over any 3D object inside the VE.

However, an issue is considered when it comes to the integration of a multimedia presentation inside a virtual environment (VE). This issue is related to the characterization of the temporal and logical structures of the multimedia document, and how to respect these synchronization constraints inside the VE. Besides, these synchronization constraints still may depend on external events to the VE, such as user interactions or events notifying the availability of resources.

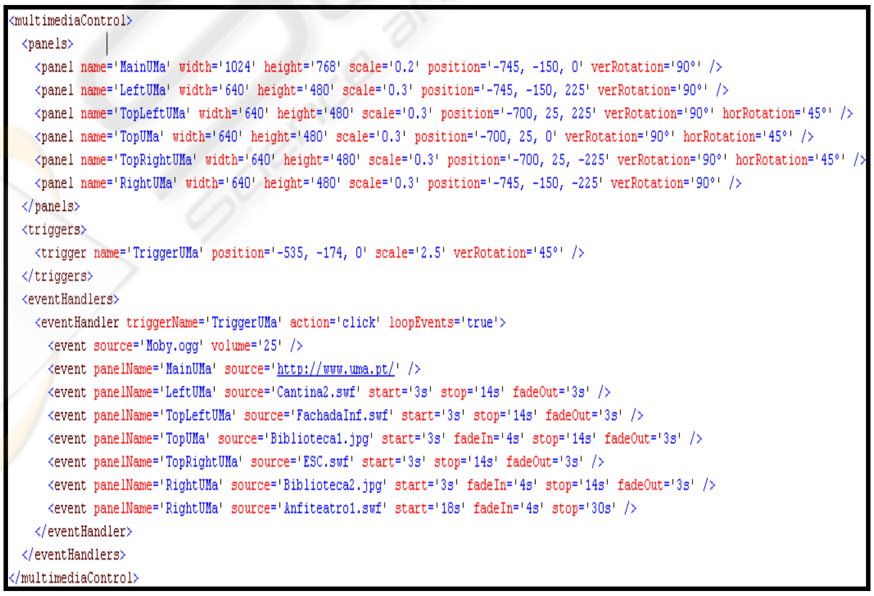
The solution for that relied on the proposal of an XML-based meta-language for describing multimedia documents inside virtual environments, or as we called the *meta-multimedia document*. The *meta-multimedia document* is applied as a multimedia authoring language where users can customize the virtual environment and describe what is going to be presented, where and when they will be presented. Briefly, it describes all the components of the multimedia presentation and their temporal and logical synchronization. The particularity about this presentation is that all the media objects (multimedia documents, flash, web-browsers, and primitive media objects such as video, image, text, audio, etc.) are synchronized and rendered anywhere inside the virtual environment. The interpretation and coordination of this document presentation inside the VE is done by the API developed for OGRE, the *OGRE-Multimedia*.

As presented in Figure 1, the structure of the *meta-multimedia document* is composed of four main elements: *panel*, *trigger*, *eventHandler* and *event*.

Each *panel* element describes a presentation panel for media objects inside the virtual environment. The container called *panels* is a set of the panel objects that must be rendered inside a VE.

Each element *trigger* characterizes an object inside the VE which controls the activation and deactivation of a multimedia presentation. The container called *triggers* is a set of all the trigger objects that will be used to control the multimedia presentations inside a VE.

Each element *EventHandler* characterizes how the presentation of the media objects associated with a given *trigger* will be controlled (e.g., start their presentation when the user clicks on the trigger or when he approximates it). The container called *eventHandlers* is a set of all multimedia presentation described by all the elements *eventHandler*.



```
<multimediaControl>
  <panels>
    <panel name='MainUMa' width='1024' height='768' scale='0.2' position='-745, -150, 0' verRotation='90°' />
    <panel name='LeftUMa' width='640' height='480' scale='0.3' position='-745, -150, 225' verRotation='90°' />
    <panel name='TopLeftUMa' width='640' height='480' scale='0.3' position='-700, 25, 225' verRotation='90°' horRotation='45°' />
    <panel name='TopUMa' width='640' height='480' scale='0.3' position='-700, 25, 0' verRotation='90°' horRotation='45°' />
    <panel name='TopRightUMa' width='640' height='480' scale='0.3' position='-700, 25, -225' verRotation='90°' horRotation='45°' />
    <panel name='RightUMa' width='640' height='480' scale='0.3' position='-745, -150, -225' verRotation='90°' />
  </panels>
  <triggers>
    <trigger name='TriggerUMa' position='-535, -174, 0' scale='2.5' verRotation='45°' />
  </triggers>
  <eventHandlers>
    <eventHandler triggerName='TriggerUMa' action='click' loopEvents='true'>
      <event source='Moby.ogg' volume='25' />
      <event panelName='MainUMa' source='http://www.uma.pt/' />
      <event panelName='LeftUMa' source='Cantina2.swf' start='3s' stop='14s' fadeOut='3s' />
      <event panelName='TopLeftUMa' source='FachadaInf.swf' start='3s' stop='14s' fadeOut='3s' />
      <event panelName='TopUMa' source='Biblioteca1.jpg' start='3s' fadeIn='4s' stop='14s' fadeOut='3s' />
      <event panelName='TopRightUMa' source='ESC.swf' start='3s' stop='14s' fadeOut='3s' />
      <event panelName='RightUMa' source='Biblioteca2.jpg' start='3s' fadeIn='4s' stop='14s' fadeOut='3s' />
      <event panelName='RightUMa' source='Anfiteatro1.swf' start='18s' fadeIn='4s' stop='30s' />
    </eventHandler>
  </eventHandlers>
</multimediaControl>
```

Figure 1: Example of a meta-multimedia document.

Each element *event* characterizes how and when the presentation of each media object of a given *eventHandler* will be carried out.

The structure of the meta-multimedia document was defined to make the process of authoring the multimedia document easier and intuitive. We consider the *meta-multimedia document* as the key-solution for the integration of multimedia content inside VEs. Indeed, with this document, the author of the application is able to customize his virtual environment without changing a single line of his code.

3 OGRE-MULTIMEDIA: INTEGRATING MULTIMEDIA WITHIN VIRTUAL ENVIRONMENTS

Most of the libraries available for creating Virtual Reality applications do not have appropriate APIs for the integration of multimedia content inside a VE. Some languages and platforms such as VRML, X3D, Java3D and OGRE, provide only APIs for the presentation of single media objects (such as video or audio) without integrating or synchronizing these objects.

The solution proposed with OGRE-Multimedia is to provide an API to integrate the presentation of different multimedia objects (rendered by different plug-ins or APIs) around the definition of the *meta-multimedia document*. In this sense, the meta-multimedia document describes all the synchronization relations among all the components (media objects, Flash presentation, web-browsers, etc.) of the document. Taking advantage of the OGRE´s component-based architecture, this API can be easily instantiated and integrated with the remaining available library. This section presents the main architecture of OGRE-Multimedia

The architecture of OGRE-Multimedia describes the integration of the *meta-multimedia document* with the Virtual Environment, which are supported by the implemented software modules and some existing APIs. This architecture is depicted in Fig. 2.

The architecture of *OGRE-Multimedia* is composed of four main components: (1) *External modules* (TinyXML, OIS, OgreAL, DevIL, Navi and OGRE), (2) *Elementary modules* (*Meta-multimedia document* and Virtual Environment), (3) *System startup*, and (4) *System Update*.

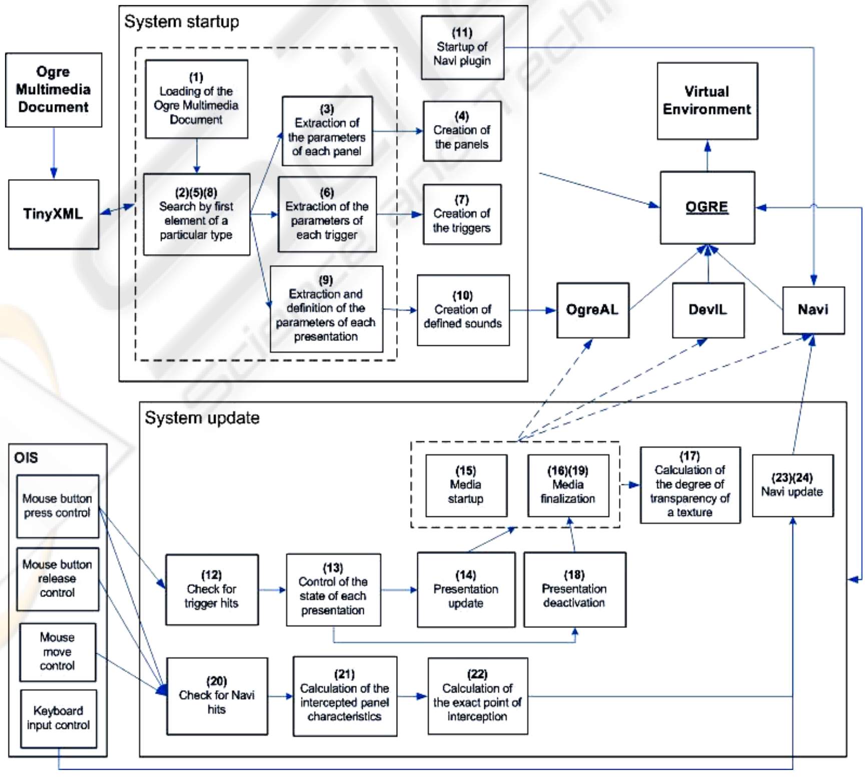

System startup
Ogre Multimedia Document
TinyXML
(1) Loading of the Ogre Multimedia Document
(2)(5)(8) Search by first element of a particular type
(3) Extraction of the parameters of each panel
(6) Extraction of the parameters of each trigger
(9) Extraction and definition of the parameters of each presentation
(11) Startup of Navi plugin
(4) Creation of the panels
(7) Creation of the triggers
(10) Creation of defined sounds
Virtual Environment
OGRE
OgreAL
DevIL
Navi
System update
OIS
Mouse button press control
Mouse button release control
Mouse move control
Keyboard input control
(15) Media startup
(16)(19) Media finalization
(17) Calculation of the degree of transparency of a texture
(23)(24) Navi update
(12) Check for trigger hits
(13) Control of the state of each presentation
(14) Presentation update
(18) Presentation deactivation
(20) Check for Navi hits
(21) Calculation of the intercepted panel characteristics
(22) Calculation of the exact point of interception

Figure 2: Architecture of OGRE-Multimedia.

External Modules. The external modules are represented by those APIs developed by other projects, or which were already provided by the OGRE´s library. These modules were helpful for the implementation of OGRE-Multimedia, and demonstrate how this API can be integrated straightforward with other plug-ins to enable the presentation of multimedia content inside VEs. These modules are:

- TinyXML: This API (TinyXML) provides an XML syntactic analyzer (parser) developed in C++. TinyXML is applied for reading information described on the *meta-multimedia document*;
- OIS: This module (OIS) describes a simple solution for managing the interaction through the input devices (such as mouse, joystick, keyboard, etc.) and feedback devices (vibrating commands, etc.);
- OgreAL: This module (OgreAL Documentation) is related to a library, based on the library OpenAL (www.openal.org), which provides functionalities for the presentation of audio inside an OGRE application. OgreAL is applied in conjunction with *OGRE-Multimedia* to provide the presentation of all the audio information declared on the *meta-multimedia document*.
- DevIL: This API (DevIL) is related to the library used by OGRE for the presentation of images inside VEs. DevIL is applied by OGRE-Multimedia for the presentation of images declared on the *meta-multimedia document*;
- Navi: This API (Simmons) enables the presentation and interaction of a web browser window (based on Firefox 2.0) in an OGRE application. This API is integrated with OGRE-Multimedia to provide the presentation of flash (.swf) content such as video or animations;
- OGRE: This module represents the graphic engine OGRE which is the main module of the system being responsible for creating, managing and updating the tri-dimensional model.

Elementary Modules. The elementary modules describe the non-functional components of the architecture, which are used as a support for the application. These modules are:

- Meta-multimedia document: Which describes all the information about the media objects and their temporal relations, as presented in section 2;
- Virtual Environment: This module is related to the tri-dimensional models (*panels* and *triggers*) applied to support the multimedia presentation inside the Virtual Reality application.

System Startup. This component is in charge to set up the presentation of multimedia content in VEs. Therefore, the modules of this component are executed only once when the application is launched. These modules are numbered following the sequence they are executed:

1) *Loading the meta-multimedia document*: This modules reads and loads into memory the multimedia document (XML), building a DOM (http://www.w3.org/DOM) out of this file. The API TinyXML is used for this purpose;

2) **5)** and **8)** *Search by the first element of a particular type*: This module analyses DOM and search for the first element of a given type. This step is executed three times, each one searching for the first occurrence, respectively, of a *panel*, a *trigger* and an *eventHandler*;

3) **6)** and **9)** *Extraction and definition of the parameters for each presentation*: These modules extracts and loads all the parameters necessary for the creation of each *panel*, *trigger* and *eventHandler* (and *events*);

4) **7)** and **10)** *Creation and rendering each panel, trigger and defined audio*: These modules create the respective *panel*, *trigger* and prepare the presentation of the audio, based on the parameters defined on the *meta-multimedia document*;

11) *Startup of the Navi plug-in*: This module prepares the system for the utilization of texture-based videos, animations and web pages, using the functionalities of *Navi*.

System Update. This component is in charge to control and update the multimedia presentation inside the VE according to possible user interactions. Therefore, the modules of this component are executed several times, once the modules of the component *startup* are executed. Similar to the modules of *startup*, the modules of this component are also numbered following the sequence of execution inside the architecture:

12) *Checking for user interaction on triggers*: This module verifies if a user interacted with a trigger by clicking over it. This module uses the functionalities of OIS and OGRE;

13) *Controlling of the state of each presentation*: which updates the control parameters related to the state of each presentation (active or inactive), according to their triggering type (selection or proximity) and user interaction;

14) *Updating the presentation*: This module is responsible for timing each multimedia presentation and controlling each media object being presented. This module also monitors and activates a *trigger* object based on the proximity of the user;

15) *Media startup*: This module launches the presentation of a media object. For this purpose, the appropriate plug-in is applied for the presentation (by the creation of a multimedia texture for visible objects), such as *DevIL* for images, *Navi* for video, animation and web pages, and *OgreAL* for audio;

16) and **19)** *Media finalization*: This module interrupts the presentation of a media object, eliminating the respective texture for visible media objects. This interruption is done based on the presentation duration for this object declared on the *meta-multimedia document*;

17) *Calculation of the degree of transparency of a texture*: This module calculates the degree of transparency of a given texture, in case there is a appearing/fading effect associated with it;

18) *Deactivation of presentation*: This module is responsible for interrupting a given presentation by invoking the module *media finalization* to interrupt the presentation of the media objects currently being presented;

20) *Checking for user interaction on Navi*: This module verifies if there was a user interaction with a panel whose texture is a Navi web browser. This module is useful to determine, in the future, the exact interaction point with the web browser;

21) *Calculation of the intercepted panel characteristics*: This module determines the characteristics of the 3D object target of the user interaction. These characteristics include the number polygons, position of polygons, etc. This information is useful to determine which polygon the user interacted with;

22) *Calculation of the exact point of interception*: This module determines the exact point of a user interaction with the Navi web site presentation. This is helpful to follows hyper-links defined in a web site.

23) and **24)** *Navi update*: which is in charge of updating a Navi web browser texture. The texture is updated as a result of a web page navigation, introduction of text in the web site, etc.

Some implementation details and issues are further discussed on the next section.

4 THE PROTOTYPE IMPLEMENTED

This section illustrates the prototype implemented by the presentation of an OGRE-Multimedia application. This application applied a virtual world (also called map) result of another project (Cardoso, 2007). This map describes a three-store building where the capabilities (distribution, communication, physics, Artificial Intelligence, enhanced textures, emission of particles, etc.) of OGRE and its related APIs were exploited. Figure 3 illustrates a global view of this application.

OGRE-Multimedia was easily integrated to this OGRE application, where the methods of OGRE-Multimedia were invoked to enable the presentation of multimedia content previously defined on the *meta-multimedia document*, called "MMDocument.mmc". In the case of this VR application, the first floor of the building is reserved for the presentation of multimedia content is an exposition-like fashion. This floor is an open-wide area, where in front of each wall there is a sensitive column which can be activated by the user's click or proximity (depending on its previous configuration on the *meta-multimedia document*) in order to trigger the multimedia presentation on the wall (Figures 3a and 4a).

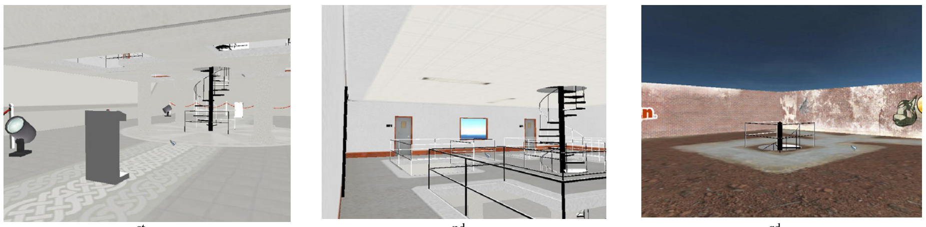

(a) 1st Floor (b) 2nd Floor (c) 3rd Floor

Figure 3: Global view of the virtual three-store building.

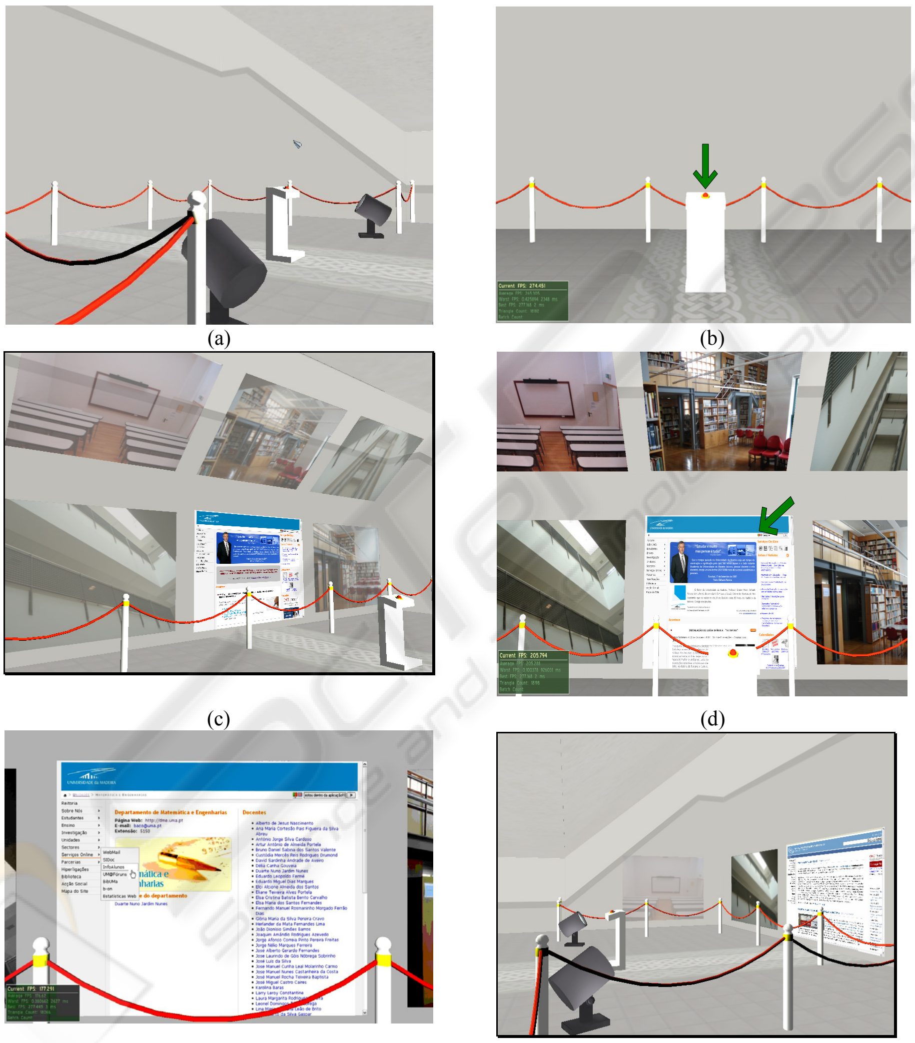

(a) (b) (c) (d) (e) (f)

Figure 4: Presentation of the multimedia content inside the virtual environment.

Figure 4(b) illustrates the interactive column in the virtual environment. In particular, this column (which is represented by a *trigger* element on the *meta-multimedia document*) is able to launch a presentation by a user´s click.

The multimedia presentation is launched after the user interacts (by clicking) with the sensitive column (Figures 4c). When the media objects start to be presented, their level of transparency is gradually changed producing the effect of fading-in. As we can see in Figure 4(d), a web-browser is also presented as one of the multimedia textures. This browser is rendered by the API Navi which allows the user to navigate on the Web. Figures 4(d), 4(e) and 4(f) present an example of this navigation.

All the media objects presented inside the VE are synchronized and managed by OGRE-Multimedia which keeps pace of each presentation starting and interrupting all the media objects according to their previous configuration on the *meta-multimedia document*. OGRE-Multimedia enables the multimedia presentation inside the VE making of it a more realistic environment and, above all, keeping the user´s focus.

5 CONCLUSIONS

This paper presented the development of an API for the presentation of integrated multimedia content inside Virtual Environments based on the Graphical Engine OGRE, called OGRE-Multimedia. The integrated presentation of multimedia content inside VEs relied on the proposal of an XML-based representation to describe all the media objects to be presented and their synchronization relations, the *meta-multimedia document*. OGRE-Multimedia can be applied straightforward in different OGRE Virtual Reality applications since it is the result of an open-architecture where different APIs were applied in conjunction to provide the presentation of different kinds of media objects including the traditional images, audio, video, animations, etc., and also multimedia documents such as FLASH, and web-browsers as well. Indeed, the combination of Multimedia and Virtual Reality can be successfully applied to the design of robust applications where users feel more comfortable and have their focus inside the VE, definitely augmenting their feeling of immersion.

REFERENCES

Chee, Y.S., Network Virtual Environments for Collaborative Learning. Invited talk. In Proceedings of ICCE/SchoolNet 2001—Ninth International Conference on Computers in Education, Seoul, S. Korea. ICCE/SchoolNet (2001) 3–11.

McArdle, G.; Monahan, T.; Bertolotto, M.; Mangina, E.: A Web-Based Multimedia Virtual Reality Environment for E-Learning. Proceedings Eurographics 2004, July 2004, Grenoble, France.(2004)

Halvorsrud, R.; Hagen, S.: Designing a Collaborative Virtual Environment for Introducing Pupils to Complex Subject Matter. NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction. Tampere, Finland, (2004), 121-130, ISBN: 1581138571.

Bochenek, G.M.; Ragusa, J.M.: Virtual (3D) collaborative environments: an improved environment for integrated product team interaction?: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003. (2003), 10 pp.

Pingali, G.; Sukaviriya, N.: Augmented Collaborative Spaces. In proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence, International Multimedia Conference. Berkeley, California, (2003).

Carrozino, M.; Tecchia, F.; Bacinelli, S.; Cappelletti, C.; Bergamasco, M.: Lowering the development time of multimodal interactive application: The real-life experience of the XVR Project.: In ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, (2005).

Rodrigues, S. G. & Oliveira, J. C.: ADVICe - um Ambiente para o Desenvolvimento de ambientes VIrtuais Colaborativos. XI Simpósio Brasileiro de Sistemas Mutimídia e Web - WebMedia2005, Poços de Caldas, MB, Brasil, (2005).

Osawa, N.; Asai, K.; Saito, F.: An interactive toolkit library for 3D applications: it3d. In proceedings of the workshop on Virtual environments 2002, ACM International Conference Proceeding Series; Vol.23, Barcelona, Spain, (2002). 149 - 157

Ficheman, I. K.; Pereira, A. R.; Adamatti, D. F.; Oliveira, I. C. A.; Lopes, R. D.; Sichman, J. S.; Amazonas, J. R. A.; Filgueiras, L. V. L.: An interface usability test for the editor musical. In proceedings: International Conference on Enterprise Information System - ICEIS 2005. Miami USA v. 5, (2005) 122-127.

Garcia, P.; Montalà, O.; Pairot, C.; Skarmeta, A.G.: MOVE: Component Groupware Foundations for Collaborative Virtual Environments. Proceedings of the 4th international conference on Collaborative virtual environments, Bonn, Germany, (2002) 55 – 62.

Walczak, K., J. Chmielewski, M. Stawniak, S. Strykowski.: Extensible Metadata Framework for Describing Virtual Reality and Multimedia Contents. Proceedings of the 7th IASTED International Conference on Databases and Applications DBA 2006. Innsbruck, Austria¸ (2006) 168-175.

OGRE 3D: Open source graphics engine. URL: http://www.ogre3d.org/

VRML: Virtual Reality Modeling Language. URL: http://www.w3.org/MarkUp/VRML/

ISO/IEC 19775:2004 — Extensible 3D (X3D). URL: http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/

Java3D API, URL: http://java.sun.com/products/java-media/3D/

RealPlayer - Real Networks, 2004. URL: http// www.real.com/player/.

GRiNS – SMIL 2.0 Player, Home Page. URL: http:// www.oratrix.com/GRiNS/SMIL2.0/.

Adobe – Flash Player. URL: http://www.macromedia.com/ software/flash/about/

TinyXML – XML Parser. URL: http:// www.grinninglizard.com/tinyxml/

OIS – Object Oriented Input System. URL: http://sourceforge.net/projects/wgois/

OgreAL Documentation. URL: http:// www.mooproductions.org/ogreal/documentation/

OpenAL – Cross-Platform 3D Audio. URL: http:// www.openal.org/

DevIL Development Team. DevIL – A full featured cross-platform Image Library. URL: http:// openil.sourceforge.net/

Simmons, A. Main Page – NaviWiki. URL: http:// navi.agelessanime.com/wiki/index.php/Main_Page

DOM – Document Object Model. URL: http:// www.w3.org/DOM/

Cardoso, G. "UMa Virtual". B.Sc. dissertation. University of Madeira (UMa), 2007.