Reports on Geodesy and Geoinformatics, 2019, Vol. 107, pp. 41–48

DOI: 10.2478/rgg-2019-0005
Received: 21 February 2019 / Accepted: 18 April 2019
Published online: 13 May 2019

ORIGINAL ARTICLE

Transformation of 3D geospatial data into CityGML – a case of Prague

Karel Janečka[1]*

[1]Faculty of Applied Science, University of West Bohemia, Technická 8, 306 14 Pilsen, Czech Republic

*kjanecka@kgm.zcu.cz

Abstract

The 3D geoinformation is becoming important for cities and their policies. The cities are therefore exploring the possibilities of 3D virtual city models for more efficient decision making. To maximize the economic benefit of such data, the cities can provide their 3D geospatial data for further usage, and so, new applications can be created. The paper defines a way how the freely available 3D geospatial data of Prague can be transformed from the proprietary data format into the open data model. The 3D geospatial data about the buildings, bridges and digital terrain model were transformed from the 3D shapefile into the CityGML. This is an application independent information model and exchange format. This will allow for the wider use of the 3D city model by different groups of users. The generated CityGML files were further imported into the spatial database with appropriate database CityGML-based scheme. It enables more efficient management and querying of CityGML data. To enable the wider audience to explore the 3D city model, the visualization in the web environment was also explored. The paper also presents the way how the attributes from the external data sources can be connected to the 3D objects in the web environment.

Key words: 3D city model, CityGML, data transformation

1 Introduction

The 3D city models are of crucial importance for the realization of city projects dealing with the third spatial dimension (elevation), for example, urban and spatial planning, environmental simulations or disaster management. Semantic 3D city models offer a reliable and increasingly available virtual representation of real world objects in an urban context (Willenborg et al., 2018). A comprehensive overview of applications of 3D city models is given by Biljecki et al. (2015).

The most important international standard that is used to model cities and landscapes in 3D is CityGML (Ohori et al., 2018). The CityGML standard is an application independent information model and exchange format for 3D geospatial data about objects like buildings or digital terrain model (DTM). From the geometry perspective, there are five standardized levels of details (LOD) in CityGML (Gröger et al., 2012). Figure 1 presents these five possible LOD.

The large cities are starting to provide their 3D geospatial data (often for free) and also in the CityGML format like Berlin[1] or Helsinki[2]. Some cities are offering their 3D geospatial data but not in CityGML data format. Therefore, there is a need for research on how to receive CityGML-based 3D city model from other data formats, both open and proprietary ones. Saran et al. (2018) introduced a flowchart for the transformation of 3D geospatial data in Collada[3] open data format into CityGML LOD3. Kolbe et al. (2015) described a creation of 3D city model for New York City (NYC) using datasets, which were freely available in NYC open data portal. These spatial datasets had only 2D geometries, except for DTM (2.5D). Floros and Dimopoulou (2016) presented the transformation of 3D geospatial data created in Google SketchUp modelling tool into CityGML

1 https://www.businesslocationcenter.de/berlin3d-downloadportal/?lang=en#/export (Accessed 8 January 2019)
2 https://hri.fi/data/en_GB/dataset/helsingin-3d-kaupunkimalli/resource/577f4286-7162-42e9-8ffe-%2052632228569e (Accessed 8 January 2019)
3 https://www.khronos.org/collada/ (Accessed 8 January 2019)

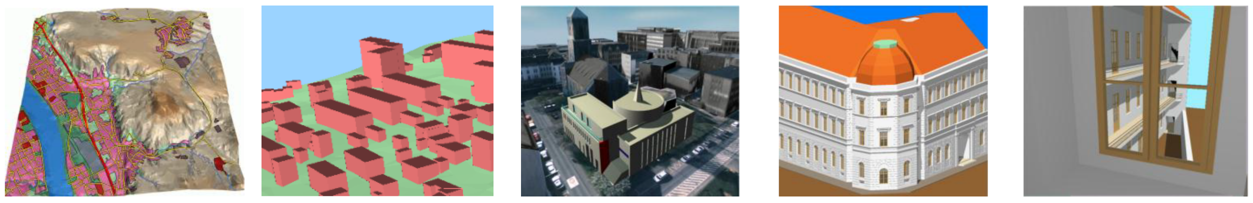

(a) LOD0 **(b)** LOD1 **(c)** LOD2 **(d)** LOD3 **(e)** LOD4

Figure 1. The five level of details defined by CityGML (Gröger et al., 2012)

LOD2 and LOD3.

The aim of the paper is to describe a transformation of the 3D geospatial data of Prague from the proprietary format (in particular, from the widely used ESRI 3D shapefile) into the CityGML, an open data model and XML-based format for the storage and exchange of virtual 3D city models. This will allow for the wider use of the 3D city model by different groups of users and also the other cities, which are going to make their 3D geospatial data available, could benefit from the findings described in the paper. Beside the data transformation, the appropriate database storage and web visualization are described.

The rest of the paper is structured as follows: Section 2 describes the input 3D geospatial data, which were available in the 3D shapefile proprietary data format. Section 3 presents the proposed transformation process using the FME tool. The validation of created CityGML data and their import into the spatial database are described in Section 4. Section 5 presents the visualization of data in the 3D web map client. It is also shown, in which way the attributes from the external data sources can be connected to the 3D data. The conclusions are stated in Section 6.

2 The source data

All the source 3D geospatial data were downloaded from Prague's geoportal.[4] The used 3D data were created by Prague Institute of Planning and Development, which is the body in charge of developing the concept behind the Prague's architecture, urbanism, development and formation. 3D data are available for the whole area of Prague.It should be noted, that Prague is currently the only city in the Czech Republic providing the 3D geospatial data in such extent for free. For testing purposes, only a subpart (the rectangle 2.5 $\times$ 2 km covering the city centre including, for example, Prague Castle or Charles Bridge) was used. The intention was to describe the principles of data transformation, storage and visualization, which can be later applied on larger data sets. In particular, the following objects were available:

- buildings,
- bridges,
- digital terrain model.

Figure 2 shows the downloaded input data visualized in Esri ArcScene desktop application. The used coordinate systems are S-JTSK/Krovak East North (EPSG:5514) and Baltic 1957 height (EPSG:8357). The 3D geospatial data were further available in proprietary formats dgn and dwg.

3 Data transformation

For data transformation, the FME software was selected as it enables the reading of input shapefiles and writing the results

[4] http://geoportalpraha.cz (Accessed 2 January 2019)

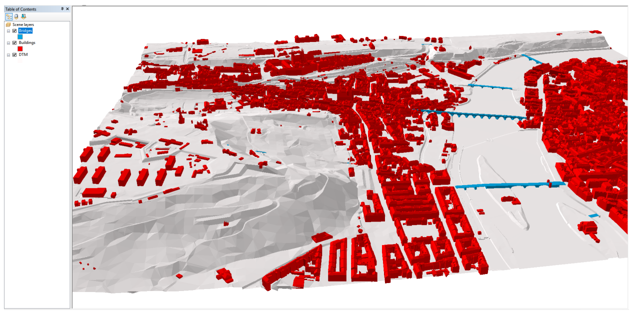


Figure 2. The source data in 3D shapefile proprietary data format visualized in Esri ArcScene (buildings – red, bridges – blue and DTM – grey)

into the CityGML data format (gml files). The source data contained no semantic information (e.g., wall surface/roof surface), only the features' geometries were available. As stated by Biljecki et al. (2016), CityGML is flexible and it does not mandate semantics, for example, an LOD2 with only valid geometry and no semantic differentiation is valid. The transformation is aimed to receive the valid gml files (in terms of CityGML syntax).

3.1 Buildings

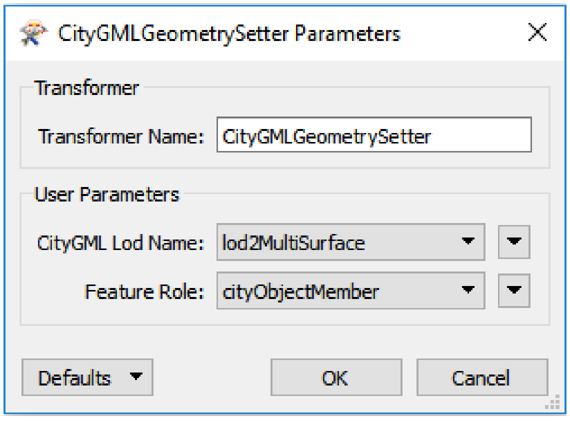
The building model of CityGML is defined by the thematic extension module Building (Gröger et al., 2012). The geometries of buildings in the input file correspond to the LOD2. According to the CityGML standard, in LOD2, the exterior shell of a building can be composed of semantic objects. However, the input data do not contain such semantic information, and therefore, the exterior shell of the building must be explicitly defined, in this case as `gml:MultiSurface` geometry. FME geometries in CityGML features types that must be tagged with their intended geometry role in order to be written out correctly. The FME CityGMLGeometrySetter transformer provides a convenient way to set both the CityGML LOD name and feature role from a pre-set list of options (see Figure 3).

CityGML is an application schema for the Geography Markup Language (GML), and therefore, every feature (like a building or a bridge) is identified by the `gml:id` attribute. It is of XML type ID, so is constrained to be unique in the XML document within which it occurs. To generate the `gml:id` attribute the FME AttributeCreator transformer was used (see Figure 4).

The final transformation schema for the buildings is shown in Figure 5. Figure 6 then depicts a part of the generated CityGML file for the buildings.

3.2 Bridges

The bridge model of CityGML is defined by the thematic extension module Bridge (Gröger et al., 2012). From the LOD perspective, the bridges in the input 3D shapefile correspond to the



CityGMLGeometrySetter Parameters

Transformer

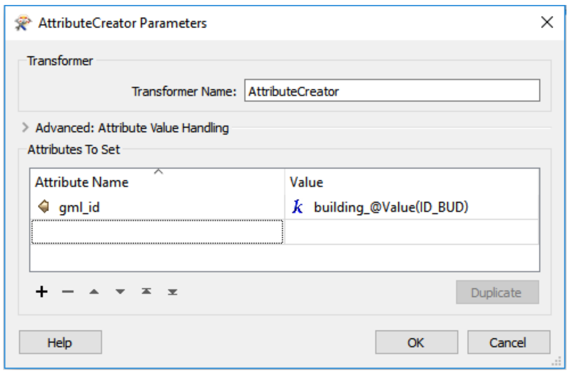
Transformer Name: CityGMLGeometrySetter

User Parameters

CityGML Lod Name: lod2MultiSurface

Feature Role: cityObjectMember

Figure 3. The setting of CityGML LOD name and feature role. According to this particular setting, the geometry of each building is represented in an output file as `gml:MultiSurface` at LOD2.



AttributeCreator Parameters

Transformer

Transformer Name: AttributeCreator

Advanced: Attribute Value Handling

Attributes To Set

Attribute Name	Value
gml_id	building_@Value(ID_BUD)

Figure 4. Generating `gml:id` attribute for buildings. The final value of `gml:id` attribute is based on the feature's identifier contained in the source 3D shapefile (in this case `ID_BUD` attribute).

LOD2. The thematic boundary surfaces of a bridge are defined in analogy to the building module. Also, in case of the bridges, no semantic information was available in the input file, and so, the exterior shell of the bridge must be explicitly defined (here as `gml:MultiSurface` geometry). Also, for each bridge, the attribute `gml:id` is generated based on the bridges' identifiers in the source 3D shapefile. The whole schema for transformation of bridges into CityGML (version 2.0) is captured in Figure 7.

After the conversion is successfully finished, the corresponding CityGML file is generated. A piece of generated CityGML file for bridges is shown in Figure 8.

3.3 Digital Terrain Model

The digital terrain model (DTM) of CityGML is provided by the thematic extension module Relief (Gröger et al., 2012). As stated in the CityGML standard, the terrain is represented by the class `ReliefFeature` (in LOD 0–4), which consists of one or more entities of the class `ReliefComponent`. The terrain may be specified as a regular raster or grid, as a TIN, by break lines or by mass points. Due to the fact, that the input digital terrain model was provided as a 3D triangle mesh, it was decided to represent the terrain as the TIN (class `TINRelief` as a specialization of the class `ReliefComponent`) at LOD2. The transformation schema for the digital terrain model consists of two parts. The first creates a parent element `ReliefFeature`, the latter the child element `TINRelief`.

To create and write the parent element `ReliefFeature`, its `gml:id` (using the AttributeCreator transformer) was generated. Then, all the geometries from the input mesh were aggregated by `gml:id`. Before the element `ReliefFeature` was written into the output gml file, the GeometryRemover transformer had been used. It completely removed the geometry of the `ReliefFeature`.

On the beginning of the second branch, the input mesh is oriented (counter-clockwise direction) using the Orientor transformer. This is done to ensure that the elements of the mesh are visible from above the relief. As the class `ReliefFeature` consists of one or more entities of the class `ReliefComponent`, it is necessary to capture this aggregation during the transformation. In FME, this relationship can be specified through the `gml_id` and `gml_parent_id` attributes. The value of the `gml_parent_id` of the child element (`ReliefComponent`) should be equal to the value of the `gml_id` of its parent element (`ReliefFeature`). For setting of these two elements, the FME AttributeCreator transformer is used (see Figure 9).

The next step was to aggregate all the geometries from the input mesh into instances of `TINRelief`. This was done using the Aggregator transformer. The geometries were aggregated by `gml_id_parent` and `gml_id`. Furthermore, to receive the right type of output geometry, the aggregated geometries were triangularized using the Triangulator transformer.

The geometry role is specified through the geometry trait `citygml_lod_name`. Due to the fact that FME CityGMLGeometrySetter did not support TIN geometry, the AttributeCreator was used again. It enabled to set `citygml_lod_name` as an attribute, and then to transform it into a trait using the GeometryPropertySetter transformer (see Figure 10).

The whole transformation schema for the digital terrain model is shown in the Figure 11. Figure 12 depicts a part of the generated CityGML file for the DTM.

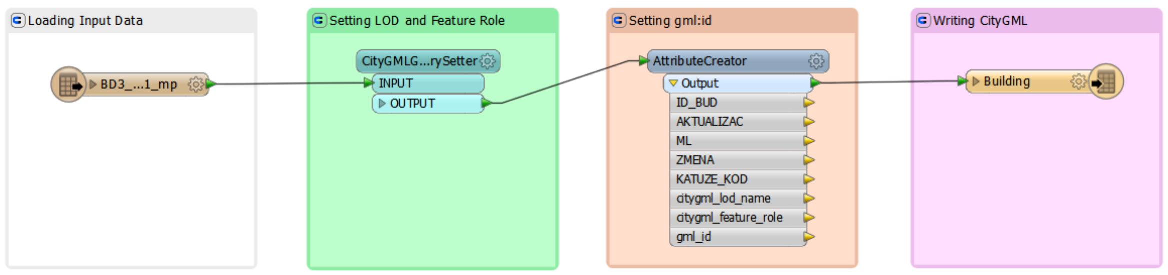
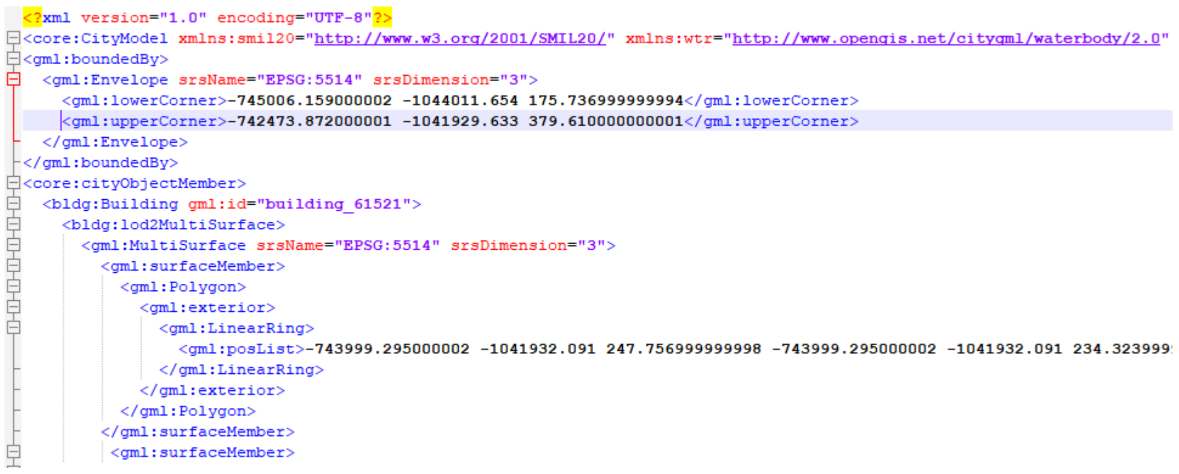


Figure 5. The transformation schema for buildings. After the input data are loaded into the memory, the LOD name and feature role are set. In the next step, the `gml:id` is generated. The conversion ends with writing CityGML into the output file.



```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns:smil20="http://www.w3.org/2001/SMIL20/" xmlns:wtr="http://www.opengis.net/citygml/waterbody/2.0"
<gml:boundedBy>
  <gml:Envelope srsName="EPSG:5514" srsDimension="3">
    <gml:lowerCorner>-745006.159000002 -1044011.654 175.736999999994</gml:lowerCorner>
    <gml:upperCorner>-742473.872000001 -1041929.633 379.610000000001</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<core:cityObjectMember>
  <bldg:Building gml:id="building_61521">
    <bldg:lod2MultiSurface>
      <gml:MultiSurface srsName="EPSG:5514" srsDimension="3">
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>-743999.295000002 -1041932.091 247.756999999998 -743999.295000002 -1041932.091 234.323999
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        <gml:surfaceMember>
```

Figure 6. Generated CityGML file for buildings. The file contains a spatial envelope (`gml:Envelope`), for which, extent is given by the coordinates expressed in requested coordinate reference system (in this case, with EPSG code 5514). The buildings are represented as multisurfaces at LOD2 (`bldg:lod2MultiSurface`).

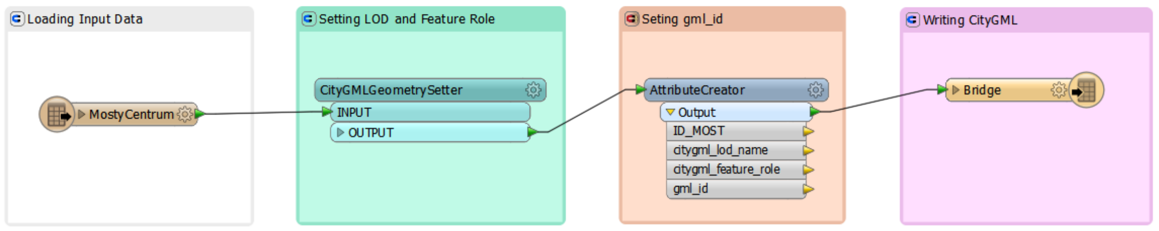
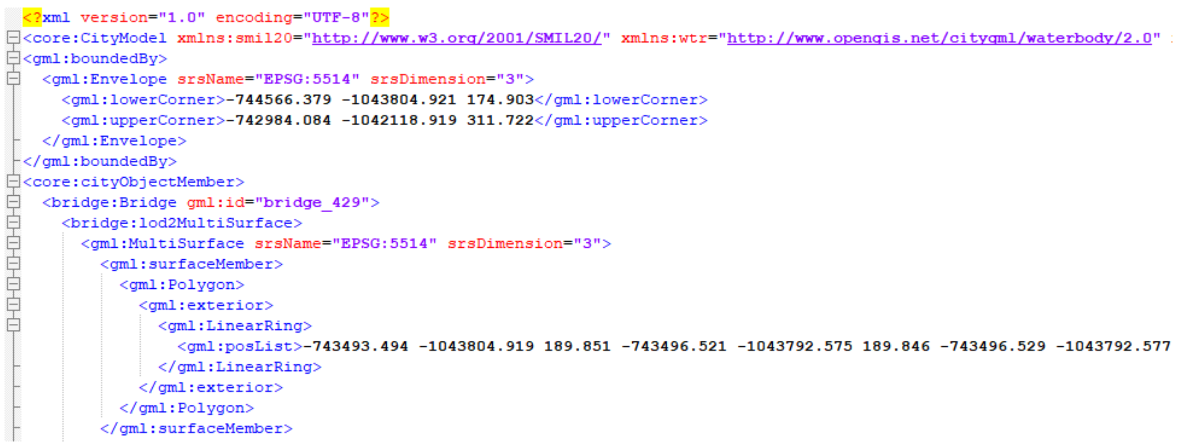
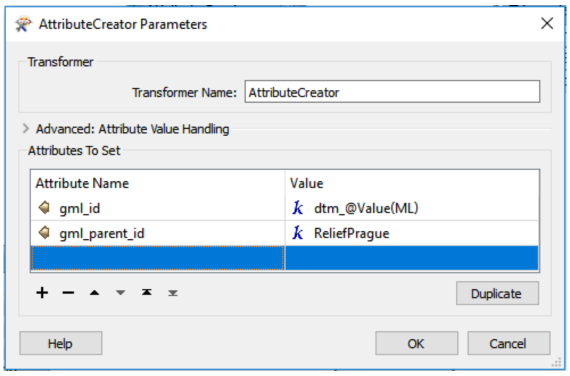


Figure 7. The transformation schema for bridges. First, the input data are loaded into the memory, then the LOD name and feature role are set. In the third step, the `gml:id` is generated. The conversion of bridges ends with writing CityGML into the output file.



```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns:smil20="http://www.w3.org/2001/SMIL20/" xmlns:wtr="http://www.opengis.net/citygml/waterbody/2.0"
<gml:boundedBy>
  <gml:Envelope srsName="EPSG:5514" srsDimension="3">
    <gml:lowerCorner>-744566.379 -1043804.921 174.903</gml:lowerCorner>
    <gml:upperCorner>-742984.084 -1042118.919 311.722</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<core:cityObjectMember>
  <bridge:Bridge gml:id="bridge_429">
    <bridge:lod2MultiSurface>
      <gml:MultiSurface srsName="EPSG:5514" srsDimension="3">
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>-743493.494 -1043804.919 189.851 -743496.521 -1043792.575 189.846 -743496.529 -1043792.577
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
```

Figure 8. Generated CityGML file for bridges. The buildings are represented as multisurfaces at LOD2 (`bldg:lod2MultiSurface`).


AttributeCreator Parameters

Transformer

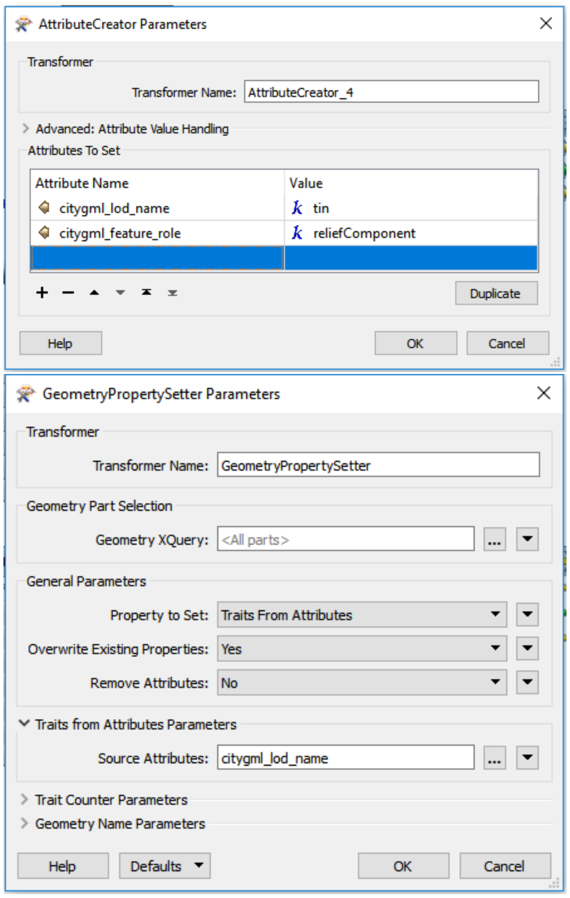
Transformer Name: AttributeCreator

Advanced: Attribute Value Handling

Attributes To Set

Attribute Name	Value
gml_id	dtm_@Value(ML)
gml_parent_id	ReliefPrague

Figure 9. Generating `gml:id` attribute for `TINRelief` features. The value 'ReliefPrague' of the attribute `gml_parent_id` corresponds to the `gml:id` of the parent feature (an instance of the class `ReliefFeature`).


AttributeCreator Parameters

Transformer

Transformer Name: AttributeCreator_4

Advanced: Attribute Value Handling

Attributes To Set

Attribute Name	Value
citygml_lod_name	tin
citygml_feature_role	reliefComponent

GeometryPropertySetter Parameters

Transformer

Transformer Name: GeometryPropertySetter

Geometry Part Selection

Geometry XQuery: <All parts>

General Parameters

Property to Set: Traits From Attributes

Overwrite Existing Properties: Yes

Remove Attributes: No

Traits from Attributes Parameters

Source Attributes: citygml_lod_name

Trait Counter Parameters

Geometry Name Parameters

Figure 10. Creation of the `citygml_lod_name` attribute (top) and its transformation into a trait (bottom)

4 CityGML data validation and database storage

The generated CityGML files were validated in terms of CityGML syntax and also the validity of 3D primitives was explored. These tests are especially important in case that one wants not only to visualize the data, but also to run some spatial analysis (e.g., noise pollution assessment).

To check the syntax, that is, to ensure that the read data are conformant to the OGC standard CityGML 2.0, the tool 3D City Database Importer/Exporter (Yao et al., 2018) was used. This is a Java-based front-end for the 3D City Database and allows for high-performance loading and extracting 3D city model data. All the CityGML files were validated with the result of zero syntax errors.

CityGML uses GML representation of 3D geometries, which are based on ISO 19107 model (ISO/TC 211, 2003). Therefore, for geometric validation of the tool val3dity (Ledoux, 2018) was used. It is an open source software to validate 3D primitives according to the international definitions of ISO 19107. The 3D primitives must be 'watertight', have no intersections, their surfaces are planar and so on. In the generated CityGML files, the volumes are represented with `gml:MultiSurface`, that is, a set of surfaces embedded in 3D, which should form a volume. The val3dity tool took the CityGML file as an input, processed city objects (in this case, buildings and bridges), validated the 3D primitives and outputted a report that helped to identify the errors. Figure 13 contains the information about the geometric validation for CityGML bridges.

For the CityGML buildings, the val3dity tool reported more than 85% of valid features. In the rest, several kinds of geometrical errors were found (i.e., `RING_SELF_INTERSECTION`). Even if some tools for automatic reparation exist (e.g., CityDoctor[5]), due to the complex cases and lack of thematic information, it is appropriate to repair the invalid geometries manually.

To efficiently manage and query the geospatial data, it is sufficient to store them into the spatial database. For storing the CityGML data, the 3D City Database (Yao et al., 2018) was used. It allows virtual 3D city models to be managed on top of a standard spatial relational database. The PostgreSQL with PostGIS option was used.

The 3D City Database's database schema implements the CityGML standard and supports all the thematic modules from CityGML 2.0 and five different LODs. The created CityGML files were successfully imported into the spatial database using the tool 3D City Database Importer/Exporter.

5 Visualization in web map client

The visualization in the web environment enables to disseminate the data to the wider audience. The 3D City Database Importer/Exporter enables to export the stored CityGML data into several data formats suitable for 3D web visualization (KML/KMZ, COLLADA, glTF). As previously mentioned, the used geospatial datasets contain neither semantic information nor the textures. Therefore, the data were exported into the KML and KMZ data formats. Tests have shown shorter loading times (in Google Earth) for the KML format (as opposed to KMZ) when loading from the local hard disk. The Earth Browser's stability also seems to improve when using the uncompressed format. On the other hand, when loading files from a server, KMZ reduces the amount of requests considerably, thus increasing performance (Kolbe et al., 2018).

Together with the fully detailed geometry information also,

5 `https://www.citydoctor.eu` (Accessed 11 January 2019)

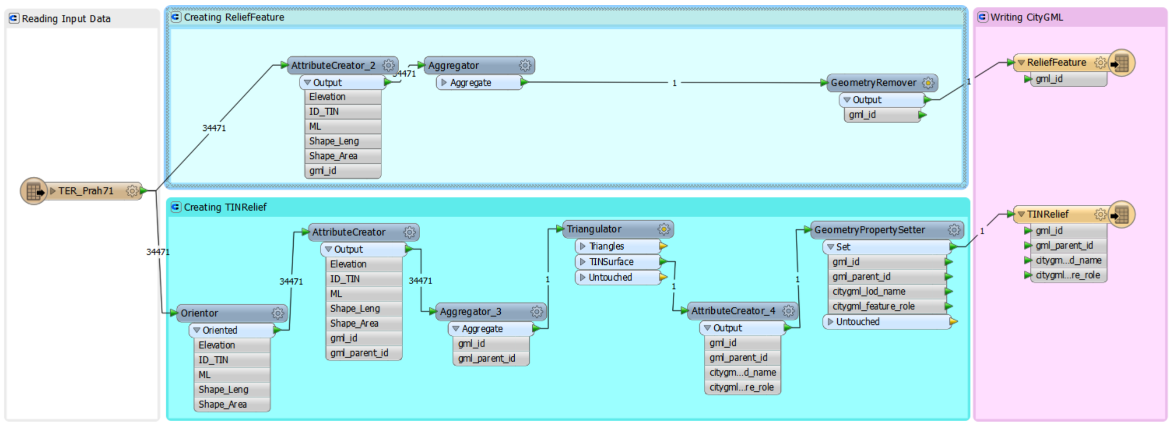


Figure 11. The transformation schema for the digital terrain model. The numbers placed on oriented arrows show the number of features going from one node to another.

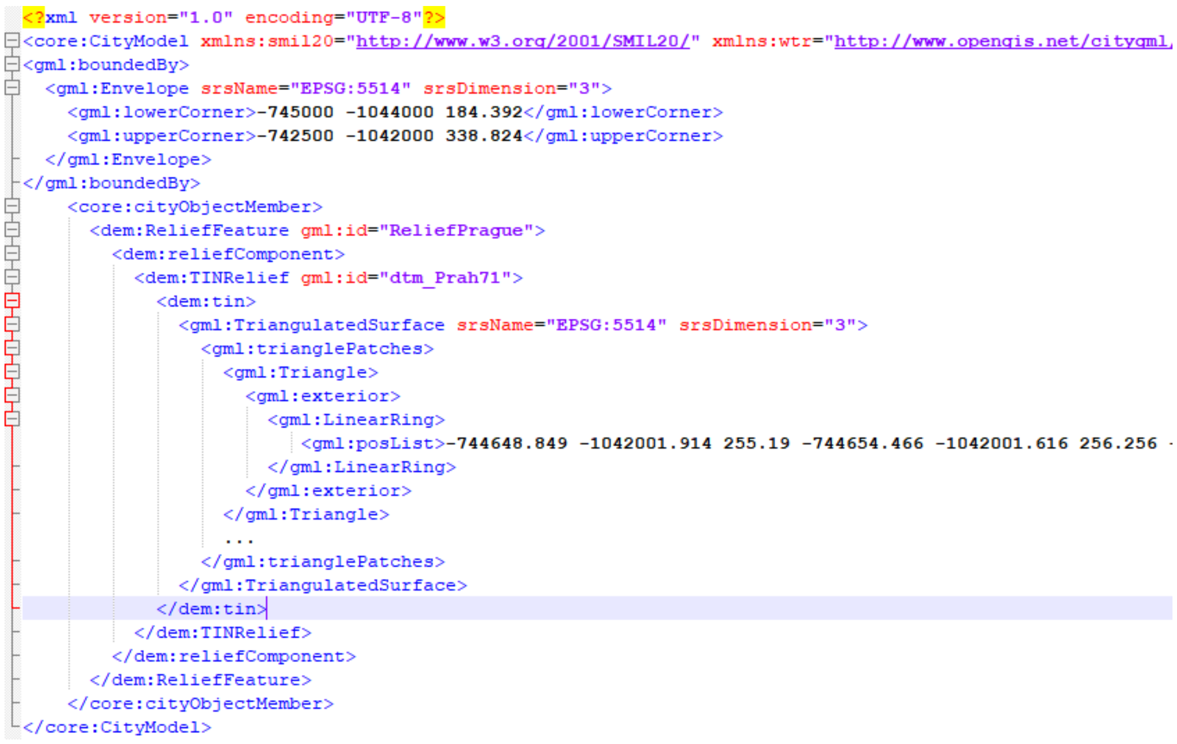
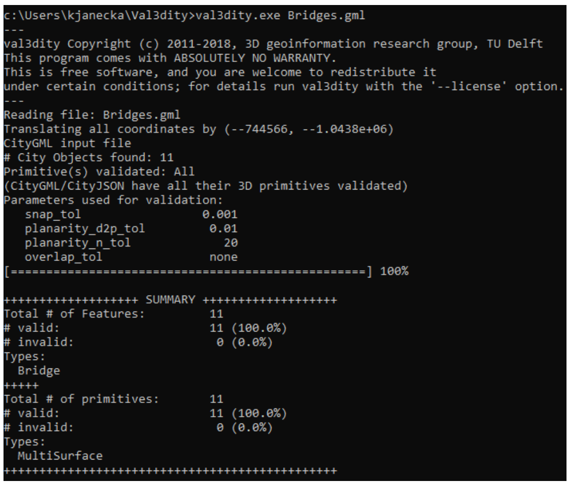

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns:smil20="http://www.w3.org/2001/SMIL20/" xmlns:wtr="http://www.opengis.net/citygml,
<gml:boundedBy>
  <gml:Envelope srsName="EPSG:5514" srsDimension="3">
    <gml:lowerCorner>-745000 -1044000 184.392</gml:lowerCorner>
    <gml:upperCorner>-742500 -1042000 338.824</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
    <core:cityObjectMember>
      <dem:ReliefFeature gml:id="ReliefPrague">
        <dem:reliefComponent>
          <dem:TINRelief gml:id="dtm_Prah71">
            <dem:tin>
              <gml:TriangulatedSurface srsName="EPSG:5514" srsDimension="3">
                <gml:trianglePatches>
                  <gml:Triangle>
                    <gml:exterior>
                      <gml:LinearRing>
                         <gml:posList>-744648.849 -1042001.914 255.19 -744654.466 -1042001.616 256.256 ·
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Triangle>
                  ...
                </gml:trianglePatches>
              </gml:TriangulatedSurface>
            </dem:tin>
          </dem:TINRelief>
        </dem:reliefComponent>
      </dem:ReliefFeature>
    </core:cityObjectMember>
</core:CityModel>
```

Figure 12. Generated CityGML file for relief represented as TIN (element `dem:TINRelief`)



```
c:\Users\kjanecka\Val3dity>val3dity.exe Bridges.gml
---
val3dity Copyright (c) 2011-2018, 3D geoinformation research group, TU Delft
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; for details run val3dity with the '--license' option.
---
Reading file: Bridges.gml
Translating all coordinates by (--744566, --1.0438e+06)
CityGML input file
# City Objects found: 11
Primitive(s) validated: All
(CityGML/CityJSON have all their 3D primitives validated)
Parameters used for validation:
   snap_tol                0.001
   planarity_d2p_tol        0.01
   planarity_n_tol            20
   overlap_tol              none
[==================================================] 100%

++++++++++++++++++ SUMMARY ++++++++++++++++++
Total # of Features:         11
# valid:                     11 (100.0%)
# invalid:                    0 (0.0%)
Types:
  Bridge
+++++
Total # of primitives:       11
# valid:                     11 (100.0%)
# invalid:                    0 (0.0%)
Types:
  MultiSurface
+++++++++++++++++++++++++++++++++++++++++++++
```

Figure 13. The geometric validation for CityGML bridges using the val3dity tool. The summary states that all bridges are geometrically valid.

the thematic surfaces were exported. This was due to the reason, that the exporting tool follows a logic, that surfaces touching the ground (having lowest z-coordinate) were considered as wall surfaces and all other were considered as roof surfaces.

For the 3D web visualization of the exported CityGML data, the 3DCityDB-Web-Map-Client was used. The 3DCityDB-Web-Map-Client is 3DCityDB's front-end for web-based 3D visualization and interactive exploration of arbitrarily large semantic 3D city models (Yao et al., 2018). It has been developed based on the Cesium Virtual Globe[6], which is an open source JavaScript library.

As the amount of 3D data to be visualized can be huge, the 3D City Database Importer/Exporter tool enables a 'tiling' option. It means that the area enclosed by the bounding box will be exported in tiles and then only the tiles in the vicinity of the viewer's location are loaded. The tiles can be created manually or automatically. The latter option was used, the side length for automatic tiling was set 125.0 meters. Given the size of the testing area, 17 $\times$ 21 tiles were generated.

To test the functionality of the 3D web client on a local machine, the 3D web client offers a lightweight JavaScript-based HTTP server (Kolbe et al., 2018). The server was run on the local machine with configuration Intel® Core™ i5-82500 CPU @ 1.60 GHz, 16 GB RAM, Windows 10 64 bit, Intel HD Graphics 620. Figure 14 shows the exported 3D geospatial data after their loading into the web map client.

The used 3DCityDB-Web-Map-Client enables to attach the additional attribute information to the displayed 3D geospatial data using their `gml:id` from an external data source. In particular, the Fusion Tables, which is a web service provided by Google for data management, was used with aim to display the information about the owner of the selected 3D object. The fusion table containing the information about the owners was created manually; however, it could be also fulfilled automatically from the appropriate information system. Figure 15 displays the information about owner of the selected 3D object; in this case, it was Charles Bridge. The proposed solution could be used, for example, for building the 3D cadastre (Janečka, 2018).

6 https://cesiumjs.org/

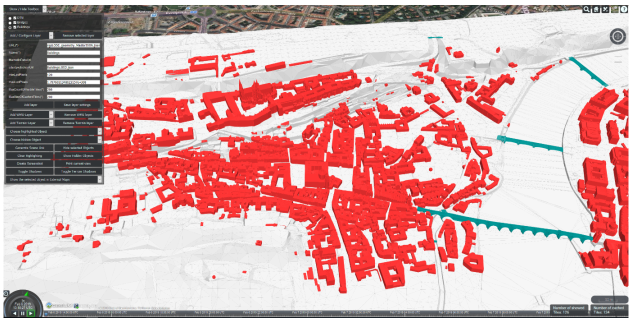


Figure 14. The 3D geospatial data covering the Prague city centre visualized in the 3DCityDB-Web-Map-Client

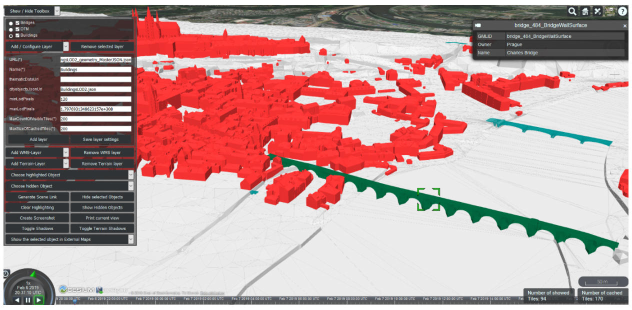


Figure 15. For 3D objects, their owners can be displayed. Charles Bridges is owned by Prague (see the info box in the right upper corner)

6 Conclusions

The 3D geoinformation is becoming important for cities and their policies. Incorporating 3D geospatial data into spatial analysis leads, in many cases, to more efficient decisions. The cities are therefore investing resources to obtain the 3D virtual city models. To maximize the economic benefit of such data, the cities provide their 3D geospatial data for further usage, and so, new applications can be created.

The paper explored how could such freely available 3D geospatial data be used for the creation of 3D city model consisting of selected 3D objects (buildings, bridges, terrain). The transformation from the proprietary 3D shapefile format into the CityGML standard was described in detail. Table 1 gives a comparison between the size of input files (3D shapefiles), generated CityGML files (gml) and exported KML/KMZ files used for 3D visualization in the web environment. It was expected that the XML based data formats (CityGML, KML) will be larger than the input 3D shapefile. To increase the performance of the 3D web visualization, the KMZ file format could be used to decrease the amount of loaded data.

The generated CityGML files were imported into the spatial database with appropriate database CityGML-based scheme. It enables more efficient management and querying of CityGML data. The applications working the 3D city model generally require more semantic information than the input 3D shapefile contained. For example, the buildings could be structured

Table 1. The comparison of the size of files

Theme	Input 3D shapefile	CityGML (gml)	KML/KMZ (inc. tiles)
Bridges	0.5 MB	1 MB	27.6 MB/1.5 MB
Buildings	36 MB	80.5 MB	166.6 MB/13.1 MB
Relief	19 MB	9 MB	13 MB/0.7 MB

according to the type of surface (e.g., roof surface). The enrichment of the generated CityGML files for this semantic information is the subject of further research, as well as an automatic reparation of invalid geometries. The type of surfaces, that is, ground surfaces or wall surfaces, could be distinguished based on numeric calculations. For example, it can be supposed that the ground surfaces consist of the vertices with the smallest value of Z-coordinate. The surfaces can be divided into the line segments and then the segments containing the smallest value of Z-coordinate will form the ground surface. An important issue in 3D city modelling is the integration of 3D objects and the terrain; another issue that should be explored is the Terrain Intersection Curve (TIC) as described in the CityGML standard. These curves denote the exact position, where the terrain touches the 3D object, for example, a building or a bridge.

Acknowledgement

The publication was supported by the Project LO1506 of the Czech Ministry of Education, Youth and Sports.

References

Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, doi:10.1016/j.compenvurbsys.2016.04.005.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, doi:10.3390/ijgi4042842.

Floros, G. and Dimopoulou, E. (2016). Investigating the enrichment of a 3D city model with various CityGML modules. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, XLII-2/W2, doi:10.5194/isprs-archives-XLII-2-W2-3-2016.

Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium.

ISO/TC 211 (2003). *Geographic information — Spatial schema*. (ISO 19107:2003).

Janečka, K. (2018). 3D cadastre — a motivation and recent developments of technical aspects. In *Proceedings of scientific conference STEPGRAD*, volume 13.

Kolbe, T. H., Burger, B., and Cantzler, B. (2015). CityGML goes to Broadway. In *Photogrammetric Week '15*, pages 343–356.

Kolbe, T. H., Nguyen, S. H., Chaturvedi, K., Willenborg, B., Donaubauer, A., Nagel, C., Yao, Z., Schulz, H., Willkomm, P., Hudra, G., and Kunde, F. (2018). *3D City Database for CityGML*. Documentation, version 4.0.

Ledoux, H. (2018). val3dity: validation of 3D GIS primitives according to the international standards. *Open Geospatial Data, Software and Standards*, 3(1):1, doi:10.1186/s40965-018-0043-x.

Ohori, K. A., Biljecki, F., Kumar, K., Ledoux, H., and Stoter, J. (2018). Modeling cities and landscapes in 3D with CityGML. In *Building Information Modeling*, pages 199–215. Springer.

Saran, S., Oberai, K., Wate, P., Konde, A., Dutta, A., Kumar, K., and Kumar, A. S. (2018). Utilities of virtual 3D city models based on CityGML: Various use cases. *Journal of the Indian Society of Remote Sensing*, 46(6):957–972, doi:10.1007/s12524-018-0755-5.

Willenborg, B., Sindram, M., and Kolbe, T. H. (2018). Applications of 3D city models for a better understanding of the built environment. In *Trends in Spatial Analysis and Modelling*, pages 167–191. Springer.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., and Kolbe, T. H. (2018). 3DCityDB — a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards*, 3(1):5, doi:10.1186/s40965-018-0046-7.