Web 3D Service Implementation

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

[1] PT Inovação SA, Aveiro, Portugal
nuno-miguel-oliveira@ptinovacao.pt
[2] Universidade do Minho, Braga, Portugal
jgr@di.uminho.pt

Abstract. In this paper we describe an open source implementing of the Web 3D Service (W3DS) based on the OGC's draft proposal. The implementation was developed on top of the open source java-based map server GeoServer, as a community module. With an open source implementation available, test beds can be promoted to better know the strengths and limitations of the current proposal. Without practical interoperability assessments demonstrated, W3DS barely become a 3D standard in urban management applications.

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 Geographical Information Systems (GIS) software stack.

Technology is enhanced every day, but new paradigms supported by technology enhancements only appear from time to time. For example, the client side AJAX support, enabling asynchronous web applications, was a key feature in the explosion of web mapping applications on the web. Software libraries, like OpenLayers or equivalent, taking advantage of AJAX support with additional supporting logic, raised the base level from which all new web mapping applications were built.

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. While HTML5 is still nothing more than the next version of HTML, with no real technological breakthrough, the WebGL is the 3D key changer. WebGL enables direct access to the GPU within 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 Open Geospatial Consortium (OGC). The Web View Service (WVS) supports an image based approach. The WVS client requests 3D rendered images from the WVS server. In the other approach, followed by the W3DS, the 3D geometries and textures are 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 types of clients. 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 development and discussion of the W3DS proposed standard [8]. We contributed with an open source implementation that can be tested in several scenarios, to provide valuable feedback about its interoperability.

We start by reviewing 3D usage in the GIS realm. Afterwards we describe the W3DS specification, introducing its operations. The implementation is described. The W3DS architecture is shown, and the major modules are briefly described. Next, we describe a very simple scenario that can be reproduced to get W3DS up and running. Finally, some conclusions are presented followed by a brief outlook.

2 State of the Art

In 1993 [4] presents a 3D GIS that use 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 [16] 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 [5] gathers some interesting publications about 3D GIS that can be seen as the state of the art of the 1990s.

Most of the recent works focus on city administration, which have become one of the top use cases for 3D GIS [7, 15, 11, 10]. OSM-3D is one example of 3D GIS projects. Its main objective is to provide a 3D view of OpenStreetMap data integrated with the elevation data of the Shuttle Radar Topography Mission (SRTM). Its implementation is made on top of OGC standards, including W3DS for 3D visualization. The reference [6] makes an overview of the current state of OSM-3D in Germany and provides a good discussion about 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 give a way to represent 3D virtual worlds that can be integrated on web pages. A VRML scene is composed of geometric primitives like points, segments and polygons. The scene may also include multimedia content like hyperlinks, images, sounds and videos. The aspect can be customized using lights effects and defining materials properties. VRML scenes can be explored in desktop software or in web browsers, using some 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 Extensible 3D (X3D) format, an XML encoding version of VRML [3]. 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 custom support of certain areas, like medicine, GIS and Computer Aided Design (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 stay the most used web based 3D formats, their use has decreased significantly compared to ten years ago. When VRML was released everyone has tried to make use of it, quickly we have seen the appearance of 3D web content everywhere. Some companies have invested large quantities of money to shift their websites to 3D. The same thing happened with GIS applications. Companies and governments have even start buying 3D georeferenced data. But the technology was not already available. Computers with the capability of rendering complex 3D scenes at acceptable frames rates were not common and those that existed were too expensive. With the poor quality offered by 3D web and once the new sensation of a third dimension had worn off, people started looking again at a 2D web.

Around 2009 WebGL appears and the doors to a 3D web are 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 to directly embed 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 to HTML5 specification [14]. The authors claim that even if using X3D or VRML we can integrate 3D content in a web page, the separation of the two concepts is well defined. On the other side, XML3D definition is based on other successful standards of W3C like HTML, DOM and CSS. All the interactions with the 3D scenes are made using the web standard path, i.e. using DOM events and JavaScript. XML3D is independent of the 3D rendering API used, in [14] the authors use a modified version of Chromium Browser that uses OpenGL, however the top rendering technology for XML3D is still WebGL.

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 were integrated. In 2009, version 0.4 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.

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 Web Map Service (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 of display elements 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, tying 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, the supported operations, available layers and their properties, 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 to compose 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.

All five proposed operations, tagged as mandatory or optional are listed in Tab. 1.

Interactive scenes with terrain and relevant 3D geographic 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.

Table 1. W3DS operations

Operation	*Use*
GetCapabilities	mandatory
GetScene	mandatory
GetFeatureInfo	optional
GetLayerInfo	optional
GetTile	optional

Table 2. GetScene parameters

Operation	*Definition*	*Use*
crs	CRS of the returned scene	Mandatory
boundingBox	Bounding rectangle surrounding selected dataset, in available CRS	Mandatory
minHeight	Vertical lower limit for boundingBox selection criteria	Optional
maxHeight	Vertical upper limit for boundingBox selection criteria	Optional
spatialSelection	Indicates method of selecting objects with boundingBox	Optional
format	Format encoding of the scene	Optional
layers	List of layers to retrieve the data from	Optional
styles	List of server styles to be applied to the layers	Optional
lods	List of LODs requested for the layer	Optional
lodSelection	Indicates method for selecting LODs	Optional
time	Date and time	Optional
offset	Offset vector which shall be applied to the scene	Optional
exceptions	Format of exceptions	Optional
background	Identifier of the background to be used	Optional
light	Add light source	Optional
viewpoints	Add viewpoints to choose from	Optional

GetScene. The GetScene operation composed a 3D scene from the available data, according to several parameters. This is the most typical operation of a W3DS service. Table 2 describes GetScene parameters (version, service and request are omitted for simplicity). This extensive list of parameters gives the necessary flexibility to request the desired scene.

In practice, not all parameters are necessary. Clients must be aware of the adequated parameters, prior to any GetScene request. Information about the layer properties and its LOD settings are stated on the GetCapabilities.

GetTile. The GetTile parameters are listed in Tab. 3. These can be submitted either encoded as KVP in a GET request, or as a XML formatted document, in a POST request.

Table 3. GetTile parameters

Operation	*Definition*	*Use*
crs	Coordinate Reference System of the returned tile	Mandatory
layer	Identifier of the layer	Mandatory
format	Tile encoding format	Mandatory
tileLevel	Level of requested tile	Mandatory
tileRow	Row index of requested tile	Mandatory
tileCol	Column index of requested tile	Mandatory
style	Identifiers of server styles to be applied	Optional
exceptions	Format of exceptions	Optional

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=7
```

Tile Metadata. Clients must know basic service and layer metadata, prior to any request. Tiles can be requested only from layers tagged in the GetCapabilities document 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>-17096.156 193503.057</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, accordingly to the draft specification version 0.4.0. The version 0.4.1 introduced the possibility of supporting non-rectangular tiles.

4 Implementation

The implementation of the W3DS service should follow the specification, providing the operations already presented.

There are several OGC compliant open source implementations of web map services, like GeoServer [1], MapServer or Deegree. W3DS ideally should be implemented as a component on top of each of them to become widely available. It should be implemented as an additional component, taking advantage of existing code of these servers to manage the request pool, parsing the requests, handling formats, etc.

We started by developing the W3DS component on top of GeoServer. It has detailed technical documentation, has clear policies regarding new contributions and uses recent development tools. It is written in Java and uses sophisticated technologies making it easier in terms of flexibility, extensibility and maintainability. These three design goals are particularly important in projects like Geoserver, since it implements several different web services and it is used by a large community, there are permanent change requests, requests for new features and new formats, bugs to be fixed, etc. It also has to comply with different versions of the same web service, for example, WMS (1.1.1 and 1.3), WFS (1.0 and 1.1), WCS (1.0 and 1.1).

4.1 Architecture

The GeoServer results from a large open source software stack, taking advantage of other modules, like GeoTools, Maven, Spring Framework, Apache Wicket, and others. The GeoServer itself is made of three types of modules: core modules, extensions and community modules. The core modules provide entry points that are used by extensions and community modules to add extra capabilities to GeoServer. The provided entry points are made available using the proprieties of Java language itself and using some major Java frameworks like Spring and Apache Wicket for example.

The different types of services in GeoServer include WFS, WMS, and WCS, commonly referred to as OWS services. The virtual service OWS can be instantiated to implement any OWS-like service. The W3DS is an instantiation of the OWS service.

Figure 1 provides an overview of the W3DS architecture, in the context of the GeoServer implementation. When a request is sent to GeoServer the responsibility of handling it is delegated to the dispatcher. From the request, the dispatcher identifies the service and the chosen operation, which can be `SERVICE=w3ds&REQUEST=GetScene`.

The dispatcher then calls the key-value pairs (KVP) parser of the selected operation (ie GetSceneKVPReader). The remaining parameters of the request are decoded and an object that contains all the necessary information to execute the selected operation is created (i.e. GetSceneRequest).

The execution phase begins, gathering all the necessary data to build the response. The response is encoded according to the requested format.

4.2 W3DS Components

The W3DS implementation is composed of five major components: types, styles, responses, service and web.

Types. The types component provides the necessary objects to represent the W3DS service domain model.

Styles. Behind the scenes GeoServer uses GeoTools styling support to handle Styled Layer Descriptors (SLD), but GeoServer doesn't provide any entry point to extend styling. The styles component contains our implementation of 3D SLD support, which extends the native SLD with 3D specific issues.

Responses. GeoServer natively give the possibility of registering a response format producer for each combination of service, request and format. The responses component contains the implementation of our supported response formats.

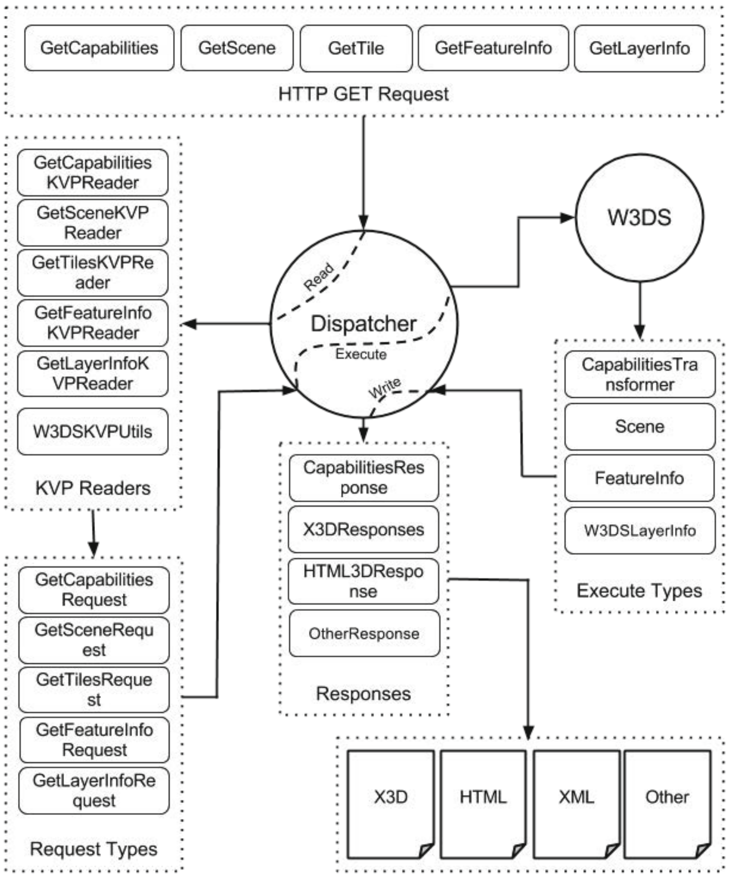

Fig. 1. W3DS Architecture overview

Service. New services need to implement some interfaces and register themselves in the application context to be recognized by GeoServer. The service component contains that implementation.

Web. The web interface of GeoServer is built on top of Apache Wicket. The web component extends the GeoServer web interface to include W3DS specific controls, taking advantage of the existing extension points.

4.3 W3DS Output Formats

The W3DS produces 3D scenes to be consumed by 3D clients. Three formats were considered: X3D, X3DOM and KML. Currently, only X3D and KML are produced, by the corresponding packages x3d and kml. The X3D format produces an X3D document or alternatively an HTML 5 document with the X3D embedded on HTML5. To select the desired output format, clients must use the following format parameter:

- `application/vnd.google-earth.kml`
- `model/x3d+xml`
- `application/x3dom`

The x3d producer was entirely developed from scratch, in the absence of a proper X3D Java library. The kml producer is built on top of the Java API for KML (JAK) library.

The integration of X3D in the HTML5 is provided by the open source X3DOM framework developed by the Fraunhofer Institute [2]. 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 we will show.

5 W3DS in Action

To get W3DS up and running, two steps are required.

First, the GeoServer needs to be compiled with the W3DS module. Since it is a community module, it is not included in the GeoServer binary by default. To compile GeoServer with any community module we need to add it to the build explicitly via profiles (-P). Using Maven, we can download the entire project sources and build it with:

```
git clone git://github.com/geoserver/geoserver.git geoserver
cd geoserver/src
mvn clean install -DskipTests=true -P w3ds
```

The previous Maven command generates a `geoserver.war` file.

It can be deployed using Jetty, Tomcat or another application server. By default, the service will be listened on port 8080. The administrative interface can be reached at `http://localhost:8080/geoserver/`.

After building and starting GeoServer, a new layer and the corresponding 3D style must be added to the configuration to take advantage of the W3DS module. This can be done through the GeoServer web administration interface.

To show an example in action, we describe a very simple scenario. A PostGIS table is required to describe the features and its location. Let's create a simple table `interest_points`, and add some points to it:

```
CREATE TABLE public.interest_points
(
  fid integer NOT NULL,
  wkb_geometry geometry,
  description character varying,
  CONSTRAINT interest_points_pk PRIMARY KEY (fid)
);
insert into interest_points (wkb_geometry, description)
values (st_geometryfromtext('POINT(-16229 199085 140)'),
  'some street furniture');
```

Besides data, we also need a, SLD style. In the following style we define that

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor>
  <NamedLayer>
    <Name>Point Cone</Name>
    <UserStyle>
      <FeatureTypeStyle>
        <Rule>
          <PointSymbolizer>
            <Graphic model="true">
       <href>http://localhost:8080/models/cube.x3d</href>
            </Graphic>
          </PointSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

The x3d model referenced in the SLD document, cube.x3d, can be as simples as the model showed in the following listing.

```
<?xml version="1.0" encoding="utf-8"?>
<X3D xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  version="3.0" profile="Immersive"
  xsd:noNamespaceSchemaLocation=
    "http://www.web3d.org/specifications/x3d-3.0.xsd">
  <Scene>
    <Shape>
      <Appearance>
        <Material diffuseColor="0.8 0.8 0.8"/>
      </Appearance>
      <Box size='16 16 16'/>
    </Shape>
  </Scene>
</X3D>
```

In this scenario, we have a PostGIS table able to capture 3D points; an X3D model and an SLD style that maps the X3D model to each point. After this data preparation step, the W3DS is able to produce 3D scenes. A simple request would be:

```
http://localhost:8080/geoserver/w3ds?
    version=0.4&service=w3ds&request=GetScene&
    crs=EPSG:27492&
    format=model/x3d+html&
    layers=interest_points&
    boundingbox=-16423,199085,-16229,199336&
    styles=point_cube
```

Such a request would produce an HTML document with the scene in X3D that can be directly shown in the browser, as illustrated in Fig. 2. It is possible to see the 3D in the browser because the HTML document also includes a reference to the X3DOM library.

Managing W3DS layers is as simple as managing any other GeoServer layer. It is possible to manage and preview the W3DS layers from the administrative interface.

W3DS also produces KML documents, which is handy for producing scenes that can be explored in Google Earth.

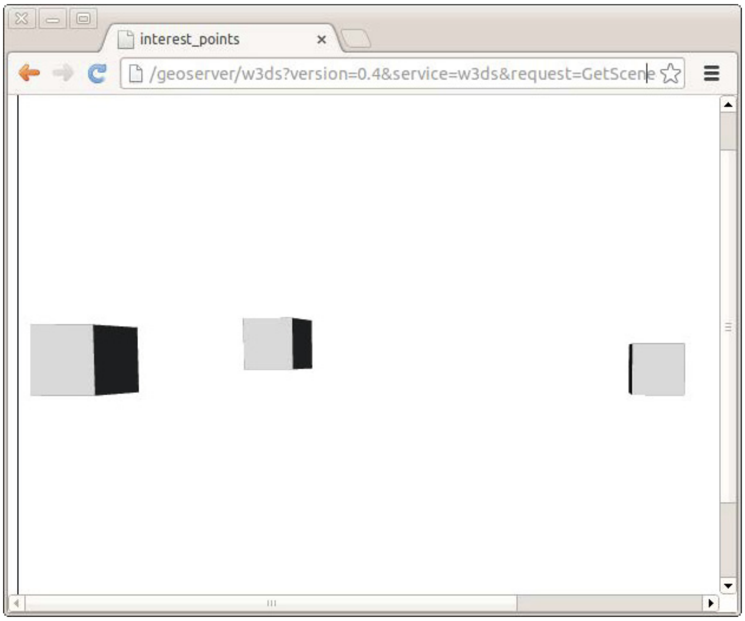

Fig. 2. 3D scene preview directly on the browser

6 Concluding Remarks

While there is much interest in 3D in the urban planning and modelling areas, we still lack a consensus on the approach and a standard, either de jure or de facto, to provide 3D scenes.

In this paper we present one open source implementation of the W3DS draft proposal, as a GeoServer community module. The architecture and the developed modules were discussed. We also showed how a simple layer can be configured and served by W3DS.

The service itself does not provide any urban planning or management oriented facilities. These specific functionalities must be included on the client side, for each use case.

To change the workflow to 3D, both data and tools must address the challenge. While we just addressed the server side tools, 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 different datasets.

The availability of services like W3DS also contributes to the development or improvement of existing 3D clients. OpenLayers is a sophisticated client library for displaying maps on the web. Since OpenLayers is being rewritten to support WebGL, it may use a W3DS service as a native data source. That would improve the development of web based urban management applications.

References

[1] GeoServer, open source Java-based map server, `http://geoserver.org`
[2] Behr, J., Eschler, P., Jung, Y., Zöllner, M.: X3DOM - A DOM-based HTML5 X3D Integration Model (2009)
[3] Brutzman, D., Daly, L.: X3D: extensible 3D graphics for Web authors (2007)
[4] Cambray, B.: Three-dimensional (3D) modeling in a geographical database (1993)
[5] Carosio, A.: La troiséme dimension dans les systémes d'information geographique et la mensuration officielle (1999)
[6] Goetz, M., Zipf, A.: OpenStreetMap in 3D - Detailed Insights on the Current Situation in Germany (2012)
[7] Held, G., Abdul-Rahman, A., Zlatanova, S.: Web 3D GIS for urban environments (2001)
[8] Kolbe, T., Schilling, A.: Draft for Candidate OpenGIS Web 3D Service Interface Standard (2010)
[9] Marrin, C.: Webgl specification (2013)
[10] Moser, J., Albrecht, F., Kosar, B.: Beyond visualization - 3D GIS analyses for virtual city models (2010)
[11] Murata, M.: 3D-GIS Application for urban planning based on 3D city model (2005)
[12] Ortiz, S.: Is 3D Finally Ready for the Web? (2010)
[13] Pesce, M.: VRML: Browsing and Building Cyberspace (1995)
[14] Sons, K., Klein, F., Rubinstein, D., Byelozyorov, S., Slusallek, P.: XML3D: interactive 3D graphics for the web (2010)
[15] Zeile, P., Schildwächter, R., Poesch, T., Wettels, P.: 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)
[16] Zlatanova, S.: VRML for 3D GIS (1997)