3D web visualization of huge CityGML models

F. Prandi, F. Devigili, M. Soave, U. Di Staso, R. De Amicis

Fondzione Graphitech, Via alla Cascata 56/c, 28123 Trento, Italy

GraphiTech
delivering tangible innovation

Introduction

- The big offer of raw Geospatial Information are increasing the availability of semantic and geometric 3D models.
- CityGML is an open data model and XML-based standard for the representation of 3D urban objects.
- CityGML is designed to represent 3D city models, but not to present or visualise 3D city models directly, to read the geometry information directly on the cityGML file is inefficient especially considering the dimension of a whole city dataset.

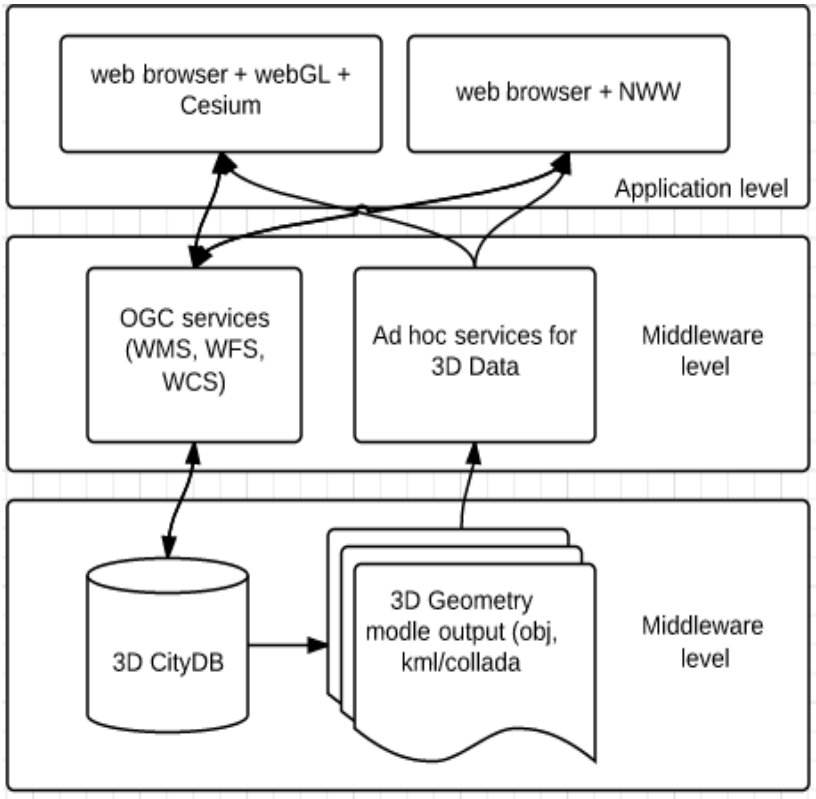
Objective

- Develop and deploy a platform to storing, analyzing and visualizing the 3D city models via web supporting the easily access and visualization to this huge amount of data through a web service.
- First: A server architecture that allows storage and fast retrieval of large amount of data.
- Second: A web client able to visualize in an effective way the data provided by the server.

In this paper, we will compare two different web client system to support the dynamic visualization of 3D city models via web; one based on the Nasa Worl Wind Java applet and the second based on the webGL framework Cesium.

Methodology (1)

The tested solutions are based on the three levels SOA architecture composed by: information level, middleware and application level.



Methodology (2)

- At the lowest level (Information level) the semantic 3D city models are stored on the 3D cityDB. The 3D geometric part of the model will be stored on the information level in the most common 3D format in order to be provided to the application level.
- The middleware level operates like a “bridge” between the information and application levels. The WFS ensure the access to the semantic information contained on the 3D city Model. Another middleware component is devoted to provide the 3D geometric data needed for the effective visualization of the 3D model in the web client.
- The application level consists on the web graphical interface that allows users to access to spatial, thematic and structural information of the semantic 3D city models.

Implementation and Case studies

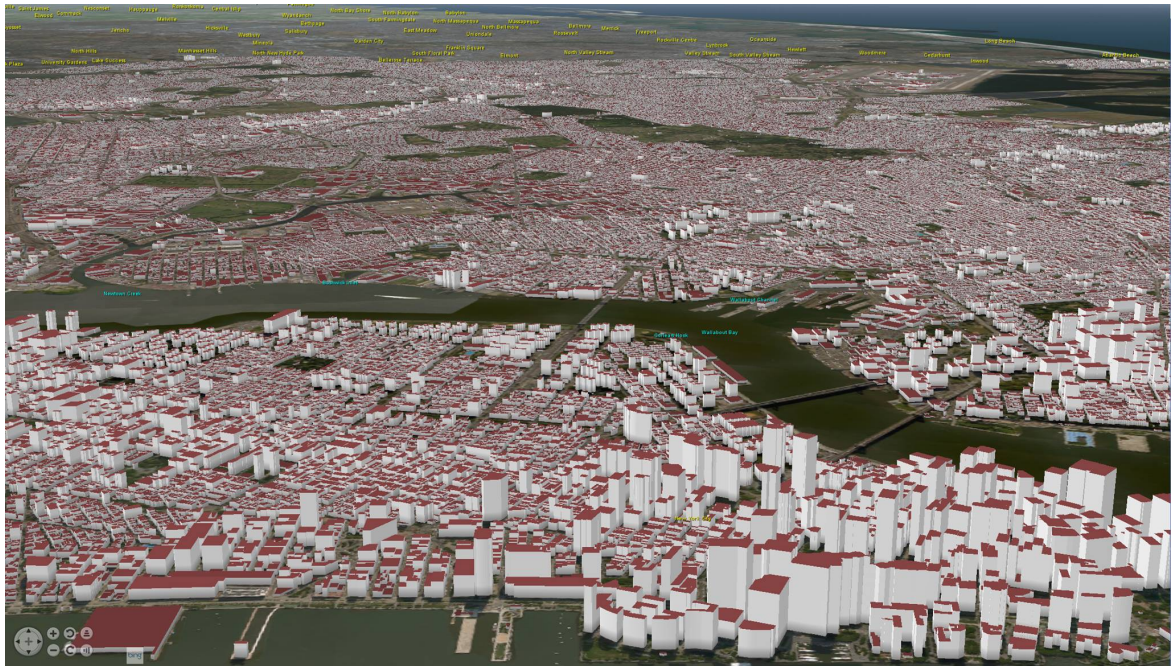
Different case studies using two of the most used, open source, 3D web visualization tools:

- CesiumJS: CesiumJS is a WebGL/Javascript based spinning globe. CesiumJS allows the visualization of geographical data through OGC standards for data interoperability. It also allows the rendering of 3D models based on the glTF data format.
- Nasa Java World Wind (NWW): Nasa Java World Wind is a spinning globe based on Java and JOGL OpenGL wrapper. NWW can be run inside an applet and thus deployed on a web browser. This solution supports most of the OCG standards but does not provide a way to render 3D models natively

Nasa World Wind java applet and custom 3D format

The first case study uses utilizes a Java applet based on Nasa World Wind with an optimized custom rendering engine for 3D buildings.

- CityGML from 3D City DB are loaded in a custom servlet which parse and transform the CityGML file into an OBJ compatible format called C3D.
- This format, optimized for size and parsing speed, contains only hierarchical, geometry and reference information.
- Client can get an object id to query a WFS service to potentially obtain all the information present in the 3D City DB.



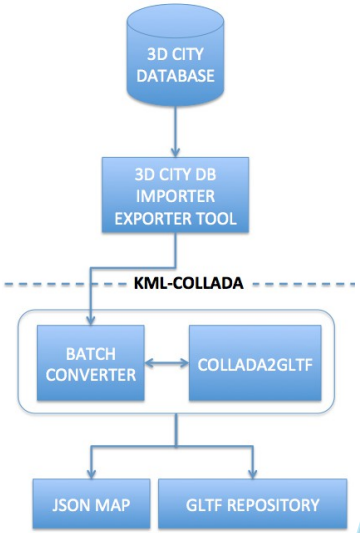
1.700.000 Buildings of New York City GML

Whole Rotterdam LOD2 CityGML

CesiumJS and GLTF format

This case study utilizes the servlet implemented in the previous case but replaces the client with Cesium. A custom geometry loader is implemented with low level Cesium APIs.

- 3D models, exported by the 3D City DB importer/Exporter tool, are converted through a batch script that invokes, for each building, the COLLADA2GLTF component previously described.
- A file-system based collection of glTF 3D models, one for each building involved in the conversion process.
- A JSON map, that will be used by the client in order to ensure the geo-reference of each component stored in the file.



Berlin LOD2 with texture CityGML

Rotterdam LOD2 with texture CityGML

Conclusion

- First Approach (NWW + custom format)
  - Pro
    - Massive rendering of large number of objects;
    - Conservation of geometry semantic properties (CityGML roofs and walls);
  - Cons
    - Java applet (deprecated from chrome);
    - Non standard format;
    - No support for texture.

Conclusion (2)

- Second Approach (CesiumJS + GLTF)
  - Pro
    - Standard format;
    - Textures support;
    - Scalable Pipeline;
    - WebGL cross platform.
  - Cons
    - Slowest;
    - Objects queryable just at building level.