Evolution of Geospatial Web Services: Proving 3D Scenes for Urban Management and Planning

Nuno Oliveira[1] and Jorge Gustavo Rocha[2]

[1] PT Inovação, Aveiro, Portugal
[2] Universidade do Minho, Braga, Portugal
email corresponding author: jgr@di.uminho.pt

Abstract

The 3D city models play a major role in urban management. Several commercial solutions are being provided by major software vendors with limited success. Major improvements must be made at several levels, from data capturing to visualization, to further improve 3D usage. Standards are necessary to support the integration of several different tools from different providers, to fully support 3D. In this paper we describe our first experience with the open source implementation of the Web 3D Service (W3DS) proposed standard on Geoserver. We want to assess W3DS functionality in a real case scenario and its interoperability with other tools. The W3DS was used within the context of city management of telecommunications infrastructures. Some remarks are presented from the lessons learned.

1. Introduction

Massive real world 3D data, from landscape models to detailed indoor textured models, are becoming available. To take advantage of this data, scientific contributions are necessary to be incorporated at all levels of the existing GIS software stack.

A recent technology development initiated by the Mozilla Foundation might be the key to massive visualization of 3D data: WebGL. Its version 1.0 was released in March 2011. By mixing JavaScript code and shaders (GPU code), sophisticated image processing effects and dynamic physics can be used to provide amazing graphics and sounds directly in the browser.

To take advantage of this support on the client side, we need to consider upstream 3D map services in the GIS software stack. These 3D services will mediate data exchange between the large 3D GIS databases and different client types.

Two service oriented approaches to 3D portrayal are being considered by the OGC. The Web View Service (WVS) supports an image based approach [6]. The WVS client requests 3D rendered images from the WVS server. The other approach is followed by the W3DS and comprises the 3D geometries and textures being rendered on the client side. The W3DS server hosts and manages all requests, but does not render any images. These two approaches are suitable for different client types. Thin clients would prefer already prepared 3D imagery, while clients with more computational resources can provide more flexible and powerful visualizations.

In this paper we focus on the usage of the W3DS standard in the context of city management, aiming to provide some remarks about the usage of the service.

We start by reviewing the state of the art of 3D usage in the GIS and web realms. Afterwards we describe the W3DS specification since it is not widely known. Next, we present the case study. Its implementation was quite demanding because lots of pre-processing was necessary to provide the first production instance of the service. The usage of two different clients is discussed. We conclude by highlighting some remarks about the W3DS service.

2. 3D usage in GIS

3D GIS is clearly not a new concept. In 1993 [3] presents a 3D GIS that uses CAD models to represent the 3D entities and the DTM. The models had three different kinds of approximation, the first two are used to index and accelerate the rendering process and the last one is a detailed representation. In 1997 [17] describes one of the first 3D Web GIS. The HTML pages were produced dynamically and the 3D data was directly retrieved from the database using SQL. The 3D scenes produced are encoded using the Virtual Reality Markup Language (VRML) which can be interpreted by the browser VRML plug-in. The reference [4] gathers some interesting publications about 3D GIS that can be seen as the state of the art of 1990s.

Most of the recent works focus on city administration, which has become one of the top use cases for 3D GIS [7, 16, 11, 10]. OSM-3D is one example of such use cases. Its main objective is to provide a 3D view of OpenStreetMap data integrated with the elevation data of the SRTM. Its implementation is made on top of OGC standards, including W3DS for 3D visualization. The reference [5] makes an overview of the current state of OSM-3D in Germany and provides a good discussion on the generation of 3D building models.

2.1. Web 3D Evolution

VRML was the first web based 3D format, released in 1995 and ISO certified in 1997. The main goal of VRML was to provide a way of representing 3D virtual worlds that can be integrated onto web pages. A VRML scene is composed of geometric primitives like points, segments and polygons. The scene may also include multimedia content such as hyperlinks, images, sounds and videos. The aspect can be customized using lights effects and by defining material properties. VRML scenes can be explored in desktop software or in web browsers, using a compatible plug-in. The reference [13] makes a good overview of the format.

In 2001, the Web3D Consortium, which has become the main supporter of VRML, releases the X3D format a XML encoding version of VRML [2]. The XML based encoding of X3D makes it more suitable for native integration in HTML pages. X3D also adds new features like the support of shaders, better event handling, new geometric primitives and others short cuts for 3D rendering. X3D brings up the concept of working groups, their job is to extend X3D to the custom support of certain areas, like medicine, GIS and CAD. The GIS working group have provided X3D with the capability to natively support the needs of GIS applications. The main features are the full support of georeferenced coordinates and custom events for geographical scenes.

Even if at this time VRML and X3D are the most used web based 3D formats, their use has decreased significantly compared to ten years ago.

When VRML was released everyone tried to make use of it, quickly we saw the appearance of 3D web content everywhere. Some companies invested large quantities of money to shift their websites to 3D. The same thing happened to GIS applications, we have observed a massive jump from 2D to 3D, companies and governments have even start buying 3D georeferenced data. The problem was that technology didn’t follow that movement. Computers with the capability of rendering complex 3D scenes at acceptable frame rates were not common and those that existed were too expensive. With the poor quality offered by 3D web and once the fad of a third dimension had worn off, people started looking again at a 2D web.

Around 2009, WebGL appeared and the doors to a 3D web were definitively opened [12]. WebGL specification is based on OpenGL ES 2. Even if it is only a draft, it has already been implemented by the majors web browsers and plug-ins have been provided for those that don’t natively support WebGL [9]. WebGL gives us the possibility to use 3D hardware acceleration from the JavaScript of web pages, like OpenGL does for desktop 3D applications. Its integration with HTML5 gives the possibility of directly embedding complex interactive 3D scenes on web pages.

Recently a new web based 3D format is being adopted: XML3D. This is the only major web 3D format that is not supported by the Web3D Consortium, however it is a candidate to become a WC3 standard. Unlike the others formats, the main goal of XML3D is to be an extension of the HTML5 specification [14]. On the other side, the XML3D definition is based on other successful standards of W3C like HTML, DOM and CSS. All interactions with the 3D scenes are made using the web standard route, i.e. using DOM events and JavaScript. XML3D is independent of the 3D rendering API used, in [15] the authors use a modified version of the Chromium Browser that uses OpenGL.

3. Web 3D Service

The W3DS is a portrayal service proposal for three-dimensional spatial data. The first proposal was presented back in 2005 by Kolbe and Quadt. Since then, some improvements have been integrated. In 2009, version 0.4.0 was accepted as a public discussion paper by the OGC. Afterwards, a version 0.4.1 was rewritten. This is the last version available, and it dates from 2010 [8].

The W3DS service delivers scenes, which are composed of display elements representing real world features. It does not provide the raw spatial data with attributes, like the Web Feature Service (WFS) service does. It only provides a view over the data, according to, for example, the level of detail.

It does not provide rendered images, like WMS does. It filters the data to be delivered according to several parameters, like a bounding box, but the result will be a graph of nodes with properties attached to each node, like shapes, materials and geometric transformations.

This graph must be handled by the client. W3DS clients must implement the necessary logic to take advantage of the W3DS operations. Typically, clients will continuously request scenes from the service, trying to minimize the data delivered to the client while providing the best user experience.

3.1. W3DS Operations

Like other OGC services, information about the service can be retrieved using the GetCapabilities operation. Two additional operations that return information about features and their attributes are provided: GetFeatureInfo and GetLayerInfo.

Two operations are provided to return 3D data: GetScene and GetTile. These two operations differ essentially in how the features are selected. GetScene allows the definition of an arbitrary rectangular box to spatially filter the features composing the scene returned to the client. GetTile returns a scene on-the-fly formed by features within a specific delimited cell, within a well-defined grid.

Interactive scenes with terrain and relevant 3D geographical features will result from multiple GetScene and GetTile requests to the service. These might be mostly called operations. For these two operations, we show how to call them with the most common parameters.

GetScene. *The GetScene operation generates a 3D scene from the available data, according to several parameters. This is the most typical operation of a W3DS service. In practice, not all the parameters are necessary. Typical requests have the following syntax:*

```
http://3dwebgis.di.uminho.pt/geoserver3D/w3ds?
VERSION=0.4.0&SERVICE=w3ds&REQUEST=GetScene&
CRS=EPSG:4326&FORMAT=text/html&
LAYERS=buildings_3d,dem_3d&
BOUNDINGBOX=-8.301200,41.437741,-8.294825,41.444161&
STYLES=buildings_by_type,dem_texture_igp
```

The previous request would return a HTML encoded file, with inline X3D. In our implementation, the scene encoding the format can be X3D or

HTML5. The HTML5 format is just a container for X3D. With the HTML5 container, the GetScene (or GetTile) result can be viewed directly on a browser supporting WebGL, such as Mozilla Firefox or Google Chrome.

The integration of X3D within HTML5 is provided by the open source X3DOM framework developed by the Fraunhofer Institute [1]. The HTML5 container will return a minimal HTML document with a header that includes:

```
<linkrel="stylesheet"
href="http://www.x3dom.org/x3dom/release/x3dom.css"/>

<script type="text/javascript"
src="http://www.x3dom.org/x3dom/release/x3dom.js">
```

The HTML5 encoding comes in very handy for previewing data directly in the GeoServer administration interface. As soon as the user publishes the layer, it can be previewed from the administration interface.

Layer metadata. *Clients must be aware of the layer parameters, prior to any GetScene request. These parameters are published by GetCapabilities. Information about the layer properties and its LOD settings are stated on the GetCapabilities.*

GetTile. *The GetTile request can be submitted either encoded as KVP in a GET request, or as a XML formatted document, in a POST request. Common GetTile requests, called with key and values pairs, have the following syntax:*

```
http://localhost:9090/geoserver/w3ds?
    VERSION=0.4&SERVICE=w3ds&REQUEST=GetTile&
    CRS=EPSG:27492&FORMAT=model/x3d+xml&
    LAYER=guimaraes&
    TILELEVEL=1&TILEROW=5&TILECOL=
```

Tile metadata. *Clients must know basic service and layer metadata, prior to any request. Tiles can be requested only from layers in the GetCapabilities document tagged with the key:*

```
<w3ds:Tiled>true</w3ds:Tiled>
```

They must also have a <TileSet> definition associated, as illustrated in the following <TileSet> definition.

```
<w3ds:TileSet>
    <ows:Identifier>guimaraes</ows:Identifier>
    <w3ds:CRS>EPSG:27492</w3ds:CRS>
    <w3ds:TileSizes>4000 2000 1000 500
```

```
250</w3ds:TileSizes>
    <w3ds:LowerCorner>-17096
193503</w3ds:LowerCorner>
</w3ds:TileSet>
```

Basically, the <TileSet> provides the lower left corner of the grid, and all available tile sizes. Tile sizes are defined as an ordered list decreasing by tile size. The number of levels available is the length of the list. Levels are numbered from 0, starting at the largest tile size. Obviously, tiles are considered of equal size in both axes, according to the draft specification version 0.4.0. The version 0.4.1 introduced the possibility of supporting non-rectangular tiles.

Styling tiles. *The style optional parameter can specify one or more styles to be used. These must be chosen from the styles indicated in the GetCapabilities response. If no style is provided in the request, the default style associated with the layer will be used. In the implementation of the GetTile described here, we support image textures over the tiles, coming from static files (on the server) or from any WMS service. The exact boundaries of the tiles are appended to the GetMap request against the remote WMS. For example, for the same tile set, we can use one style that employs aerial imagery and another one with Open Street Map rendered images.*

4. Use case

We were challenged to provide a running instance of the W3DS to provide 3D urban models. We used the only known open source W3DS implementation, which is built on top of Geoserver.

4.1. W3DS service deployment

Since W3DS is provided as a community module, it is not included in the Geoserver binary distribution. It has to be compiled from the sources. This can be done with the following steps:

```
git clone git://github.com/geoserver/geoserver.git
geoserver
cd geoserver/src
mvn clean install -DskipTests=true -Pw3ds
```

The previous command generates a geoserver.war file, that must be deployed using Jetty, Tomcat or another application server. By default, the service will be running on port 8080, providing an administrative interface for the service configuration.

4.2. Dataset

The dataset used is composed of the terrain data, the city models (mostly buildings and some city furniture) and telecommunications infrastructure, like poles, cables, junctions, etc. Some of these cables and junctions are below ground level, while others are some meters over ground level (aerial cables). In Fig. 1 we present the architecture of the case study. We will show the necessary steps to feed the server, and then describe the clients used.

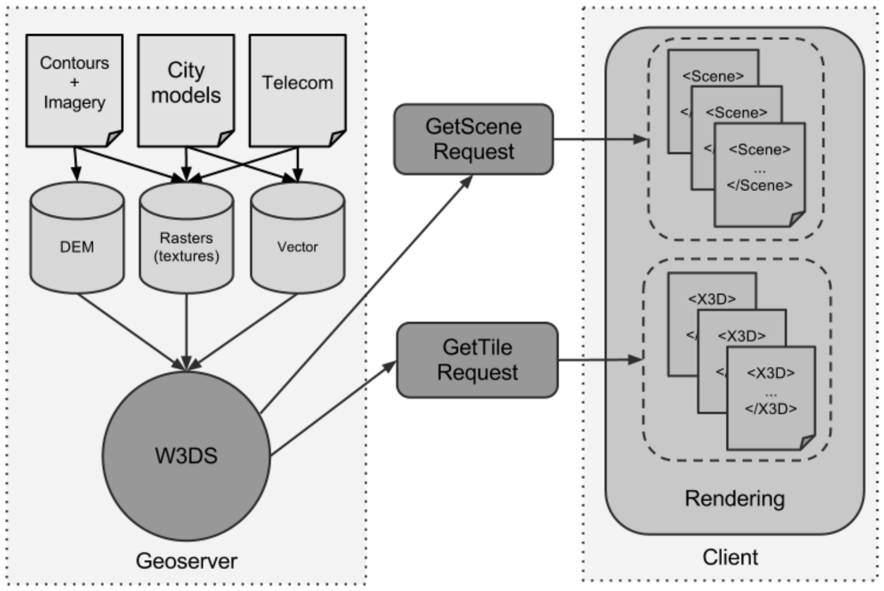

Figure 1. Case study architecture.

4.3. Data preparation

The most time consuming operations were related to data preparation. The terrain data, city models and telecommunications assets were provided by three different sources. While the datasets look pretty well on 2D visualizations, preliminary 3D visualizations showed several problems in the datasets. These problems were identified only when comparing the Z-values provided by the different datasets.

Converting utilities to 3D. The 3D features of our use case can be separated in three categories: terrain, urban models and telecommunications infrastructures. In the solution presented we have prepared a georeferenced 3D model for each element of our dataset and stored it in the database. In PostGIS, those models are stored using primitive 3D geometries like TIN, POLYGON and MULTIPOLYGON. GeoServer catalogue using GeoTools utilities reads the 3D models from the database and converts them to a suitable Java Topology Suite (JTS) geometry.

A 3D model is not only defined by is geometry, it also requires some visual properties that are mainly defined by textures and materials. These additional properties are defined using Style Sheet Descriptors (SLD).

To encode the response of a GetScene or GetTile operation, our W3DS converts the JTS geometries provided by the GeoServer catalogue and the properties defined in the SLD layer to the requested format. The client who receives such scenes has to read the geometry definition and its visual properties to render them. One style property can be a 3D detailed model. 3D models were produced for all the equipment using Blender, an open source 3D authoring tool. The models became a style property of each equipment type. The same models are available in two formats: Collada and OpenSceneGraph file format.

Tiling. *The usual way for accessing spatial data in the Web Map Service (WMS) service, for example, is to request one or more layers with the desired styles, within a certain bounding box. While this is the most flexible way to get rendered maps, in some applications Web Map Tile Service (WMTS) are preferred over WMS when performance is important. When data does not often change tiles can be prepared offline, freeing the WMS from rendering the same area over and over again. Tiles are rendered once and served many times. WMTS trades the flexibility of custom rendered maps for fast delivery of fixed tiles.*

Delivering fixed sets of tiles also enables simple scalability mechanisms. Tile caches can be distributed among different servers.

While the WMTS provides a complementary approach to the WMS for tiling maps, the W3DS includes both the flexibility of arbitrary scene creation though the GetScene operation and fast tile delivery through the GetTile operation.

W3DS clients should be able to request tiles and display them directly without any additional 3D manipulation or geographical positioning correction.

To prepare the terrain data, we develop “Tool For 3D Terrains” (TF3T), a tool build on top of Computational Geometry Algorithms Library (CGAL) and Geospatial Data Abstraction (GDAL) libraries. TF3T allows us to execute most of the operations related to 3D terrains, from their creation to the positioning of features over them. Using the GDAL supported output formats, TF3T allows us to feed W3DS with 3D tiled terrains at different levels of detail.

Tile storage. *We chose to store all tiles in the spatial database after they have been calculated by the TF3T tool. Each tile became a row in the database and can be retrieved using its level, row and column number as keys. Alternatively, tiles could be stored as files, and served from the file system. The hierarchical organization of the file system can use the three different keys (level, row and column) to organize the tiles in folders within folders.*

4.4. W3DS visualization

In the previews section we describe the necessary data preparation to get the W3DS service up and running, serving 3D scenes and tiles.

To take advantage of this service, two different clients were developed to test and assess W3DS server functionality.

Simultaneous 2D and 3D visualization using Google Earth. *The first client developed is a web based visualization service, featuring two synchronized views: 2D and 3D. The 2D view is a typical map based visualization of the features. It is written using the OpenLayers library. The 3D view is based on the Google Earth plugin. The 3D features are overlaid on this globe. Features are requested from the W3DS in the KML format. The KML file format allows us to reference the Collada models, and the plug-in renders the scene as expected. In Fig. 2 several telecommunications infrastructures rendered on top of the GoogleEarth plugin are shown.*

In this first client, we take advantage of all existing services for the same layer to create both a 2D and 3D visualization. Each zoom or pan event in a view, injects the same event on the other view, to keep both synchronized.

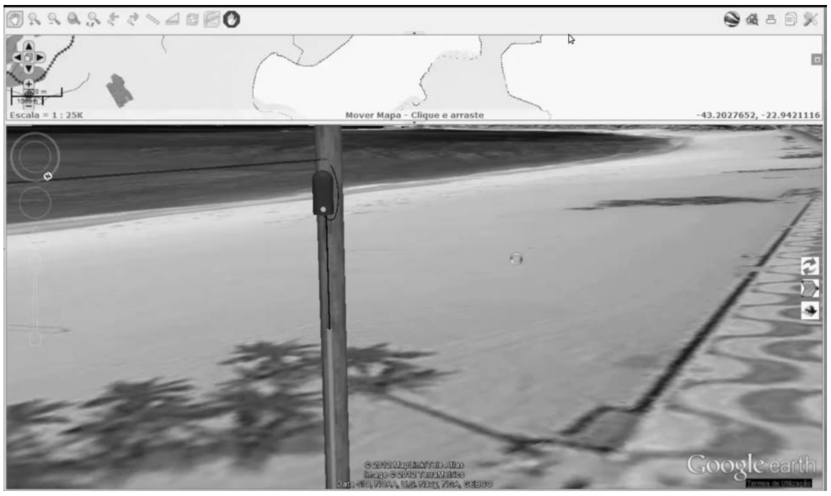

Figure 2 W3DS requests encoded in KML and rendered on top of GoogleEarth plugin.

Alternative visualization using OsgEarth. *Virtual globes are natural visualizations of geographic information. While we used the GoogleEarth plugin for the previous client, in this second client we choose to use OsgEarth which is an open source implementation of a virtual globe. Since it is open source, we can develop specific functionalities from the project requirements, which we were unable to implement with GoogleEarth.*

Since the case study includes several underground telecommunications infrastructures, we need to show and follow underground pathways. We also need to implement our own logic to request the scenes from the server, managing the requests in our own way. That’s why we chose to use OsgEarth.

To provide 3D scenes to OsgEarth we provide additional SLD styling capabilities more adequate to OsgEarth’s needs. No other changes were necessary. Since all changes are restricted to style definition, we can use the same data with a style mode adequate for Google Earth or OsgEarth.

While the visualization in OsgEarth is quite similar to Google Earth, the usage of another client demonstrated how important it is to keep the contents (geometry data) separate from the presentation (styles).

5. Concluding remarks

Supporting 3D applications might not be as easy as we would like. To change the workflow to 3D, both data and tools must address the challenge.

From our experience, moving to 3D has a major impact on the data side. It might be quite challenging, but it is an opportunity to further validate and improve consistency among existing datasets. In our case study, many consistency problems were only revealed in the original 2D datasets when merged with 3D sources.

Elevation models are split into several tiles with different LOD. A common client will issue many GetTile requests. These GetTile requests can be very expensive, depending on terrain data resolution. But terrain data is not updated very often. Based on our experience, we need a very efficient and scalable cache system for high resolution terrain. Tiles should be prepared in advance.

Like terrain, urban models can be very expensive to handle. City models also do not change often. W3DS does not provide a specific operation that lets us retrieve a single 3D model as GetTile does for tiled terrains. This makes it more difficult to develop an efficient cache system. However, the main formats (X3D, KML and XML3D) used to encode GetScene or GetTile responses give the possibility of referencing external 3D models. Basically, instead of having our 3D models stored in the database and translated to a specific format by W3DS, we only place a reference to that georeferenced model. This functionality can be achieved by extending SLDs to support the inclusion of external 3D models. In that situation the rendering engine on the client side will be responsible for requesting and handling the 3D model. For example, we only create one 3D model for each pole type, as so many poles exist in the city. Then, on the client side, only one model is downloaded for each pole type. On the server side, instead of storing thousands of pole models, no model at all is stored in the database.

For the application domain used, since there are thousands of elements from a limited number of different types, the use of this strategy greatly improved performance. This improves the performance and storage space on the server side, minimizes data transferred between the server and the client, and also improves efficiency on the client side.

References

1. Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. X3DOM - A DOM-based HTML5 X3D Integration Model. 2009.
2. Don Brutzman and Leonard Daly. X3D: extensible 3D graphics for Web authors. 2007.
3. Béatrix Cambray. Three-dimensional (3D) modeling in a geographical database. 1993.
4. Alessandro Carosio. La troiséme dimension dans les systémes d'information geographique et la mensuration officielle, 1999.
5. Marcus Goetz and Alexander Zipf. OpenStreetMap in 3D - Detailed Insights on the Current Situation in Germany. 2012.
6. Benjamin Hagedorn. Web View Service Discussion Paper, 2010.
7. Georg Held, Alias Abdul-Rahman, and Siyka Zlatanova. Web 3D GIS for urban environments. 2001.
8. Thomas Kolbe and Arne Schilling. Draft for Candidate OpenGIS Web 3D Service Interface Standard, 2010.
9. Chris Marrin. Webgl specification, 2013.
10. J. Moser, F. Albrecht, and B. Kosar. Beyond visualization - 3D GIS analyses for virtual city models. 2010.
11. Masahiko Murata. 3D-GIS Application for urban planning based on 3D city model. 2005.
12. Sixto Ortiz. Is 3D Finally Ready for the Web? 2010.
13. Mark Pesce. VRML: Browsing and Building Cyberspace. 1995.
14. Kristian Sons, Felix Klein, Dmitri Rubinstein, Sergiy Byelozyorov, and Philipp Slusallek. XML3D: interactive 3D graphics for the web. 2010.
15. Kristian Sons and Philipp Slusallek. Demo: XML3D - Interactive 3D Graphics for the Web. 2012.
16. Peter Zeile, Ralph Schildwächter, Tony Poesch, and Pierre Wettels. Production of virtual 3D city models from geodata and visualization with 3D game engines. A Case Study from the UNESCO World Heritage City of Bamberg Problem Status - Starting Point. 2004.
17. Siyka Zlatanova. VRML for 3D GIS. 1997.