DESIGN AND DEVELOPMENT OF A FRAMEWORK BASED ON OGC WEB SERVICES FOR THE VISUALIZATION OF THREE DIMENSIONAL LARGE-SCALE GEOSPATIAL DATA OVER THE WEB

E. Roccatello [a], A. Nozzi [a], M. Rumor [b]

[a] 3DGIS – Italy – www.3dgis.it
[b] University of Padua – Italy

KEY WORDS: framework, standards, 3D, large urban data

ABSTRACT:

This paper illustrates the key concepts behind the design and the development of a framework, based on OGC services, capable to visualize 3D large scale geospatial data streamed over the web.
WebGISes are traditionally bounded to a bi-dimensional simplified representation of the reality and though they are successfully addressing the lack of flexibility and simplicity of traditional desktop clients, a lot of effort is still needed to reach desktop GIS features, like 3D visualization.
The motivations behind this work lay in the widespread availability of OGC Web Services inside government organizations and in the technology support to HTML 5 and WebGL standard of the web browsers. This delivers an improved user experience, similar to desktop applications, therefore allowing to augment traditional WebGIS features with a 3D visualization framework.
This work could be seen as an extension of the Cityvu project, started in 2008 with the aim of a plug-in free OGC CityGML viewer. The resulting framework has also been integrated in existing 3DGIS software products and will be made available in the next months.

1. INTRODUCTION

WebGISes are traditionally bounded to a bi-dimensional simplified representation of the reality. Though they have overcome the lack of flexibility and simplicity of traditional desktop clients, a lot of effort is still needed to reach desktop GIS functionalities, like 3D visualization.
3D modeling and visualization have a variety of applications in geography, urban studies and other fields. On the other hand such models often require to render significant amounts of three dimensional geospatial data, so being very demanding in terms of computing capability and memory usage. In order to efficiently manage large scale data rendering and reach a reasonable compromise between quality and performances some optimizations are needed.
While working in a 2D space, most current WebGISes are leveraging open source technologies and are standard based. Open Geospatial Consortium's work on standardization has, in fact, delivered a great opportunity about interoperability and web services like OGC WMS and OGC WFS are very common nowadays.

1.1 The evolution of the web browsers

Empowered by increased browser's capabilities, currently developed WebGISes are very promising and able to replace traditional desktop GISes for a wide range of applications.
While WebGISes advantages are well known, a lot of work needs to be done about 3D data visualization over the Web.
During the past years, browser technology has been updated to support HTML 5 and CSS 3 in order to allow great user experience, similar to desktop applications. WebGL extensions support permits the development of a complete hardware accelerated 3D engine running in the browser without any third party plugin.

1.2 Research motivations and objectives

Following our products development, we managed to achieve a good experience upon WebGISes and we learned that many, if not all, organizations are able to deploy, or have deployed, their spatial data using OGC Web Services, especially using OGC WMS and WFS.
At the same time it is now possible to leverage the latest browser technologies in order to augment traditional WebGIS features with a 3D visualization framework.
Therefore we started a project with the aim to design a streaming framework for the visualization of three dimensional large-scale geospatial data over the web with real-time generation of features, based on OGC Web Services.

1.3 Extending Cityvu

Employing this approach, a 3D scene is incrementally built and dynamically updated run-time, taking into account the movements of the camera and its field of view. To effectively and efficiently achieve this behavior, proper mechanisms of tiling and caching have been implemented.
The framework implementation focuses on textured terrain and buildings streaming. Despite the scope limitation, the defined streaming paradigm has general validity and can be applied to more complex 3D environments.
The addition of other features on top of the existing ones is straightforward and does not imply substantial modifications to the framework.
In order to make the framework standard compliant and platform independent, it has been designed to work with WMS and WFS OGC web services and the widely adopted web-based approach has been chosen.
As a result, any WebGL compliant browser can run web applications built on top of this framework without the use of

plug-ins or additional software: this approach could be seen as an extension of the Cityvu project, started in 2008 with the aim of a plug-in free CityGML viewer.

2. DEVELOPMENT PLATFORM

The framework's main aim is to achieve 3D visualization and querying with good performance even in low-mid end computers while keeping ease to develop, portability and wide browser support.

Our development platform is bounded to available browser technology. In order to achieve hardware accelerated 3D support in browsers without any plug-ins, two key components of HTML 5 must be supported: WebGL and JavaScript.

Portability and ease of development have been enhanced using existing foundation frameworks, which allows faster development and a good level of abstraction for each component.

2.1 Open source 3D engine

WebGL is a very low-level API, so even the most trivial task requires a considerable amount of lines of code to be accomplished.

For this reason, developing over native WebGL API is not the best choice to build complex 3D applications and the adoption of a 3D engine is a mandatory task.

A 3D engine is designed to hide most of low-level details, and to provide ready-to-use high-level primitives for both basic and advanced functionality; a good engine has to achieve a reasonable level of abstraction, while allowing the developer to direct access lower-level capabilities if needed.

We carried out a software selection among existing open source WebGL engines, evaluating technical features, browser independence, scene graph support (in order to manage complex 3D scenes efficiently), ease to develop, community support and documentation.

We found Three.js as the most complete WebGL engine available, which perfectly fits the project needs.

Three.js aims to create a lightweight JavaScript 3D engine with a very low level of complexity. The engine provides an interface that completely hides the underlying rendering mechanisms, so that the same code can be ported on multiple renderers. Three.js is based on a scene-graph approach, provides a complete light system, while keeping access to fragment shaders in order to achieve advanced lighting, implements a complete materials library and is able to load models from a wide range of sources.

2.2 Enterprise level JavaScript framework

The main purpose of a JavaScript framework is to extend and improve native JavaScript functionality, available to the web browser.

Most of JavaScript frameworks are also designed to hide the differences between browser-specific implementations, thus providing a cross-browser programming environment.

In order to meet project's needs, a JavaScript framework must be browser independent, modular, easy to use and quickly adoptable, while being well supported and documented.

The toolkit must also include support for:

- DOM manipulation;
- Events handling;
- Asynchronous programming (deferred, promises, …);
- Class based programming.

We evaluated the most widely used and valued JavaScript frameworks and Dojo Toolkit has shown to be the best fit for the project.

Dojo is a modular JavaScript framework designed to ease the rapid development of cross-platform, JavaScript/AJAX-based applications and web sites. Modularity is one of the most representative characteristics of Dojo as full support for Asynchronous Module Definition (AMD) has been introduced in the latest releases.

AMD allows to selectively load modules at startup, thus eliminating the need to load the entire toolkit. Moreover, it is possible to define and load custom modules, and any third-party AMD-compliant module, while keeping compatibility with traditional JavaScript.

Dojo Toolkit is developed and supported according to an enterprise approach, featuring namespace extensive support and Java-like class structure.

3. SOFTWARE DESIGN

The framework has been designed using UML modeling and requirements analysis has been carried out according the FURPS+ model, focusing on functionality, usability, reliability, performance and supportability.

3.1 User requirements

The project's user requirements could be summarized as follows.

Framework based applications should be able to:

- run on multiple platforms and environments (desktop and mobile);
- visualize 3D geospatial data;
- load, parse and create a viewable model using 2D data with attributes;
- allow mouse picking in the scene;
- show information about scene objects;
- show general information about the scene;
- support navigation within the scene;
- support informative layers selection;

3.2 Software requirements

Software requirements specification is a complete description of the whole system and could be seen as an extension of the user requirements with nonfunctional elements, which impose constraints on the design or on the implementation. The following requirements have to be added to the previous ones.

Framework based applications should be able to:

- run inside the web browser without installing any specific client software;
- support large data set streaming;
- generate terrain using 2.5D data;
- generate buildings using footprints;
- load and map textures on model elements;
- load existing 3D models with a geospatial emanation point;
- load existing geospatial 3D models;
- provide an extension mechanism for new feature types;
- support custom geographic projections;
- use hardware acceleration;
- supply default materials if not present in the model;

3.3 Architectural design

The framework architecture has been designed in order to allow fast extendibility of geographic features support and it is structured with three main components: Foundation, Scene and Query.
Foundation component is the framework core component. It manages the ellipsoid model and the geospatial projection, supporting a wide range of extensible projection types. Camera viewport is integrated in the foundation component within Tile Manager, which manages the currently loaded set of map tiles and supports dynamic level of details with a Quad Tree backend.
On the top of the Foundation, the Scene component orchestrates external data loading through the Layer Manager. The framework supports data loading from OGC WMS and OCG WFS services with on the fly re-projection support, thanks to the embedded proj4 library.
A layer abstraction is used to allow ease of extension for custom features. The framework currently implements Terrain and Building layers with streaming support. Terrain is streamed using tiled height maps via WFS and buildings are generated using WFS data through on the fly extrusion. An undefined number of data layers is allowed on the top of the Tile Manager.
The third main component of the framework is the Query component, which allows external data integration and querying.

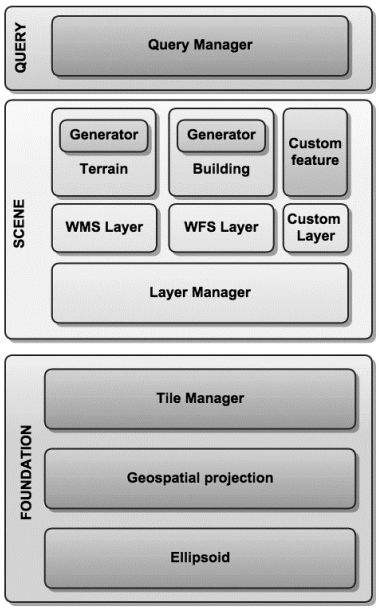

Figure 1. Framework architecture with implemented layers

4. FIRST RESULTS

The result of the development process is a fully functional 3D framework with basic navigation features.
This viewer implements the designed streaming strategy with excellent results in terms of performances (50 - 60 frames per second during navigation with occasional drops to 30 - 40 frames per second while loading) and memory usage (20 - 30 MB).
All requirements have been fulfilled, performance seems more than adequate but needs additional testing in existing scenarios.
3DGIS has planned a wide adoption of the developed framework into its GIS product, being able to extend traditional WebGIS with a feasible 3D visualization tool.

Some examples of the framework are illustrated in the following figures.

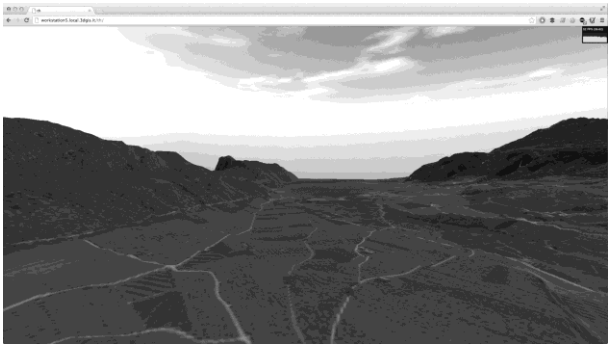

Figure 2. Terrain visualization.

Figure 3. External models and vegetation support.

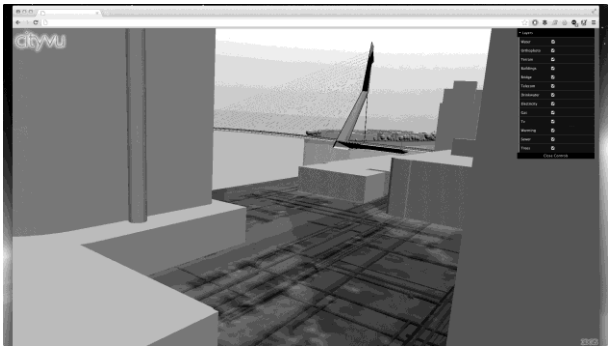

Figure 4. Underground pipes visualization.

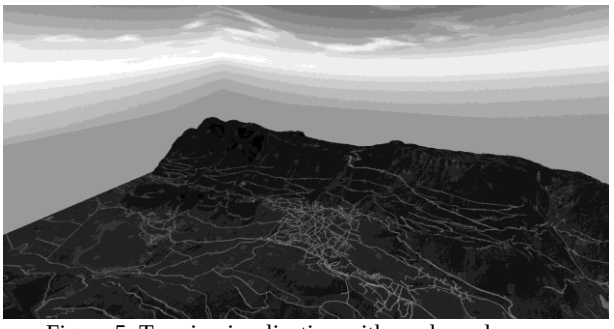

Figure 5. Terrain visualization with road overlay.

5. FUTURE WORK

The framework has been designed to be a solid basis which can be used to develop customized software solutions, built on specific needs.
The developed software is a fully functional and complete streaming system for three dimensional geospatial data, which could be extended with custom feature types.

We envisage its usage for LIDAR data visualization and measurement, underground infrastructure visualization and urban project visualization.
Future developments may also include OGC CityGML support with a software server side companion, extended mobile support and semantic texturing.

6. REFERENCES

Eduard Roccatello, Massimo Rumor, 2009. Design and development of a visualization tool for 3D geospatial data in CityGML format. CRC Press, UDMS Annual, 2009. ISBN 978-0415556422

Aaftab Munshi, Dan Ginsburg, and Dave Shreiner. OpenGL ES 2.0 Programming Guide. Addison-Wesley, 2008. ISBN 978-0321502797.

Khronos Group. WebGL 1.0 specification, February 2011. URL https://www. khronos.org/registry/webgl/specs/1.0/.

Open Geospatial Consortium (OGC). WFS Standard Specification, May 2005.
URL http://portal.opengeospatial.org/files/?artifact_id=8339.

Open Geospatial Consortium (OGC). WMS Standard Specification, March 2006.
URL http://portal.opengeospatial.org/files/?artifact_id=14416.

Dojo Toolkit website.
URL http://dojotoolkit.org.

Three.js website.
URL http://threejs.org.