Massive Terrains in CityGML

Kumar, Kavisha

Report: Massive Terrains in CityGML

Geonovum & 3D Geoinformation, TU Delft

Date: 11-11-2016
Version: 1.0

Kavisha Kumar

3D geoinformation
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

Contents

List of Figures

List of Tables

Abbreviations

ADE Application Domain Extension.

AHN2 Actueel Hoogtebestand Nederland (version 2).

CityGML City Geography Markup Language.

COLLADA COLLAborative Design Activity.

CZML Cesium Language.

DTM Digital Terrain Model.

GIS Geographic Information System.

glTF GL Transmission Format.

GML Geography Markup Language.

KML Keyhole Markup Language.

TIFF Tagged Image File Format.

TIN Triangulated Irregular Network.

TMS Tile Map Service.

VRML Virtual Reality Modelling Language.

W3DS Web 3D Service.

WMS Web Map Service.

WMTS Web Map Tile Service.

XML eXtensible Markup Language.

Chapter 1

Storing & visualizing massive terrains

1.1 Introduction

A 3D city model is a digital representation of the geographical objects within a city (Stadler and Kolbe, 2007). At first, 3D city models were mainly used for visualization but with the advancements in geoinformation technologies, they have gained importance in different applications like urban planning (Döllner et al., 2006; Kolbe et al., 2015), 3D cadastre (Stoter et al., 2013; Çağdaş, 2013; Guo et al., 2013), building rooftop solar irradiation estimation (Biljecki et al., 2015a; Eicker et al., 2014), building energy demand estimation (Kaden and Kolbe, 2014; Krüger and Kolbe, 2012), noise mapping (Stoter et al., 2008), population estimation (Biljecki et al., 2016), etc.; see Biljecki et al. (2015b) for an overview. It should be noticed that so far in practice, the applications of 3D city modelling are mostly centred around the buildings; other features, e.g. terrain/relief, vegetation, roads, water bodies, bridges, are often ignored. Here, the main focus is on the storage and dissemination of massive terrains in the context of 3D city models. Generally grids and TINs (Triangulated Irregular Networks) are considered as the basic GIS structures for the representation of terrains. We focus here on TINs.

The 3D GIS standard CityGML (City Geography Markup Language) allows to store terrains as TINs and grids (Groger et al., 2012). The storage as grids in CityGML is simple and is based on GML. The grids can be stored either inline as a finite number of geometric locations (x,y) with elevation values or as a hyperlink to an external file (say a TIFF file) containing the geodata (x,y,elevation). CityGML follows the OGC Simple Feature structure for the storage of TIN geometry (Figure 1.6). Although we see OGC Simple Feature as the current acceptable solution for storing 3D city objects, we say that it is not suitable to efficiently store massive TINs. Firstly, with massive TINs, the datasets become very large, which greatly hinders exchange and dissemination. Secondly, there is very little topological information stored, which prevents us from using the triangles for analysis. The cause of these two problems is that triangles are represented as linear rings, which stores each triangle independently, and moreover repeats several vertices (Figure 1.6). The rest of the limitations of the current solution are discussed in Section 1.3.

As an example, let's consider 3DTOP10NL, the 3D city model of Netherlands (Kadaster, 2015), which covers the whole country, including buildings, terrain, roads, canals, etc. (see Figure 1.1). It is constructed by adding the third dimension from the AHN2 point cloud, obtained from airborne laser-scanners, to the objects in the 2D topographic map TOP10NL (Elberink et al., 2013). Its terrain is a constrained TIN with more than 1 billion triangles (1,156,641,666 to be exact). Storing it with the current solution of CityGML requires around 686 GB of storage space for the terrain geometry. One can imagine that if all the elevation points from the point cloud are used (around 640 billions, thus around 1.3 trillion triangles), then the file size would clearly prevents us from using the dataset in practice. With the increasing size (in terabytes) of these datasets, the biggest challenge lies in their storage, management and dissemination.

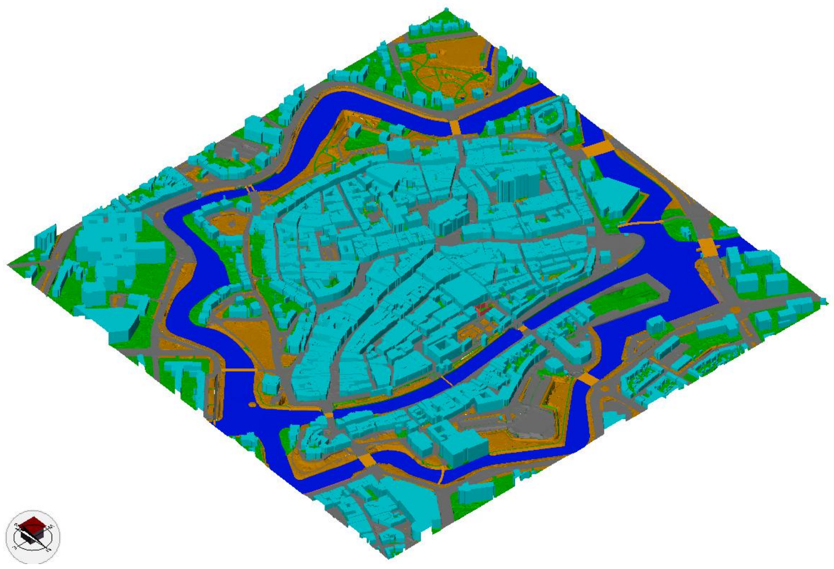


Figure 1.1: Snapshot of an area of 3DTOP10NL in CityGML. Notice that the terrain, roads, water courses are all triangulated, forming one large triangulation for the whole of the Netherlands.

1.2 Terrains in CityGML

CityGML is an XML based data model for the storage and exchange of virtual 3D city models (Groger et al., 2012). It is implemented as an application schema of GML3 (Geography Markup Language version 3.1.1) and models 3D geometry along with semantics. The data model of CityGML comprises of a core module and several thematic extension modules like `Building, Relief, Bridge, LandUse, Transportation, Vegetation, WaterBody, etc.` (Groger et al., 2012). The terrain/relief is an integral part of a 3D city model. CityGML allows to store terrains as TINs and Grids. In CityGML, the DTM is provided by the thematic module `Relief`. The terrain can be represented either as a TIN `(TINReflief)`, or as a Grid `(RasterRelief)`, or as masspoints `(MasspointRelief)`, or as breaklines `(BreaklineRelief)` (Figure 1.2). The corresponding GML3 classes are: `gml:RectifiedGridCoverage` for grids, `gml:MultiCurve` for break lines, `gml:MultiPoint` for mass points and `gml:Triangulated-Surface` or `gml:Tin` for TINs (Groger et al., 2012). It is also possible to represent a terrain with a combination of different terrain types within a single dataset. For instance, terrain can be modelled by a coarse grid with some areas depicted by detailed TIN (Figure 1.3) or as a TIN with break lines to depict a constrained triangulation, etc. The validity of each terrain type is limited to a certain area defined by the `validity_extent_polygon`.

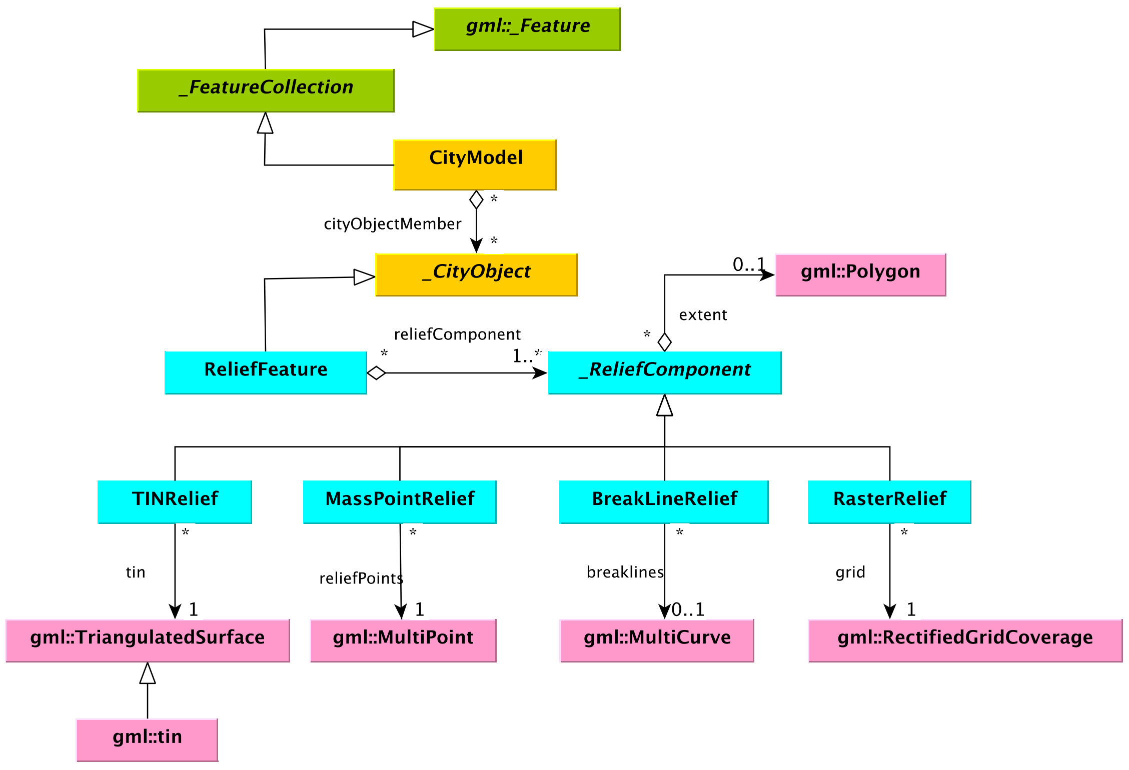


Figure 1.2: Digital Terrain Model in CityGML (Groger et al., 2012)

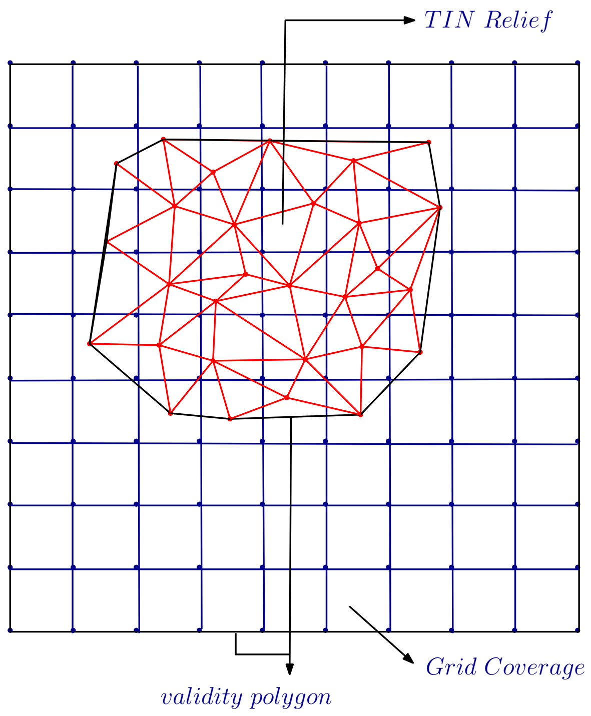
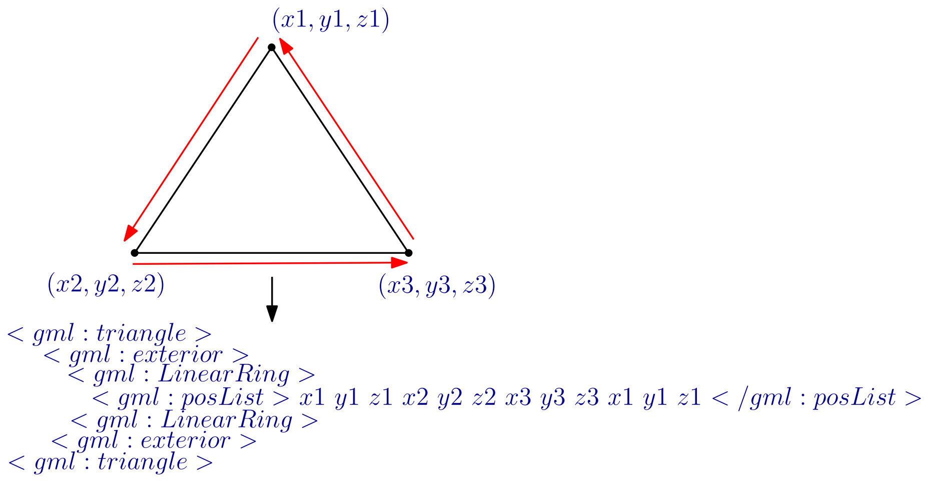


Figure 1.3: TIN + Grid combination in CityGML. TIN vertices may lie anywhere on the grid and not necessarily at the centre of each grid pixel. (Kumar et al., 2016)

1.2.1 TINs in CityGML

The class `TINRelief` describes a terrain as a triangulated surface (i.e. a TIN) embedded in 3D space. Its geometry is specified by the GML class `gml:TriangulatedSurface` which is a collection of triangles with `gml:Triangle` geometry (Figure 1.6). The triangles are stored as per the Simple Feaure structure (Figure 1.4) i.e. as a closed linear ring of vertices with the repetition of first vertex as the last vertex of the ring. Within its subclass `gml:Tin`, only the points (with 3D coordinates) are represented, along with optional breaklines (which acts as constraints in the triangulation (Shewchuk, 1997)), control points, etc. The support for TINs in GML3 as Linear Rings is consistent with OGC Geometry Abstract Specifications, ISO 19107:2003 Spatial schema and OGC Simple Feature Common Architecture (OGC, 2011).



```
< gml : triangle >
    < gml : exterior >
       < gml : LinearRing >
          < gml : posList > x1 y1 z1 x2 y2 z2 x3 y3 z3 x1 y1 z1 < /gml : posList >
       < gml : LinearRing >
     < gml : exterior >
< gml : triangle >
```

Figure 1.4: OGC Simple Feature

1.2.2 Grids in CityGML

The class `RasterRelief` defines the storage of a terrain as a grid coverage in CityGML. It's features are defined by the GML class `gml:RectifiedGridCoverage`. The rectified grid coverage is a discrete point coverage in which the grid points are geometrically referenced (Portele, 2012). It has a domain which is a rectified grid comprising of a finite number of geometric locations. The range of the coverage has a finite number of attribute values associated with the point locations. The range can also have a hyperlink to an external file (say a GeoTIFF file) containing the geodata (x, y, elevation). The geometry of a rectified grid is shown in Figure 1.5. A point location is defined as the grid origin (O) and the offset vectors (u, v) specifies the position of the next point locations.

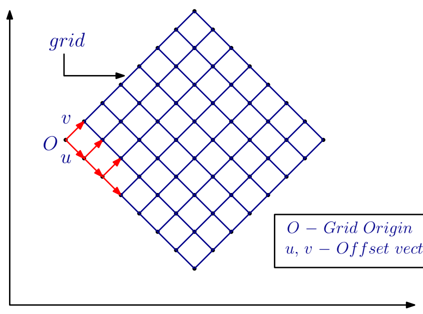


Figure 1.5: Geometry of Rectified Grid (Portele, 2012)

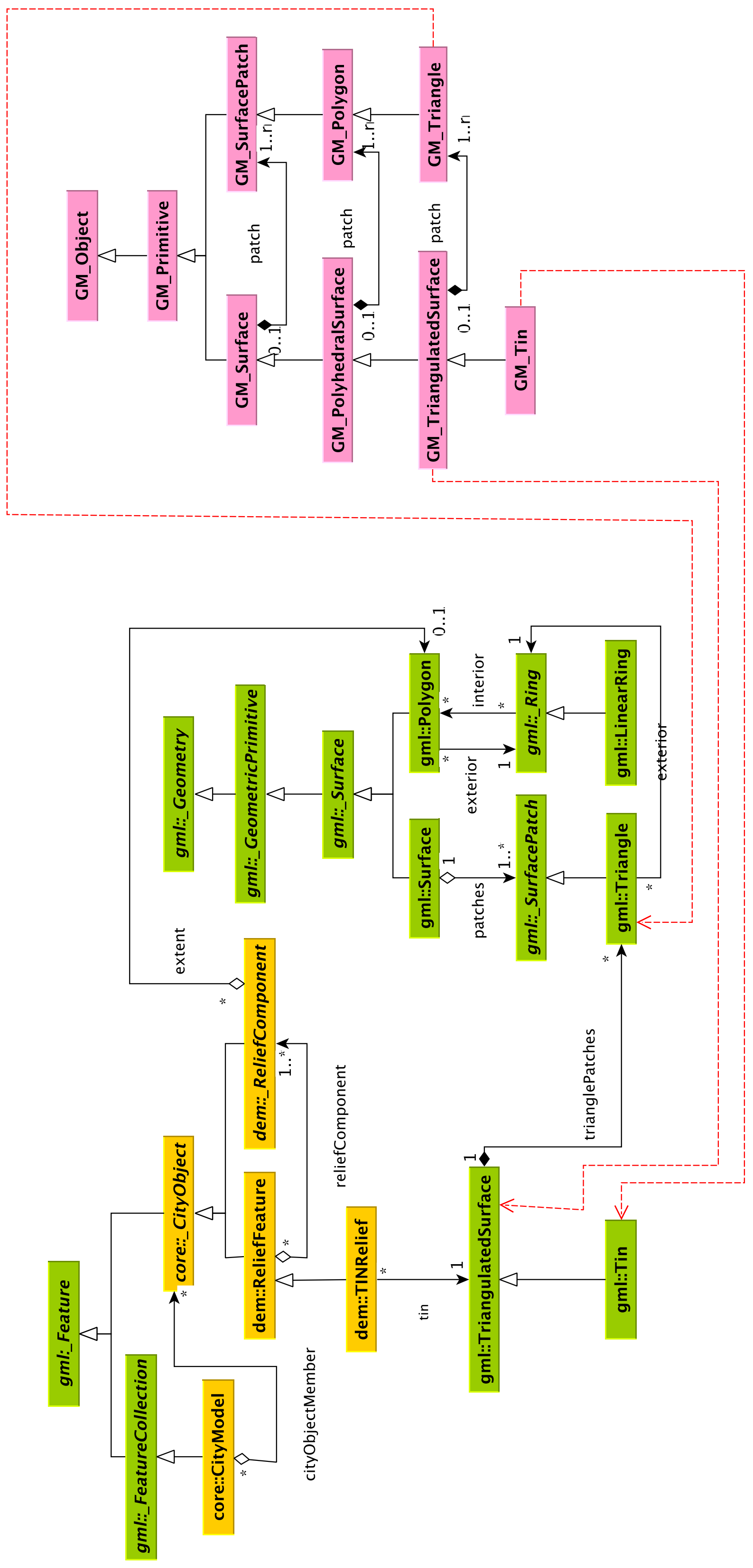


Figure 1.6: TINs in CityGML and ISO 19107:2003 Spatial schema (Kumar et al., 2016)

1.3 Problems in storing massive terrains in CityGML

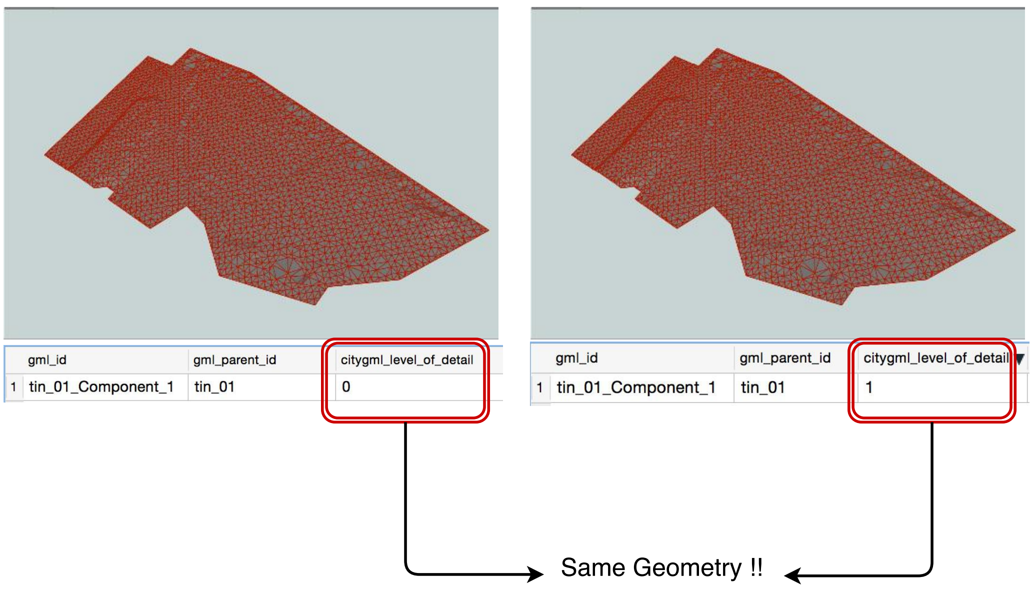
Nowadays 3D data acquisition and processing techniques such as airborne laser scanning and multi-beam echosounding generate billions of 3D points for simply an area of few square kilometers. The TIN generated from these points could be massive in size even for a small city like Delft. Kumar et al. (2016) enlists several problems associated with the storage of these massive triangulations with current CityGML Simple Feature storage solution. Some of them are:

1. *Huge data volumes* : Since every triangle is stored as a linear ring of vertices in Simple Feature structure (Figure 1.4), the size of the CityGML files is increased considerably with the repeated storage of vertex information.

2. *No referencing of triangles and their vertices* : There is no referencing scheme for the vertices of a triangle in Simple Feature structure. Each of the triangle is specified with the coordinates of its vertices in full which takes more space (Figure 1.4). This is also the main cause of large size of CityGML datasets, as mentioned in problem 1.

3. *No topology* : There is no storage of topological information of TIN in Simple Feature and triangle structure. The triangles are stored individually irrespective of their neighbours. This hinders spatial analysis greatly.

4. *No specifications for terrain LODs* : There is no distinction between different LODs of a terrain in CityGML at geometrical and semantic level. The CityGML 3.0 provides extended LODs for the `Building` module only and the LOD specifications of other modules like Relief, etc. are left out as unessential features (Löwner et al., 2013; Gröger and Löwner, 2016). Giving only a `gml:lod` does not solve the issue if we cannot identify the difference between LODs (Figure 1.7).



	gml_id	gml_parent_id	citygml_level_of_detail
1	tin_01_Component_1	tin_01	0

	gml_id	gml_parent_id	citygml_level_of_detail
1	tin_01_Component_1	tin_01	1

Figure 1.7: CityGML LOD concept for a terrain

5. *Vertical triangles are not handled* : 3DTOP10NL models urban objects like buildings, roads, etc. integrated in the terrain. In a way it is not completely 2.5D but a 2.75D model with vertical walls. A 2.75D models is a 2-manifold surface embedded in a 3D space (Gröger and Plümer, 2005). When a 2.75D model is projected on a 2D surface, the vertical surfaces flatten out which distorts the geometry of the model. There is no mechanism in CityGML to mark out these vertical surfaces so as to remove them while transforming from 3D to 2D.

1.4 Visualization of massive terrains

Visualization is an important and complex issue in the context of 3D city models. A large number of software have been developed for the offline visualization of CityGML data such as FZK Viewer, CityGML SpiderViewer, Autodesk LandXplorer, FME Data Inspector, etc. However, these software are required to be installed on the user machine in order to work with CityGML files. The visualization of 3D city models represented with CityGML on web is still a challenging area. CityGML is designed for the storage and exchange of 3D city models and not for visualizing them. Visualizing CityGML over web requires to follow another pathway of separating the geometric information from semantic part in the commonly used 3D graphics formats and using them to visualize the model. With the increasing size of datasets, CityGML files become too large and cannot be rendered directly over the web browser. Also, sometimes the 3D data cannot be visualized as the user did not install the right browser plugins. In order to visualize CityGML data over a web browser, several 3D graphical standards like X3D[1], KML[2]/COLLADA[3], etc. are used.

Terrain rendering has been an open problem for quite a long time. Many different applications require visualizing the 3D city models at different LODs. There is no distinction (geometric and semantic) between the LODs of a terrain in the CityGML. The enormous amount of data to be fetched, the heterogeneity of data sources, the complexity of rendering are only a few parts of this challenge. The crucial thing for rendering terrains is the geometry. This answers two questions:

- What is the elevation of the points in terrain?
- What is the representation used for surface geometry? A heightmap or a triangle mesh?

Lindstrom et al. (1996) first proposed the quadtree structure for rendering height maps which has been adopted by closed source systems like Nokia maps, Apple maps, etc. for creating virtual earths and map applications. Popular application Google Earth rely on KML (Keyhole Markup Language) for rendering the 3D city models. The open source community has good examples like World Wind (Bell et al., 2007), Cesium JS [4], OpenWebGlobe [5] or WebGLEarth [6]. With terrain datasets crossing the billion mark in the count of triangle geometries, the common bottleneck of web visualization is the rate at which these triangles can be rendered over the graphics engine.

1.4.1 X3D

X3D is an XML based, open 3D data format and is used for representing 3D scenes in web environment. It is the successor of VRML (Virtual Reality Modelling Language)[7]. Several studies have been conducted to visualize CityGML data over the web browser using X3D. Mao and Ban (2011) developed a framework for the online visualization of CityGML models. According to the research, 3D scenes are generated from the CityGML data based on the geometric and semantic information, which are then viewed in the web browser using X3DOM. Supporting the importance of X3D, Prieto et al. (2012) introduced a framework for the visualization of CityGML data over web (without any dependency over plugins) using X3D and W3DS (Web 3D Service).

1.4.2 KML/COLLADA

KML (Keyhole Markup Language) is a file format used to display geographic data in an Earth browser such as Google Earth. KML Version 2.2 has been adopted as an OGC implementation standard. KML

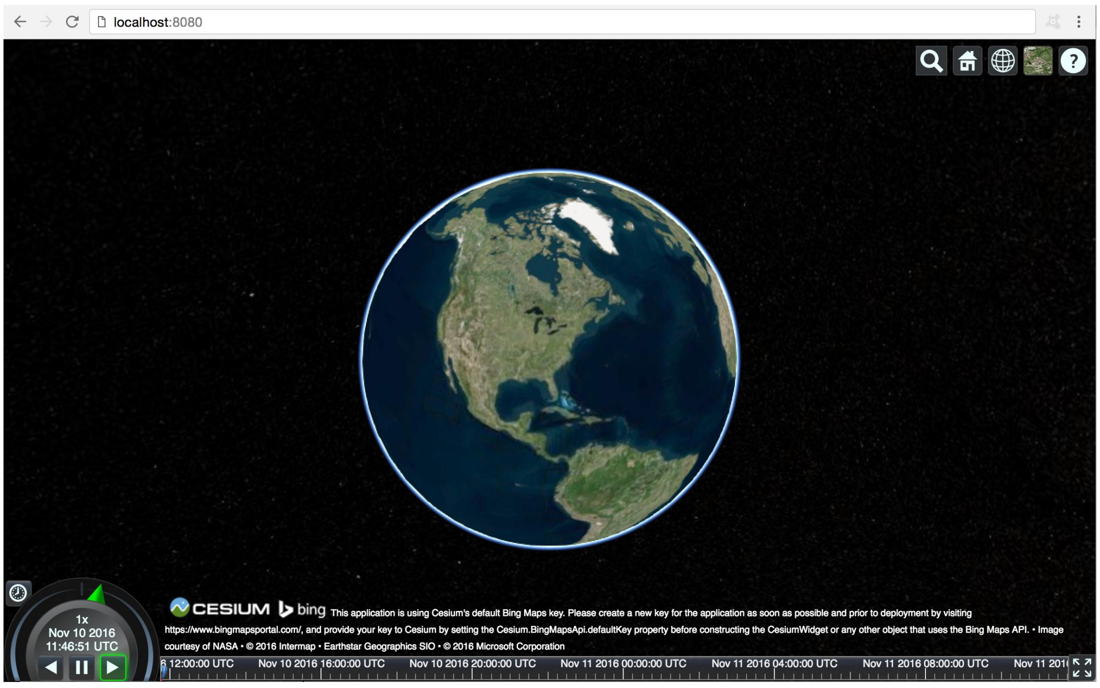
[1]http://www.web3d.org/x3d/what-x3d
[2]https://developers.google.com/kml/
[3]https://www.khronos.org/collada/
[4]http://cesiumjs.org/
[5]http://www.openwebglobe.org
[6]http://www.webglearth.org/
[7]http://gun.teipir.gr/VRML-amgem/spec/index.html

focuses on geographic visualisation, including annotation of maps and images. Although KML is not designed for 3D visualisation, it uses COLLADA for 3D modelling. COLLADA (COLLAborative Design Activity) is an XML based open standard for the representation and exchange of 3D assets between the applications. It focuses on the exchange of geometric data and 3D scenery. KML/COLLADA is designed for Earth browser, while X3D is a better choice to present online 3D city models for its compatibility with the HTML and wide support from popular browsers such as Firefox or Chrome.

1.4.3 Virtual globes

With the advancements in the development of 3D web based applications, virtual globes have emerged as a new medium for visualizing and interacting with the geodata. They provides the user ability to freely move around in the virtual environment by changing the viewing angle and position. To develop cross-platform and cross-browser applications, several WebGL based virtual globes have been developed like Cesium, OpenWebGlobe, WebGLEarth, etc.. The virtual globe worth mentioning is *Cesium*. Cesium is an open-source JavaScript library to create 3D virtual globes as well as 2D maps on a web browser Figure 1.8. The main features of Cesium are:

- a 3D globe, 2D map, and Columbus view (2.5D) with the same API.
- support for visualizing 3D models in industry standard formats like KML, glTF (GL Transmission Format), GeoJSON, TopJSON, CZML (Cesium Language), etc.
- support for high resolution terrain visualization.
- layer imagery from multiple sources, including WMS, TMS, WMTS, Bing Maps, Mapbox, Google Earth Enterprise, OpenStreetMap, ArcGIS MapServer, standard image files, and custom tiling schemes.
- includes extensive libraries which support 2D as well as 3D geometries. The user can draw polyline, polygon, ellipsoid, sphere, labels, billboards and sensors.
- includes handlers to control mouse/keyboard events, camera movements and zoom and pan the virtual globe.



This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • Image courtesy of NASA • © 2016 Intermap • Earthstar Geographics SIO • © 2016 Microsoft Corporation

Figure 1.8: Cesium web globe

Chapter 2

Proposed solution

Based on the weaknesses of the current storage solution (as highlighted in Section 1.3), a solution has been proposed (Kumar, 2015) for:

- developing a robust schema for the compact storage of terrains (TINs) in CityGML and
- visualizing and disseminating the CityGML terrains.

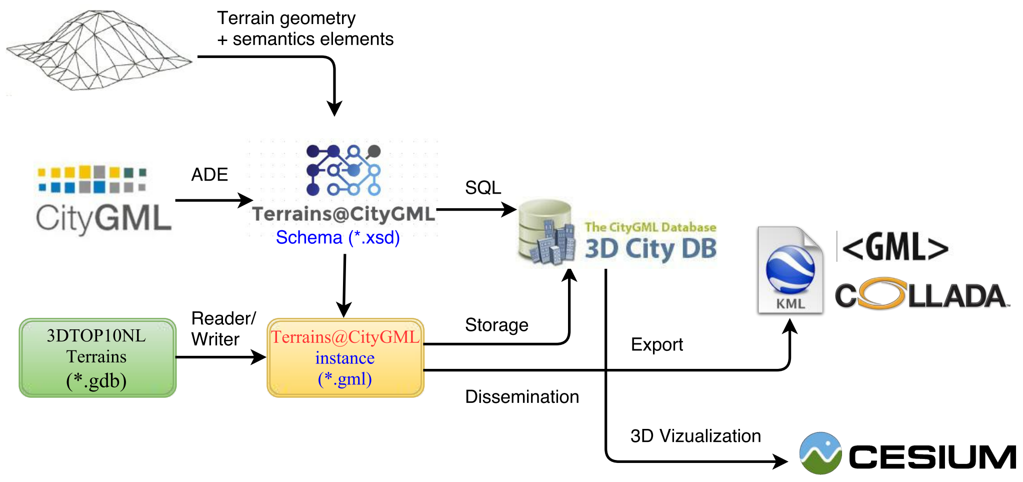


Figure 2.1: Framework for proposed solution. CityGML/GML are extented to store terrain geometry and semantics as proposed.(Kumar, 2015)

2.1 For storage

The proposal is to develop an alternative representation for terrain (TINs), `Terrains@CityGML` as an extension to CityGML by adding new geometry encoding schemes for TINs in the CityGML and GML schema. CityGML has the concept of *ADE (Application Domain Extension)* to extend the schema with new classes/attributes which are not explicitly modelled in CityGML. ADEs are increasingly being used in creating application-specific extensions like for energy modelling (Nouvel et al., 2015), BIM-IFC integration with CityGML (de Laat and Van Berlo, 2011), IMGeo for modelling Dutch topographic data in CityGML (Brink et al., 2013), indoor modelling (Kim et al., 2014), noise modelling (Groger et al., 2012), etc. Similarly, there are many specialized versions of GML. For instance, *FieldGML* is GML based implementation for the representation of geofields in 2D and 3D (Ledoux, 2008). Another GML based implementation is *NcML-GML* which provides for storing the metadata of netCDF files in GML (Nativi et al., 2005). Others include *CSML* (Woolf et al., 2006), *GeoSciML* (Sen and Duffy, 2005), etc.

The ADE will cover all the four aspects of modelling a terrain in CityGML: *geometry, semantics, texture and LOD*. New geometry types for indexing vertices and triangles will be introduced in the GML schema. They can be:

1. a set of vertices/points in 2D or 3D space.
2. a set of connected triangles and their vertices in 2D or 3D space.
3. a set of connected triangles, their vertices and additional constraints (breaklines) in 2D or 3D space.
4. a set of vertex stars.

The CityGML schema will be extended to include the new geometry types and semantics for the buildings, water bodies, terrain/relief, roads, etc.. The ADE will have two main terrain representation types `TINTerrain` and `StarTerrain` capable of storing TINs as a collection of triangles (VTP, 2012) and as a collection of stars (Ledoux, 2015), respectively. A bucketing or a tiling scheme will be introduced for managing the massive TINs. Figure 2.2 depicts a snapshot of the preliminary ADE schema for the `TINTerrain` type. The experimental results of prototype implementation to convert TINs to the proposed CityGML ADE structure shows that it is possible to compress the data size up to a factor of 25 with massive real-world terrains (more than 1 billion triangles) (Kumar et al., 2016).

2.2 For visualization

For visualizing, the proposed framework is based on using Cesium 3D webglobe for rendering terrains over web. Cesium has two terrain formats: *heightmaps*[1] and *quantized mesh*[2]. The *heightmaps* can be considered as a uniform grid of point-height values organized in tiles. Each tile overlaps with its neighbours at the edges so as to have a seamless terrain. The *quantized mesh* format has similar tile structure as the heightmaps, but here each tile is optimised for rendering terrains at a large scale. An irregular mesh of triangles is pre rendered for each of the tiles. It provides a better representation of terrains with less details in flat areas and more details for a steep terrain. It is more memory efficient and is rendered fast.

Both these formats require converting the terrain data to Cesium specific format glTF (GL Transmission Format [3]) before rendering over Cesium Figure 2.3. As per the research requirements, the input for rendering should be a well formed *Terrain@CityGML* ADE instance. The plan is to use Cesium quantized mesh format for rendering massive terrains without following the conversion to Cesium specific glTF. The *Terrain@CityGML* is already an indexed based structure and can be easily integrated in the Cesium API. The open source *Cesium API* will be extended to include readers/writers for *Terrain@CityGML* data. *Cesium Terrain Builder*[4], a C++ library, will be used to create the terrain tiles. A *Cesium Terrain server*[5] will be setup to host our tile-set over web. Figure 2.4 depicts the process of generating tiles for the *Terrain@CityGML* data. The dissemination products will be in CityGML, OBJ and 3D SHP formats. The export functionality for these formats will be added to the 3DCityDB Exporter tool and an open source transformation tool for the file based outputs will be developed.

[1]http://cesiumjs.org/data-and-assets/terrain/formats/heightmap-1.0.html
[2]https://cesiumjs.org/data-and-assets/terrain/formats/quantized-mesh-1.0.html
[3]https://github.com/KhronosGroup/glTF
[4]https://github.com/geo-data/cesium-terrain-builder
[5]https://github.com/geo-data/cesium-terrain-server

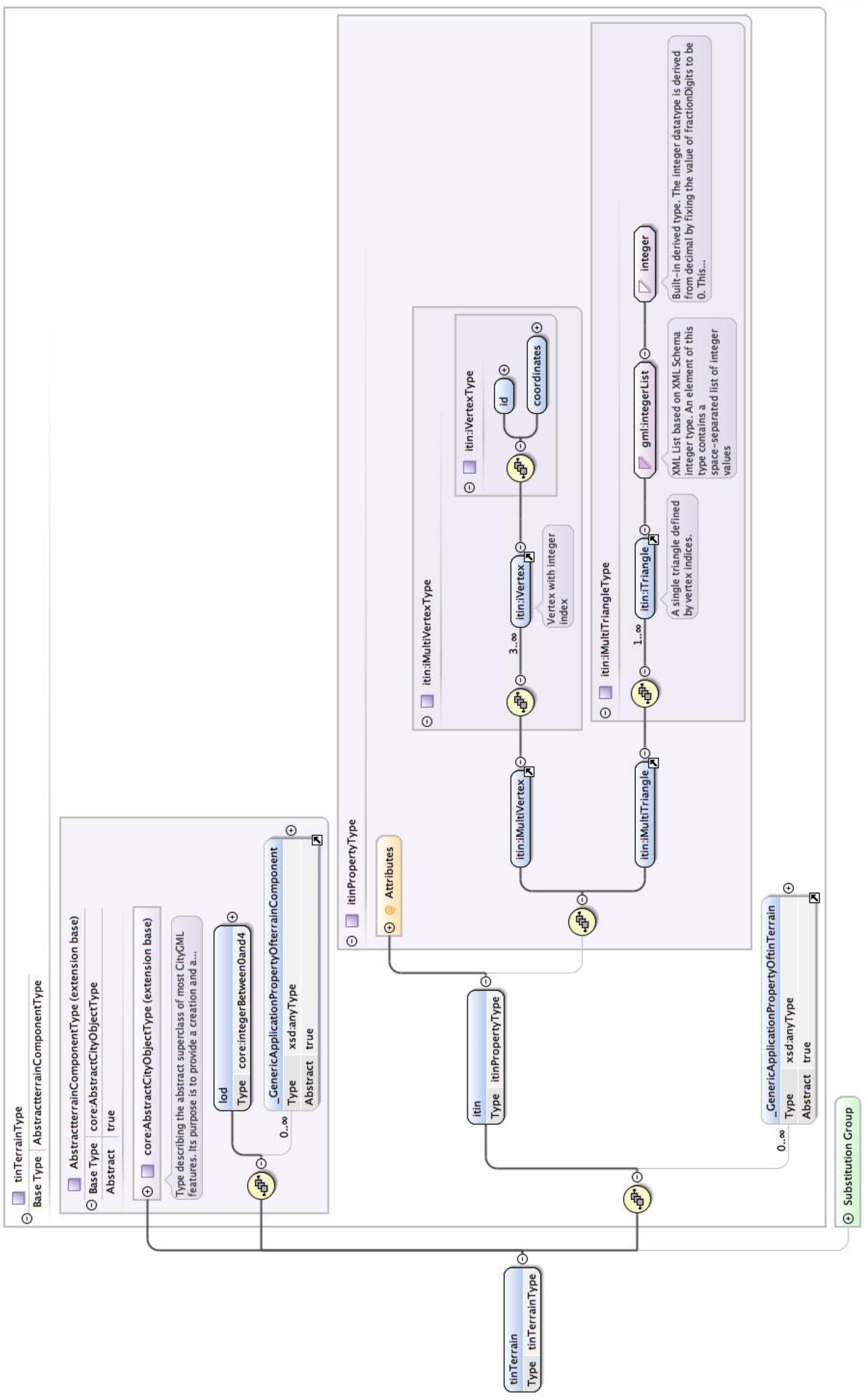

tinTerrainType
Base Type AbstractterrainComponentType
AbstractterrainComponentType (extension base)
Base Type core:AbstractCityObjectType
Abstract true
core:AbstractCityObjectType (extension base)
Type describing the abstract superclass of most CityGML features. Its purpose is to provide a creation and a...
lod
Type core:integerBetween0and4
_GenericApplicationPropertyOfterrainComponent
Type xsd:anyType
Abstract true
0..∞
itinPropertyType
Attributes
itin:iMultiVertex
itin:iMultiVertexType
3..∞
itin:iVertex
Vertex with integer index
itin:iVertexType
id
coordinates
itin:iMultiTriangle
itin:iMultiTriangleType
1..∞
itin:iTriangle
A single triangle defined by vertex indices.
gml:integerList
XML List based on XML Schema integer type. An element of this type contains a space-separated list of integer values
integer
Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
itin
Type itinPropertyType
_GenericApplicationPropertyOftinTerrain
Type xsd:anyType
Abstract true
0..∞
Substitution Group
tinTerrain
Type tinTerrainType

Figure 2.2: Snapshot of the new geometry types in GML for the `TINTerrain` type (Kumar, 2015)

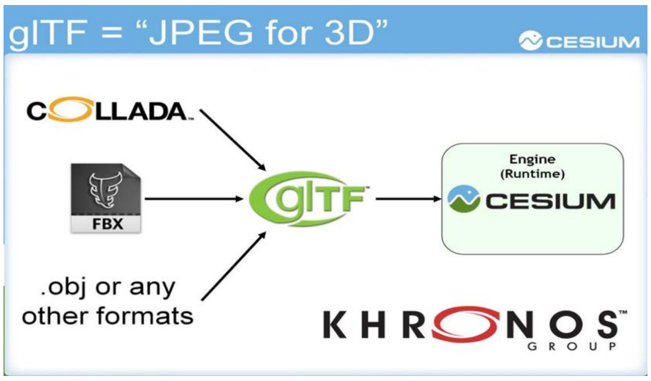


Figure 2.3: Cesium glTF pathway (Mathew Amato, 2015)

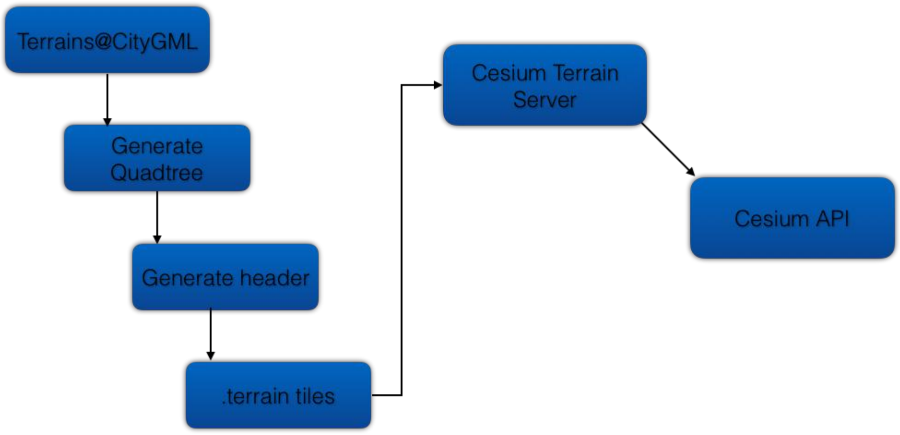


Figure 2.4: TIN terrain as quantised mesh (Kumar, 2015)

Chapter 3

Imagery services with Cesium

Cesium provides support for rendering high resolution imagery from different standard data services. Several different layers can be ordered and blended together. A particular advantage of using these data services is that the data remains at the source and is not physically delivered. This makes it possible to guarantee the timeliness, reliability and availability of data. Cesium has support for different data services like WMS, WMTS, TMS, Open Street Maps, Bing Maps, ESRI ArcGIS MapServer,Google Earth Enterprise, etc.

3.1 WMS (Web Map Service)

The OpenGIS WMS[1] provides a simple HTTP interface for requesting georeferenced map images from distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more georeferenced map images. A map is not the data itself but is usually an image (in JPEG, PNG, etc. format) which can be displayed in web applications.

In Cesium, `WebMapServiceImageryProvider` is used to display the image maps hosted by a WMS server. It has different options[2] to specify the properties of the imagery to be fetched from the server. Table 3.1 lists the options utilised in the implementation. Given below is a code snippet for WMS in Cesium. The map layer used is the `TOP100Raster`[3](EPSG:28992) of Netherlands.

The Cesium output is shown in Figure 3.2

```
 var rect = new Cesium.Rectangle(
     Cesium.Math.toRadians(3.3700),
     Cesium.Math.toRadians(50.7500),
     Cesium.Math.toRadians(7.2100),
     Cesium.Math.toRadians(53.4700));
var provider = new Cesium.WebMapServiceImageryProvider({
     url: 'https://geodata.nationaalgeoregister.nl/top100raster/wms?",
     layers: 'top100raster'
     tilingScheme : new Cesium.GeographicTilingScheme(
     { rectangle: rect, }
     ),
     tileWidth:256,
     tileHeight:256
})
```

[1]http://www.opengeospatial.org/standards/wms
[2]https://cesiumjs.org/Cesium/Build/Documentation/WebMapServiceImageryProvider.html
[3]https://www.pdok.nl/nl/service/wms-top100raster

Name	Type	Description
url	String	URL of the WMS service
layers	String	Names of the layers to be displayed
tileWidth	Number	(optional) Width of each tile in pixels
tileHeight	Number	(optional) Height of each tile in pixels
tilingScheme	TilingScheme	(optional) The tiling scheme corresponding to the organization of the tiles in the TileMatrixSet
rectangle	Rectangle	(optional) The rectangle covered by the layer
minimumLevel	Number	(optional) The minimum level-of-detail supported by the imagery provider
maximumLevel	Number	(optional) The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit

Table 3.1: `WebMapServiceImageryProvider` options

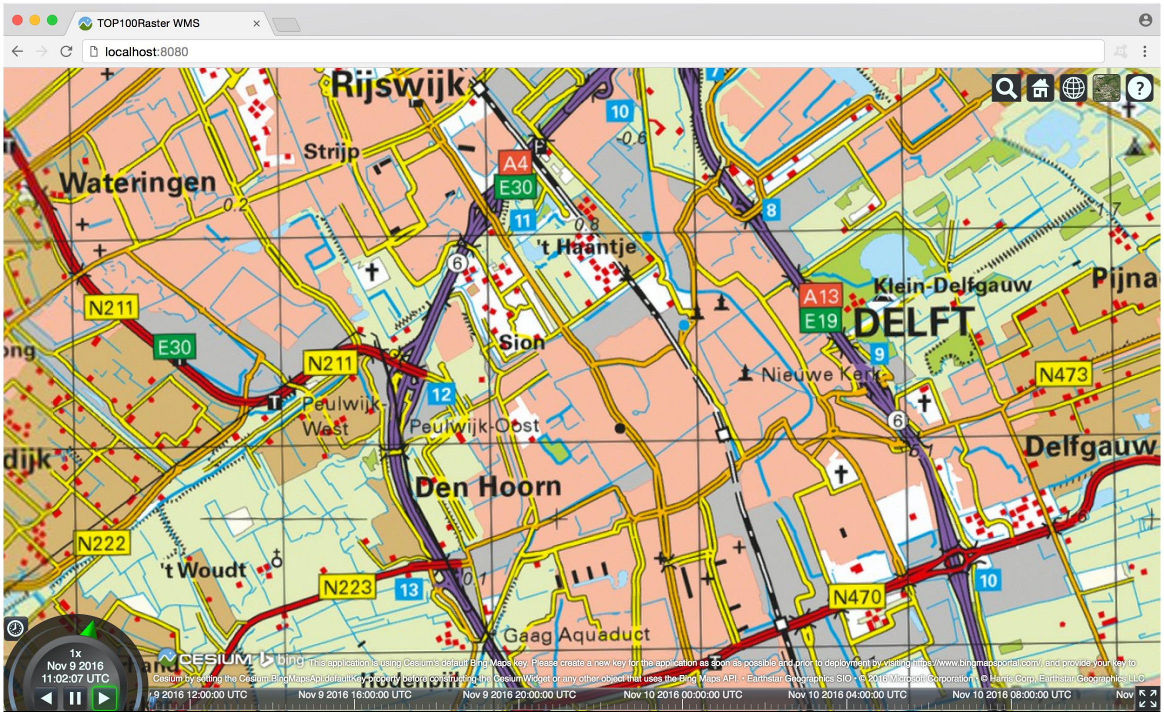


Figure 3.1: `TOP100Raster` WMS over Cesium webglobe

3.2 TMS (Tile Map Service)

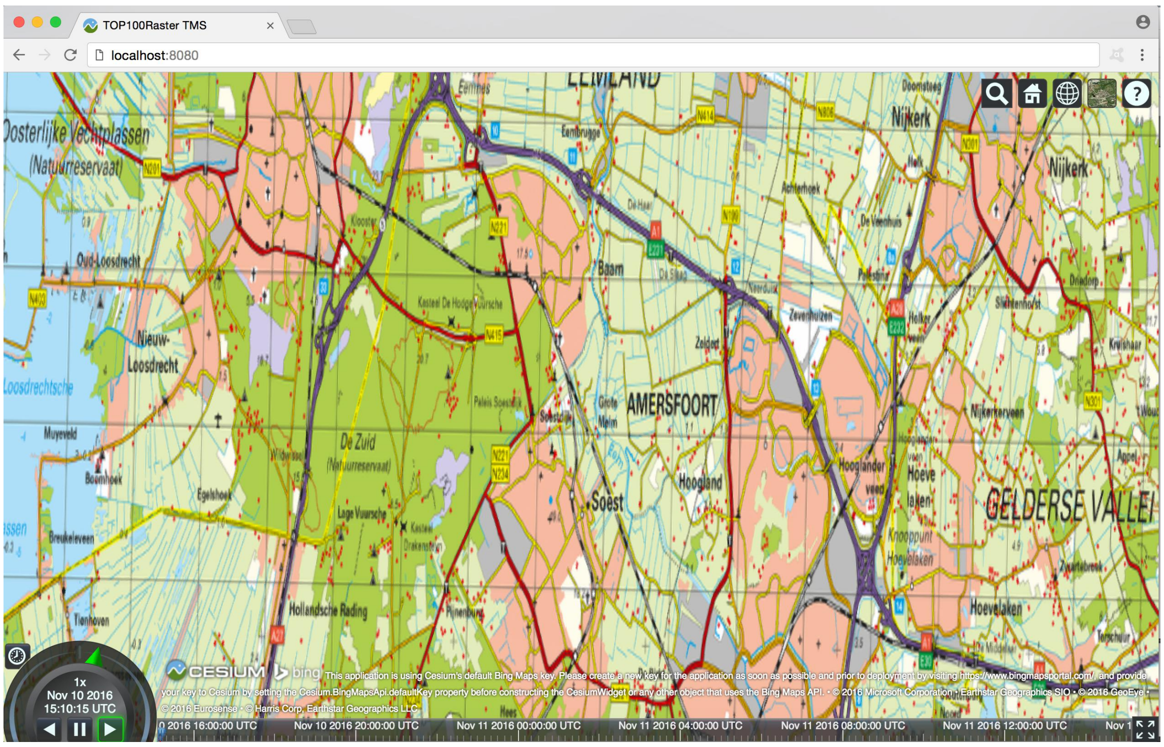
OSGeo TMS[4] delivers map tiles of georeferenced data at fixed scales. TMS is pure RESTful implementation build from scratch. Cesium has `createTileMapServiceImageryProvider` to deliver the tiled imagery served by TMS compliant servers. For instance, the GDAL2Tiles[5] is an open-source project which allows to create TMS compatible map tiles. It has different options[6] to specify the properties of the imagery to be fetched from the server. Table 3.2 lists the options utilised in the implementation. Given below is a code snippet for TMS in Cesium. The map layer used is the `TOP100Raster`[7] of Netherlands. The Cesium output is shown in Figure 3.2.

```
 var rect = new Cesium.Rectangle(
     Cesium.Math.toRadians(3.3700),
     Cesium.Math.toRadians(50.7500),
     Cesium.Math.toRadians(7.2100),
     Cesium.Math.toRadians(53.4700));
var provider = Cesium.createTileMapServiceImageryProvider({
     url : 'https://geodata.nationaalgeoregister.nl/tms/1.0.0/
     top100raster@EPSG:28992@png',
     fileExtension: 'png',
tab tilingScheme : new Cesium.GeographicTilingScheme(
     { rectangle: rect }
     ),
     tileWidth: 256,
     tileHeight: 256,
     maximumLevel: 14
});
```

[4]http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification
[5]http://www.klokan.cz/projects/gdal2tiles/
[6]https://cesiumjs.org/Cesium/Build/Documentation/createTileMapServiceImageryProvider.html
[7]https://www.pdok.nl/nl/service/wms-top100raster

Name	Type	Description
url	String	Path to image tiles on the server
fileExtension	String	File extension for the images on the server
tileWidth	Number	(optional) Width of each tile in pixels
tileHeight	Number	(optional) Height of each tile in pixels
tilingScheme	TilingScheme	(optional) The tiling scheme corresponding to the organization of the tiles in the TileMatrixSet
rectangle	Rectangle	(optional) The rectangle covered by the layer
minimumLevel	Number	(optional) The minimum level-of-detail supported by the imagery provider
maximumLevel	Number	(optional) The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit

Table 3.2: `createTileMapServiceImageryProvider` options



This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2016 Microsoft Corporation • Earthstar Geographics SIO • © 2016 GeoEye • © 2016 Eurosense • © Harris Corp, Earthstar Geographics LLC

Figure 3.2: `TOP100Raster` TMS over Cesium webglobe

3.3 WMTS (Web Map Tile Service)

The OpenGIS WMTS[8] delivers map tiles (mostly 256x256 pixel size) of georeferenced data using tile images with predefined content, extent, and resolution. The advantage of tiles is that they can be pre-renderd on the server side, and cached on the client side. This will reduce waiting time for the data and bandwith. WMTS is inspired by the OSGeo TMS.

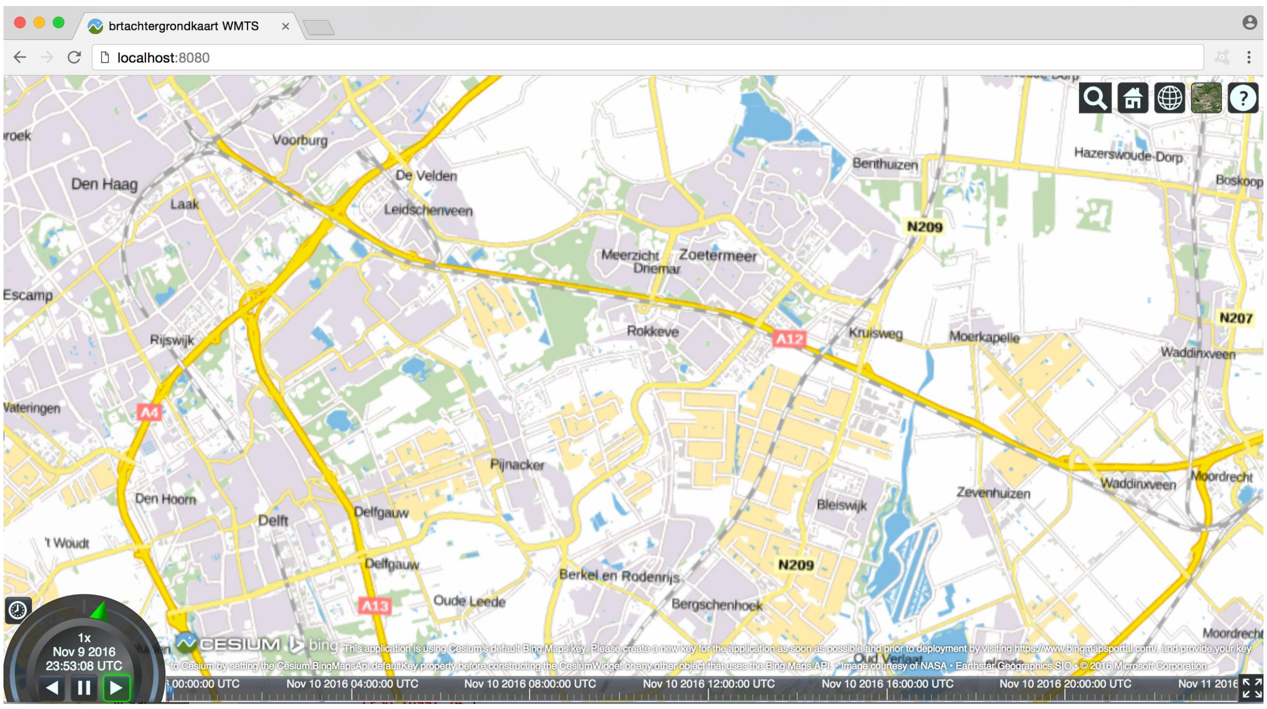
Cesium has `WebMapTileServiceImageryProvider` to deliver the tiled imagery served by WMTS 1.0.0 compliant servers. It has different options[9] to specify the properties of the imagery to be fetched from the server. Table 3.3 lists the options utilised in the implementation. Given below is a code snippet for WMTS in Cesium. The map layer used is the `brtachtergrondkaart`[10](Background map) of Netherlands. The Cesium output is shown in Figure 3.3 and Figure 3.4. Figure 3.3 depicts the tile layer (5,6) and Figure 3.4 depicts the tile layer (5,10).

```
 var rect = new Cesium.Rectangle(
     Cesium.Math.toRadians(3.3700),
     Cesium.Math.toRadians(50.7500),
     Cesium.Math.toRadians(7.2100),
     Cesium.Math.toRadians(53.4700));
var provider = new Cesium.WebMapTileServiceImageryProvider({
     url :  'https://geodata.nationaalgeoregister.nl/
     tiles/service/wmts/brtachtergrondkaart?request=GetCapabilities',
     layer :  'brtachtergrondkaart',
     style :  'default',
     format :  'image/png',
     tileMatrixSetID : 'EPSG:28992',
     tileMatrixLabels :  ['EPSG:28992:0','EPSG:28992:1','EPSG:28992:2',
     'EPSG:28992:3','EPSG:28992:4','EPSG:28992:5','EPSG:28992:6',
     'EPSG:28992:7','EPSG:28992:8','EPSG:28992:9','EPSG:28992:10',
     'EPSG:28992:11','EPSG:28992:12','EPSG:28992:13','EPSG:28992:14'],
     tilingScheme :  new Cesium.GeographicTilingScheme(
     { rectangle:  rect }
     ),
     tileWidth:  256,
     tileHeight:  256,
     maximumLevel:  14
})
```

[8]http://www.opengeospatial.org/standards/wmts
[9]https://cesiumjs.org/Cesium/Build/Documentation/WebMapTileServiceImageryProvider.html
[10]https://www.pdok.nl/nl/service/wmts-brt-achtergrondkaart

Name	Type	Description
url	String	The base URL for the WMTS-GetTile operation
format	String	Mime type for the image to be fetched from the server
layer	String	Name of the layer to be displayed
rectangle	Rectangle	(optional) The rectangle covered by the layer
tileMatrixSetID	String	The identifier of the TileMatrixSet to use for WMTS requests
tileMatrixLabels	Array	(optional) A list of identifiers in the TileMatrix to use for WMTS requests, one per TileMatrix level
tileWidth	Number	(optional) Width of each tile in pixels
tileHeight	Number	(optional) Height of each tile in pixels
tilingScheme	TilingScheme	(optional) The tiling scheme corresponding to the organization of the tiles in the TileMatrixSet

Table 3.3: `WebMapTileServiceImageryProvider` options



This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • Image courtesy of NASA • Earthstar Geographics SIO • © 2016 Microsoft Corporation

Figure 3.3: `brtachtergrondkaart` WMTS over Cesium webglobe. Query parameters: service:WMTS version:1.0.0, request:GetTile, tilematrix:EPSG:28992:4, layer:brtachtergrondkaart, style:default, tilerow:5, tilecol:10, tilematrixset:EPSG:28992, format:image/png

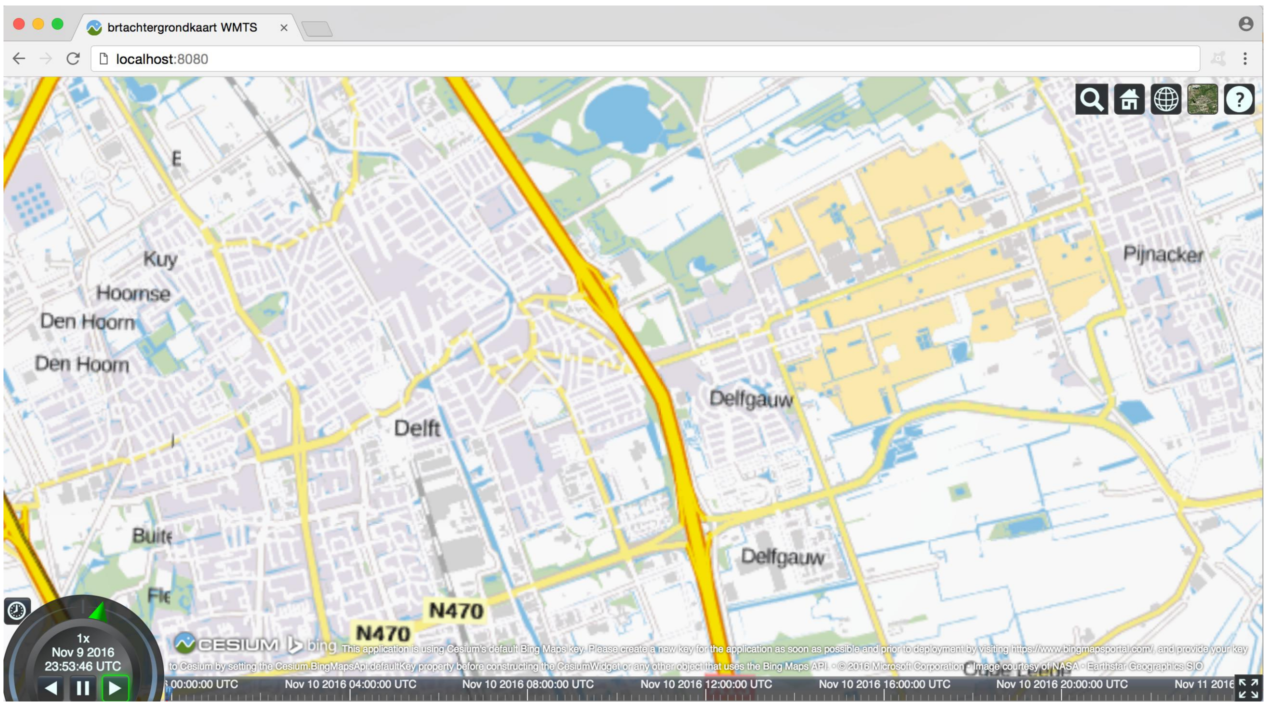


Figure 3.4: `brtachtergrondkaart` WMTS over Cesium webglobe. Query parameters: service:WMTS, version:1.0.0, request:GetTile, tilematrix:EPSG:28992:4, layer:brtachtergrondkaart, style:default, tilerow:5, tilecol:6, tilematrixset:EPSG:28992, format:image/png

References

Bell, D. G., Kuehnel, F., Maxwell, C., Kim, R., Kasraie, K., Gaskins, T., Hogan, P., and Coughlan, J. (2007). NASA World Wind: Opensource GIS for mission operations. In *2007 IEEE Aerospace Conference*, pages 1–9. IEEE.

Biljecki, F., Heuvelink, G. B., Ledoux, H., and Stoter, J. (2015a). Propagation of positional error in 3D GIS: estimation of the solar irradiation of building roofs. *International Journal of Geographical Information Science*, 29(12):2269–2294.

Biljecki, F., Ohori, K. A., Ledoux, H., Peters, R., and Stoter, J. (2016). Population estimation using a 3D city model: A multi-scale country-wide study in the netherlands. *PloS one*, 11(6):e0156808.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015b). Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.

Brink, L., Stoter, J., and Zlatanova, S. (2013). UML-based approach to developing a CityGML Application Domain Extension. *Transactions in GIS*, 17(6):920–942.

Çağdaş, V. (2013). An Application Domain Extension to CityGML for immovable property taxation: A Turkish case study. *International Journal of Applied Earth Observation and Geoinformation*, 21:545–555.

de Laat, R. and Van Berlo, L. (2011). Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In *Advances in 3D geo-information sciences*, pages 211–225. Springer.

Döllner, J., Kolbe, T. H., Liecke, F., Sgouros, T., Teichmann, K., et al. (2006). The virtual 3D city model of berlin-managing, integrating, and communicating complex urban information. In *Proceedings of the 25th Urban Data Management Symposium UDMS*, volume 2006, pages 15–17.

Eicker, U., Nouvel, R., Duminil, E., and Coors, V. (2014). Assessing passive and active solar energy resources in cities using 3D city models. *Energy Procedia*, 57:896–905.

Elberink, S. O., Stoter, J., Ledoux, H., and Commandeur, T. (2013). Generation and dissemination of a national virtual 3D city and landscape model for the Netherlands. *Photogrammetric engineering & remote sensing*, 79(2):147–158.

Groger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). OGC City Geography Markup Language (CityGML) Encoding Standard version 2.0.0.

Gröger, G. and Löwner, M.-O. (2016). Proposal for a new LoD concept for CityGML 3.0. Technical report, CityGML OGC Work Package 03.

Gröger, G. and Plümer, L. (2005). How to get 3-D for the price of 2-D? topology and consistency of 3-D urban GIS. *Geoinformatica*, 9(2):139–158.

Guo, R., Li, L., Ying, S., Luo, P., He, B., and Jiang, R. (2013). Developing a 3D cadastre for the administration of urban land use: A case study of Shenzhen, China. *Computers, Environment and Urban Systems*, 40:46–55.

Kadaster (2015). 3DTOP10NL. `http://arcg.is/1GKYy7E`. (Last accessed: April 15, 2016).

Kaden, R. and Kolbe, T. H. (2014). Simulation-Based Total Energy Demand Estimation of Buildings using Semantic 3D City Models. *International Journal of 3-D Information Modeling*, 3(2):35–53.

Kim, Y., Kang, H., and Lee, J. (2014). Developing CityGML indoor ADE to manage indoor facilities. In *Innovations in 3D Geo-information sciences*, pages 243–265. Springer.

Kolbe, T. H., Burger, B., and Cantzler, B. (2015). CityGML goes to broadway. In *Photogrammetric Week*, volume 15, pages 343–356.

Krüger, A. and Kolbe, T. (2012). Building analysis for urban energy planning using key indicators on virtual 3d city models?the energy atlas of berlin. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39(B2):145–150.

Kumar, K. (2015). Storage, maintenance and dissemination of massive 3D city models. PhD Proposal, 3D Geoinformation Group, Delft University of Technology, The Netherlands.

Kumar, K., Ledoux, H., and Stoter, J. (2016). A citygml extension for handling very large tins. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1:137–143.

Ledoux, H. (2008). FieldGML: An alternative representation for fields. In *Headway in Spatial Data Handling*, pages 385–400. Springer.

Ledoux, H. (2015). Storing and analysing massive TINs in a DBMS with a star-based data structure. Technical Report 2015.01, 3D geoinformation, Delft University of Technology, Delft, the Netherlands.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. (1996). Real-time, continuous level of detail rendering of height fields. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 109–118. ACM.

Löwner, M.-O., Benner, J., Gröger, G., and Häfele, K.-H. (2013). New concepts for structuring 3D city models - An extended level of detail concept for CityGML buildings. In *Computational Science and Its Applications–ICCSA 2013*, pages 466–480. Springer.

Mao, B. and Ban, Y. (2011). Online visualization of 3d city model using citygml and x3dom. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 46(2):109–114.

Mathew Amato (2015). Cesium and the future of 3D geospatial standards. `http://cesiumjs.org/presentations/Cesium-and-the-Future-of-3D-Geospatial-Standards.pdf`. JS Geo, Philadelphia. (Last accessed: July 10, 2016).

Nativi, S., Caron, J., Davis, E., and Domenico, B. (2005). Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-GML). *Computers & Geosciences*, 31(9):1104–1118.

Nouvel, R., Bahu, J.-M., Kaden, R., Kaempf, J., Cipriano, P., Lauster, M., and Casper, E. (2015). Development of the CityGML Application Domain Extension energy for urban energy simulation. *Proceedings of Building Simulation 2015*.

OGC (2011). Opengis® implementation specification for geographic information-simple feature access-part 1: Common architecture. *OGC document version 06-103r4*.

Portele, C. (2012). OGC Geography Markup Language (GML) Extended schemas and encoding rules GML Version 3.3 Doc. No. 10-129r1.

Prieto, I., Izkara, J. L., and del Hoyo, F. J. D. (2012). Efficient visualization of the geometric information of citygml: application for the documentation of built heritage. In *International Conference on Computational Science and Its Applications*, pages 529–544. Springer.

Sen, M. and Duffy, T. (2005). GeoSciML: development of a generic geoscience markup language. *Computers & Geosciences*, 31(9):1095–1103.

Shewchuk, J. R. (1997). *Delaunay Refinement Mesh Generation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburg, USA.

Stadler, A. and Kolbe, T. H. (2007). Spatio-semantic coherence in the integration of 3D city models. In *Proceedings of the 5th International Symposium on Spatial Data Quality, Enschede.*

Stoter, J., De Kluijver, H., and Kurakula, V. (2008). 3D noise mapping in urban areas. *International Journal of Geographical Information Science*, 22(8):907–924.

Stoter, J., Ploeger, H., and van Oosterom, P. (2013). 3D cadastre in the netherlands: Developments and international applicability. *Computers, Environment and Urban Systems*, 40:56–67.

VTP (2012). ITF Format - Virtual Terrain Project. `http://vterrain.org/Implementation/Formats/ITF.html`. (Last accessed: July 25, 2016).

Woolf, A., Lawrence, B., Lowry, R., Kleese van Dam, K., Cramer, R., Gutierrez, M., Kondapalli, S., Latham, S., Lowe, D., O'Neill, K., et al. (2006). Data integration with the climate science modelling language. *Advances in Geosciences*, 8(8):83–90.