Virtual Rome : a FOSS approach to WEB3D

Luigi Calori
CINECA, Bologna – IT
(www.cineca.it)
l.calori@cineca.it

Carlo Camporesi
CNR ITABC, Rome – IT
(www.vhlab.itabc.cnr.it)
carlo.camporesi@itabc.cnr.it

Sofia Pescarin
CNR ITABC, Rome – IT
(www.vhlab.itabc.cnr.it)
sofia.pescarin@itabc.cnr.it

ABSTRACT

The goal of VirtualRome project (www.virtualrome.it) is to provide a web-based infrastructure for distribution, collection, annotation and sharing over the web of 3D interactive content such as actual and reconstructed landscape. For this project we have developed a framework for the integration of 3D realtime application within web browsers; the full functionality is currently available only for Windows Firefox, limited versions are available for Firefox Linux and Explorer. We present a completely FOSS approach to the web deployment of a 3d database consisting of landscape data at different time and different resolution. Components of a deployment platform will be analyzed and We will outline our approach comparing with other available systems.

Categories and Subject Descriptors

Keywords
Browser, 3D, virtual Archaeology

1. INTRODUCTION

How does our landscape looks like in the past? How was the original aspect of that monument or archaeological site, that today appears so ruined and also so immersed in our crowded built cities? Virtual Rome project web 3d application has been designed to help archaeologist answer these questions by using a web deployed 3D application that could deliver their reconstructions to the public.

The goal of being able to switch between different models in time is not new but is not so common for large landscape area. We also would like to being able to show models in high detail and being able to switch between actual and reconstructed state. Another requirement was complete integration in browsers. At the beginning of the project, we did a survey of available off the shelf solutions. At that time we did not found any able to match all the requirements within our budget.

So we decided to find the Open Source project that was most viable and try to add the feature needed.

By taking that decision we were knowingly trading off stability versus feature and performance versus flexibility.

Being the project completely non profit and mostly a research one, we decided to put the developed code in the public domain.

In the following sections we will describe the components of our system in relation with the current state of the art.

Is to be underlined that the evaluation of the state of the art is more directed towards available implementation than academic papers: we did not had enough manpower to allow for much more than integration, thus we concentrated on solutions that were either available as open source components or quite straightforward to implement.

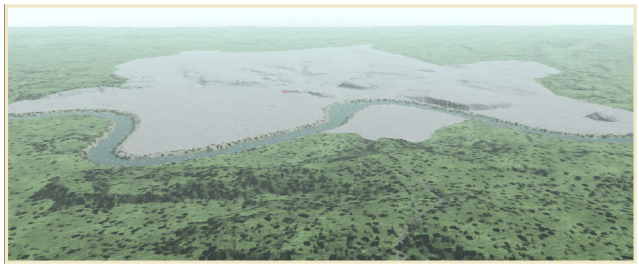
We did not exclude to use also proprietary sw components, unfortunately very few are available as individual reusable libraries.

2. Project requirements

Virtual Rome had the following requirements:

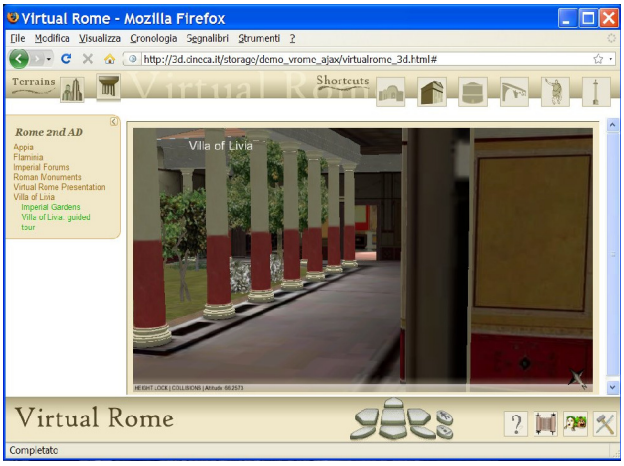
- paged geospatial dataset support (fig 1,2)
- coordinate and projection handling (both in input and in output)
- large 3d terrain dataset management
- 3d models integration (modelled with software such as 3D Studio Max, etc)
- natural elements, such as vegetation, integration
- vector layers integration
- on-line 3d data publication and interaction, possibly embedded into a web browser
- Fly and walk navigation tools
- Behaviours integration:
  - terrains, models switching
  - vector information loading
  - models loading
  - picking and loading external pages or multimedia contents
  - overview map
  - environment integration

1 current landscape



2 Ancient landscape

3 Ancient vegetation



4 Interiors

5 Browser integration

The project started in 2006, at that time, not many system could be used to try to match all the requirements; nevertheless any system is likely to evolve, so we will try to summarize the systems that were considered at time and how they have evolved regarding our requirements list.

To be clearer we will try to detail better some of the requirements:

- **Paged geospatial dataset support** paged we intend the system should be able to access a multi gigabyte hierarchical dataset of geospatial data (images, terrain, features) in real time adapting the image quality produced to the available bandwidth, technique used involve image and terrain compression multithreading, caching etc. Ideally the system should provide both a player component as well as a server component able to build hierarchical database out of standard GIS georeferenced data. The performance of such a systems can be qualitative compared by looking at how fast the perceptual quality increases when a completely new viewpoint is select given a fixed amount of bandwidth and hardware resources. Another point is the amount of visual artefacts such as popping, aliasing, seams that the multi resolution visualization generates.

- **3d models, natural elements (vegetation), visual quality effects (lighting, shadows etc)** These requirements are connected with the flexibility of the system: if there are ways to import models generated with off the shelf modellers, if the paging or streaming system can handle 3d models of interior scenes; is vegetation rendering supported, are state of the art rendering techniques such as shadows and other visual effects supported. A rough qualitative measure could be obtained by comparing visual quality of the system on modelled input with that attained by commercial game rendering engines.

- **web integration** This include all the aspect related to the web usability and ease of integration with typical web portal and applications. The comparison would be again qualitative and somehow subjective: to name some criteria:
  Browsers, operating systems and graphics platform supported
  Browser integrated or standalone application
  Requirements of special streaming server

3. State of the art

In 2006 (and even now) we can subdivide the available systems in:

- VRML-X3D based systems
- Global earth geo-viewers
- Flash based systems
- General purpose Scene Graph based systems

3.1 VRML-X3D systems

While our first implementation of a time aware web application [Nume] was done in VRML and the X3D adoption would have been the most elegant solution: having all the data coded in a

web oriented markup language for which there were several players integrated into browser would have been theoretically the best solution, in 2006 we found that existing implementation of X3D players were quite behind expectation. Being focused on standard compliance, X3D players have difficulty to stay tuned on the last hardware features.

Furthermore, we did found X3D systems lacking features for paging capabilities needed for wide geo-databases.

Furthermore, X3D had not been adopted by big players as Google or Microsoft for their Geo-viewers, giving the impression that data standardization was still a long shot.

Now the X3D players have evolved much, but still their performance in world database paging has to be proven.

3.2 Global Earth Geo-viewers

We put into this category applications like Google Earth, Microsoft Virtual Earth and Nasa Worldwind.

These applications are designed and optimized for interactive browsing of whole earth databases and their performance in that field are the best available (apart from some recent advances see RATMAN CRS4 system[1]) The problem we found in adopting was about flexibility and browser integration: in 2006 the support of general interior model was scarce (kml was not supporting paging), Nasa Worldwind[2] and Microsoft VE did not allow user model insertion. Nasa Worldwind, being open source, could be customized, unfortunately at time it was based on .Net technology that was not mature for cross platform development.

Regarding browser integration, at the time none of them was integrated in browser, all required special server streaming technology, and only windows platform was supported.

Now these products have evolved: both GE and VE are now embeddable in browser with sound javascript SDK [3] (the importance of browser embedding has been acknowledged) and latest version of Worldwind is Java based with better cross-platform support.

Regarding flexibility in data handling, both GE and VE are showing urban building and architectural coverage, furthermore GE, is supporting Collada inside kml (see export from SketchUp)

Recently Google hosted the project Rome Reborn that is showing some sort of paging applied to a wide architectural reconstruction of Rome at imperial age..[4]

3.3 Flash - Shockwave based systems

These application are based on Shockwave[5] that exhibit widespread availability and good browser integration (Windows and OSX only). It has been extensively used for online games and 3d content. The paging system is not built in but must be implemented within the internal scripting language. Some examples: Seat Visual[6] and GeoMind[7] products. Being a closed system, the graphics quality attainable is limited by the functionality available in Shockwave. The performances of these platform seem lower than those of dedicated Geo-viewers and there is almost no open source project.

3.4 General purpose Scene Graph systems

This Is the most flexible solution: A scene graph is a middleware library with a level of abstraction somewhat in the middle between the base graphics library (OpenGL or Direct3D) and a complete application such as a game engine. Our requirements did not need game function such as avatars and advanced interaction such as physics, so we did not need a full game engine. But we did strong multithread capability to allow for efficient data paging. In 2006 there were not much game engines available that were supporting paging, and none among the open source or cheap ones. Regarding Scene Graph middleware, having used SGI Performer and having learned the hard way what happen when you layer on a proprietary middleware that get discontinued, the open source choice was a must.

The sw that best matched our need was OpenSceneGraph[8], especially because it had already built in both the general purpose paging scheme as well as the batch application to build hierarchical tiled geo-databases from GIS data. The paging quality and performance are inferior respect to dedicated Geo-viewers but better than almost all the other solutions. The middleware was also cross platform (limited mainly by bad OpenGL drivers) OpenSceneGraph had also a fairly large of input formats. It had also a quite healthy development community oriented to simulation market where high end hardware capability were stressed. There was also some example of small game using the library. To confirm our selection, another project of similar size and scope, Cannes 3D[9] used a proprietary web plug in based on OpenSceneGraph.

Also OpenSceneGraph has been extended with osgearth[10] plugin that is able to page in directly from OGC repositories as well as Google maps and Nasa servers .

Recently other systems has showed suitable for the projects: one is the open source engine Ogre[11], which recently showed able to do some king of paging, also the proprietary engine Unity3d [12]has been enhanced with paging / streaming capability so, being already embeddable in browser, could fulfill the requirements.

[1] http://ratman.sourceforge.net/

[2] http://worldwind.arc.nasa.gov/

[3] http://code.google.com/intl/it-IT/apis/earth/

[4] http://www.romereborn.virginia.edu/.

[5] http://www.adobe.com/products/shockwaveplayer/

[6] http://www.visual.paginegialle.it/

[7] http://www.geomind.it/website/home.htm

[8] http://www.openscenegraph.org/projects/osg

[9] http://3d.cannes.fr/

[10] http://wush.net/trac/osgearth

[11] http://www.ogre3d.org/

[12] http://unity3d.com/

Recently (april 2009) Google has released O3D[13], a new scene graph library wrapped as browser web plugin; A rich Javascript API provide access to the scene graph interface such as loading new models as Collada files. From preliminary evaluation the system does not seem implementing paging out of the box. The support from Google as well as the tight javascript integration make it really promising as a middleware for 3d web applications.

An even more radical approach has been proposed by Mozilla with their Canvas 3d [14]project that provide full javascript access to OpenGL library.

4 Implementation notes

Adoption of OpenSceneGraph as the rendering middleware led us to implement the following components:

1) core application functionalities: all the customization and extension to implement our 3d rendering functions using OpenSceneGraph functionalities, the main being improved scene navigators, pickers and scene management. This has been programmed in C++

2) Embedding the 3D application in the web browser and providing the necessary functionality to cross platform building, packaging, installing, deploying and upgrading. In order to being able to eventually change the rendering middleware and to allow for multiple application to be deployed and updated at the same time, we have a strict decoupling between rendering engine code and browser code. Regarding deployment, we relied on Firefox extension framework for being able to install and update in a cross platform way not requiring administrative privileges. To our knowledge, Explorer is currently not providing any user level ActiveX installation.
This part has been implemented in C++ and get built with Cmake cross platform build system

3) Exposing the core functionality to web page for GUI like user interaction: a simple command list API has been defined to let C++ implementation to communicate with Javascript API library this has been implemented in C++ and Javascript .

4) Server side: the data exchanged are in the native OpenSceneGraph format to allow for best efficiency and are retrieved by the Plugin OpenSceneGraph component thru plain http requests. The Terrain database is generated in a batch preprocessing phase, then published as an http folder. The dynamic information, such as viewpoints, paths, model placements and 3d hyperlinks are entered with a CMS-like simple back office interface and are kept in a MySQL database

5) The front end applications has been designed as a standard AJAX style dynamic web interface, using JSON as the data transport layer. The only caveat is that the embedded OpenSceneGraph plug in is quite heavy in startup/shutdown cycles, so unnecessary page reload should be avoided.

5 Lesson learned and future work

We found that the most time consuming task had been the browser integration and relative testing: the amount of code itself is quite limited but it has a lot of interaction with all the other aspect of the web application such as user interaction (reload, resize, page change, tab opening, multiple instance) tend to generate browser hang and/or memory leaks as two distinct application such as the browser and the rendering engine have to cooperate in the same address space.

One future direction, as showed by Google plugin is to try to execute the rendering engine in a separate process: the main rendering application will be so executed in a separate process that inherit window from the browser but is not able to crash the browser itself in case some problem occurs.

We also would note that the integration and testing code is largely independent from the application and also to the rendering engine used, so it would be a good candidate for an open source project involving several FOSS applications and rendering engines.

6 Acknowledgments

Our thanks to SEAT Pagine Gialle for Sponsorship and data providing, to OpenSceneGraph community for the SW and help

7 References

AKENINE-MÖLLER T. , E. HAINES, R*eal-Time Rendering Second Edition*, pp.477-479, A K Peters Natick, Massachussets US, 2002

KUEHNE B MARTZ P., OpenSceneGraph Reference Manual ver. 2.2, Skew Matrix Software and Blue Newt, 2007

PESCARIN S., FORTE M., CALORI L., CAMPORESI C., GUIDAZZOLI A., IMBODEN S., *Open Heritage: an Open Source approach to 3d real-time and web-based landscape reconstruction,* in VSMM2005, Proceedings of XI International Conference on Virtual Systems and Multimedia: Virtual Reality at work in the 21st century, Oct. 3-7 2005 Ghent Belgium, ISBN 9638046635 (Thwaites Ed.) Budapest 2005, pp.313-320

[13] http://code.google.com/intl/en-EN/apis/o3d/

[14] http://www.c3dl.org/