Creating 3D Scenarios with OGRE Creativity Labs

Paulo N.M. Sampaio[1], Duarte J.O. Teixeira[2], Duarte M. Fernandes[3]

Salvador University (UNIFACS)[1]
Computing and Systems Research Center - NUPERC
41950-275, Salvador, Bahia, Brazil

email: paulo.sampaio@pro.unifacs.br

Madeira Interactive Technologies Institute (M-ITI)[2,3]
University of Madeira (UMa)
Funchal, Madeira, Portugal

email: duartejoaoornelas@hotmail.com[2],
dmf9000@yahoo.com[3]

Abstract — **Currently, there are several languages and tools to provide the creation of 3D scenarios. However, the existing approaches are not intuitive and require further knowledge of the user. This paper introduces an application to allow the creation of virtual scenarios with high graphical quality called OGRE Creativity Labs (OGRE-CL). OGRE-CL provides OGRE application developers with a graphical environment for the rapid prototyping of 3D scenarios, and the subsequent automatic generation of the respective OGRE code. With this tool, developers can have their development time optimized since they rather focus on the codification of the dynamics and strategies of the application being developed without spending much time with the design of its graphical components.**

Keywords-component; 3D, OGRE, Virtual Reality, XML, Authoring Tool, Virtual Scenarios, OGRE-CL

I. INTRODUCTION

Once 3D application developers have available the set of 3D objects to be deployed in their application, they can apply different languages and tools available for the development of virtual worlds. Nevertheless, the existing solutions are not intuitive and require from the developer a deeper knowledge of their components, representation and structure in order to be able to create a virtual environment. Furthermore, most of the existing solutions still lack the possibility of creating more complex virtual scenarios with: a high graphical quality; the possibility to integrate multimedia presentations inside the virtual world; the development of distributed applications which are able to provide remote navigation or communication tools such as VoIP (Voice over Internet Protocol), chat, etc., among other advanced features.

One of the existing solutions for the development of Virtual Reality applications is the utilization of the graphical rendering engine OGRE (Object-Oriented Graphics Rendering Engine) [1]. OGRE is a scenario oriented 3D game engine written in C ++ which makes available a library of classes (APIs) for the description of virtual worlds and objects in a high abstraction level and graphical quality.

In order to understand the features of the existing OGRE authoring tools, and to better determine the functionalities of the application to be developed, it was important to carry out a literature review. Some of the most promising OGRE editors available are Ogitor [2], OGRE Editor Multi Scene Manager Project Environment [3] and OGRE – MOGRE Editor [4]. These applications are in general well conceived and simple in order to allow the creation of virtual scenarios and they can be found in different states of completeness, sometimes being able to adapt themselves to new APIs and libraries. However, most of these tools have a proprietary representation for the virtual scenario created, and they do not allow exporting the respective OGRE code for further implementations.

The main goal of this paper is to introduce and discuss the development of OGRE Creative Labs (OGRE-CL), a graphical authoring tool for providing the intuitive creation of OGRE virtual scenarios. The development of OGRE-CL has been based on the proposal of an XML-based language called OGREML which is applied for providing the interoperability of OGREML-compliant authoring tools. OGRE-CL is a meaningful solution to OGRE application developers since this tool allows the rapid prototyping of 3D environments, and the subsequent automatic generation of OGRE code. OGRE-CL allows developers to optimize the prototyping of their applications since the time dedicated to the design of their graphical interface is reduced, allowing developers to focus on the programming of the dynamics and strategies of their applications. Thus, OGRE-CL is helpful to reduce considerably the development time of OGRE applications.

This paper is organized as follows: In the next section the structure of OGREML is presented; After that, some OGRE-CL´s implementation issues are discussed, followed by the presentation of some functionalities of this tool; The automatic generation of OGRE code is also discussed in the sequence, and; finally some conclusions of this work are presented.

II. OGRE MARKUP LANGUAGE (OGREML)

According to the literature review, it was possible to verify that there are few authoring tools for OGRE 3D environments. Even though, these tools have a proprietary code and do not allow the generation of the final OGRE code (C++ or C#). For this reason, it was important to provide a solution for generating OGRE code automatically, thus optimizing the prototyping of OGRE applications.

With this goal in mind, the XML-based language OGREML (OGRE Markup Language) was proposed at University of Madeira [5] as a common representation for the description of OGRE 3D scenarios to be exchanged among different OGREML-compliant development tools. OGREML is composed of four major components: *Ambient*, *Objects*, *Multimedia* and *NetworkConfiguration*. All these components are gathered in the main block SceneConfiguration, as depicted in Figure 1.

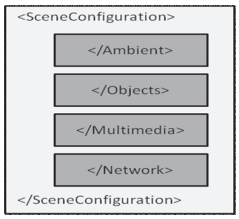


Figure 1. Main components of OgreML

The component *Ambient* contains all information about the environment of the scenario, such as the color of ambient light, fog, sky, plane and camera. This component also contains a description of the sky, plane and values of position and rotation of the camera.

The *Objects* component contains a description of all objects in the scene and their attributes such as: id, position in the scenario, scale and rotation, mesh file and if it can cast shadows.

The component *Multimedia* defines synchronized multimedia presentations that may consist of videos, images and sounds, and they will be displayed in the virtual world after the activation of a trigger [6].

The *NetworkConfiguration* represents the configuration of different network services among distributed OGRE applications, such as chat, VoIP and distributed navigation [7].

OGREML uses a simple description to represent all the components of an OGRE 3D scenario, being at the same time expressive since it supports most of the OGRE development components. In the next section we present further details about the implementation of OGRE-CL.

III. OGRE-CL: Design and Implementation Issues

Some of the main issues related to the implementation of OGRE-CL are related to its requirements, architecture, and implementation aspects.

The requirements are related to the functional and non-functional features of a system to be developed [8,9]. Some of the functional requirements defined for the development of OGRE-CL are:

- create a 3D virtual scenario;
- open a previously created virtual scene, and;
- Configure and edit a virtual scenario according to a set of parameters supported by OGREML, such as:
  - set the ambient color of the scene;
  - set the shadow type used in the scenario;
  - define the type of fog technique;
  - add/remove lights to the scenario;
  - change and define the light type (specular/diffuse light);
  - change light position/direction;
  - define sky type;
  - define the plane;
  - define camera values;
  - add/remove objects to the scenario;
  - change position, scale and rotation of the objects in the scene, and;
  - insert and remove mesh files to/from the library of objects.

The OGRE-CL architecture is divided into three sub-modules (Figure 2): User Interface, OGRE-CLGUI and OGRE-CL3D.

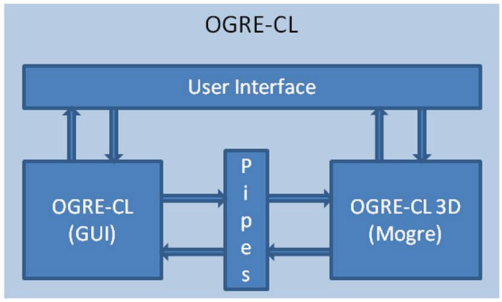


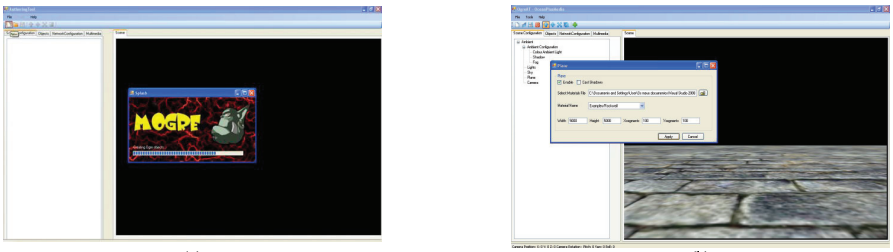
Figure 2. Architecture of OGRE-CL

The User Interface is responsible for receiving all the user interactions and directing them to their respective module for subsequent processing. The module OGRE-CLGUI is responsible for handling the interactions with the user, opening new projects, configuring the 3D scenarios and saving projects in OGREML files. At last, The module OgreCL3D is responsible for interacting with OGRE, applying the changes to the virtual scenario. Besides setting the scenario, this module is also responsible for managing user interactions. The communication between OGRE-CLGUI and OGRE-CL3D is done through pipes - an application sends a command, the other application receives it, interprets it and, based on internal processes, executes it.

OGRE-CL has been implemented using Visual Studio C # 2008 [10].

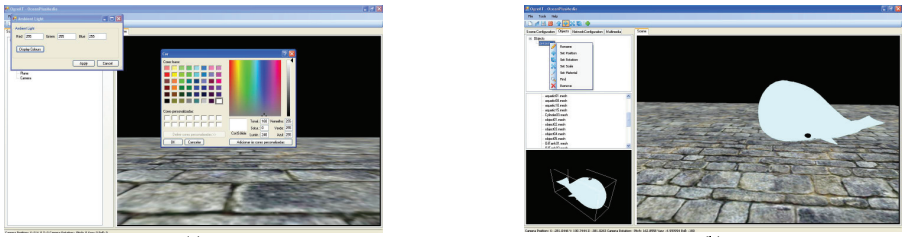
IV. OGRE-CL Functionalities

This section illustrates the utilization of OGRE-CL for the authoring of 3D scenarios. After starting OGRE-CL, the initial screen presented to the user is composed of two main parts (Figure 3a): (1) the left column where all the configuration of the scenario can be done (configuration area), according to the OGREML scenario structure (e.g., Scene Configuration, Objects, Multimedia and Network Configuration), and; the right window, also called OGRE rendering window, where the virtual scenario is presented.



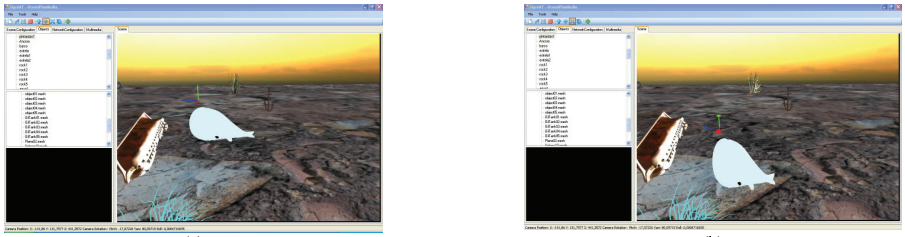
(a) (b)

Figure 3. Initialization of OGRE-CL and Plane Configuration



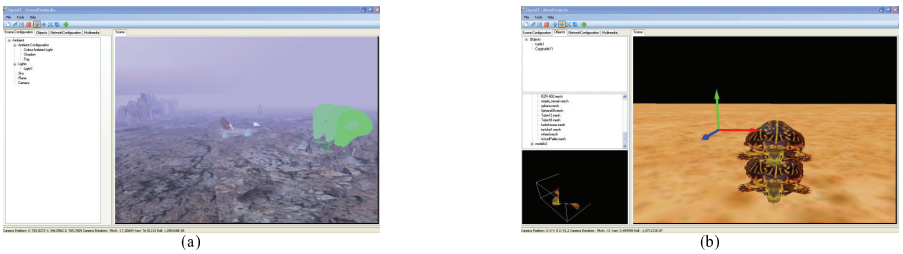
(a) (b)

Figure 4. Ambient Light Color Configuration / Choosing objects from the library



(a) (b)

Figure 5. Move an object position / scale and object in the scenario



(a) (b)

Figure 6. Application of a fog technique / Cloning an object

When the user creates a new project (virtual scenario) with OGRE-CL, an empty rendering window is presented. Initially, when creating a new scenario, we should configure the main characteristics of this scenario (Scene Configuration). Figure 3b illustrates the plane configuration. After entering values and clicking on the "Apply" button, we can see in the rendering window the presentation of the newly created plane. At any time, these parameters can be modified in order to customize the scenario.

Some of the functionalities of OGRE-CL illustrated through Figures 3 to 6 are:

- Configuration of the ambient light color using a palette of colors (Figure 4a);
- Addition of new objects from the library to the scenario (Figure 4b);
- Manipulation of the 3D objects within the scenario according to an X (red arrow), Y (green arrow) or Z (blue arrow) axis, as depicted in Figure 5a;

- Scaling an object´s size directly in the rendering window (Figure 5b);
- Application of a fog technique (Figure 6a), and;
- Cloning one or more objects in the scenario simultaneously (Figure 6b).

As we can notice, OGRE-CL is an easy to use graphical environment for optimizing the creation of OGRE applications. With this tool OGRE developers are able to quickly design the main virtual scenarios of their applications, being able to further generate the respective C# code related to the virtual scenarios conceived.

V. Generating OGRE Code Automatically

The development of OGRE applications requires a considerable programming effort from the OGRE developer since all the code must be developed in C++ or C#. For this reason, the development process of these applications is rather complex and time-consuming since the OGRE developer has to code all the virtual scenarios to be applied in his application. After that, the developer is supposed to implement all the dynamic aspects of his application, such as the strategies, events handling, artificial intelligence, etc.

In order to facilitate the development of OGRE virtual scenarios a generic solution was proposed for promoting the interoperability among different OGRE authoring tools, and further automatic generation of the respective code with the description of the virtual scenarios. This solution relies on the utilization of OGREML for the complete description of OGRE virtual scenarios and further exchanging this description among different OGREML-compliant tools.

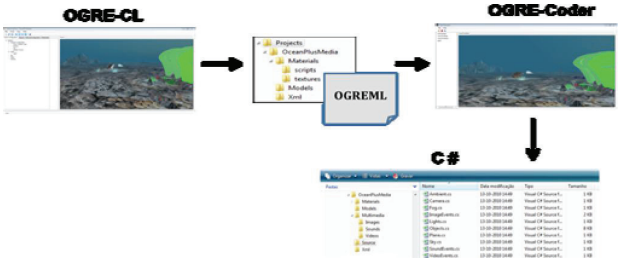


Figure 7. Interoperability among OGREML-Compliant tools

For instance, virtual scenarios can be created using OGRE-Creative Labs (OGRE-CL), and by means of OGREML, be exported to Ogre Coder (OGRE-Coder), also developed at University of Madeira [5] for the automatic generation of C # code, as illustrated in Figure 7. The integration of an authoring tool for the creation of virtual scenarios (OGRE-CL) with OGRE Coder for the automatic generation of C # code allows a more rapid prototyping of OGRE applications.

VI. Conclusions

In this paper we presented the main aspects of the development of a useful tool for the rapid development of 3D scenarios, OGRE Creative Labs (OGRE-CL). The development of OGRE applications attracts a large community of interested developers in particular due its high graphical quality and the amount of APIs available for development. OGRE-CL provides a simple interface for the composition of virtual scenarios based on OGRE. Also some other important characteristics of OGRE-CL is that it relies on an extensible library of objects (meshes) that can be used to create the virtual scenarios, and; with the proposal of OGREML, this language provides the complete description of OGRE virtual scenarios, and promotes the interoperability of OGRE authoring tools.

With the utilization of OGREML, OGRECL becomes an interesting solution for OGRE developers since they can minimize the time for designing the graphical interface of their applications, and after that they can apply OGRE-Coder to generate automatically the OGRE code related to his scenarios. Another advantage of this approach is that it encourages the team development since graphical designers can work together with OGRE developers in order to produce more appealing and effective OGRE applications.

References

1. OGRE – Open Source 3D Graphics Engine. (2001). Last visited in November 2010. http://www.ogre3d.org/
2. Ogitor SceneBuilder. Last visited in November 2010. http://www.ogitor.org/HomePage
3. OGRE Editor Multi Scene Manager Project Environment. Last visited in November 2010. http://www.youtube.com/watch?v=TU_Tc4EBLHQ
4. OGRE – MOGRE Editor. Last visited in November 2010. http://www.youtube.com/watch?v=xXJDvKzY1Us&feature=related
5. Fernandes, M.D. (2010). Basis for the Automatic Generation of OGRE Virtual Scenarios (In Portuguese). M.Sc. Dissertation in Informatics Engineering – University of Madeira, Madeira, Portugal.
6. Freitas, R. (2007). Multimedia Presentation in Virtual Environments (In Portuguese). B.Sc. Dissertation in Informatics Engineering – University of Madeira, Madeira, Portugal.
7. Cardoso, G. (2007). Uma Virtual – Basis for the development of complex virtual environments (In Portuguese). B.Sc. Dissertation in Informatics Engineering – University of Madeira, Madeira, Portugal.
8. Oberg, Roger. Probasco, Leslee. & Ericsson, Maria(2010). Applying Requirements Management with Use Cases. Rational Software – Technical Paper TP505. On November 8th 2010, from http://www.wthreex.com/rup/papers/pdf/apprmuc.pdf
9. Malan, R.; Bredemeyer, D. (2009). Functional Requirements and Use Cases. Bredemeyer Consulting - White Paper. 8/3/01 On November 8th 2010, from http://www.bredemeyer.com/ use_cases.htm
10. Visual Studios 2010 Editions – Microsoft Visual Studio. Last visited on November 2010. http://www.microsoft.com/visualstudio/en-us/products/2010-editions