

Towards a Semantic 3D Model of Sofia City

Dessislava Petrova-Antonova[1,2(✉)] and Angel Spasov[1]

[1] Sofia University "St. Kliment Ohridski", GATE Institute, 1113 Sofia, Bulgaria
{dessislava.petrova,angel.spasov}@gate-ai.eu
[2] Sofia University "St. Kliment Ohridski", FMI, 1164 Sofia, Bulgaria

Abstract. More and more cities worldwide are working on their digital transformation in order to deliver better living environment for citizens. Over the last decades, a lot of research efforts are dedicated to development of semantic 3D city models, covering variety city dimensions based on standards such as CityGML. The complexity of cities' processes, the heterogeneity of data they produced and different special scale requirements for planning and managing urban environment bring emerging challenges to implementation of semantic 3D city models. In order to address these challenges, this paper presents the process of development a semantic 3D model of the city of Sofia and its visual representation through a simple web application. As a pilot area, District "Vazrajdane" is selected, consisting of several neighbourhoods close to the center of the city. The 3D transformation of proprietary geospatial data into CityGML schema, covering building and terrain modules, is described. Further, the implementation of the web application for user interaction with the semantic 3D city model is presented. Its main features include silhouetting a single building on handling mouse events and showing relevant overlay content, displaying shadows and styling of buildings depending on their data attributes.

Keywords: Semantic 3D city model · CityGML · Buildings and terrain · FME · Cesium

1 Introduction

The semantic 3D models of cities have significantly advanced recently, especially with the adoption of the Digital Twin's concept. They provide a solid foundation for a wide range of urban analysis and simulations. Since such models go far beyond the simple visualisation, they are used for a variety of applications such as disaster management, urban planning and design, solar potential analysis, air pollution simulation, etc. [1, 2].

Currently, more than a thousand city models exist worldwide [3]. The majority of them are implemented using either the CityGML or IFC standard [4, 5]. A number of techniques can be applied for their development such as aerial photographs and laser scanning [6, 7], extrusion from 2D footprints [8], airborne point clouds [9–11], architectural drawings and plans [12, 13], procedural modelling [14, 15], and open street map [16, 17]. The 3D models, based on CityGML, represent the cityscape with respect to geometry, topology, semantics and appearance of common urban objects. They can be

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12952, pp. 230–241, 2021.
https://doi.org/10.1007/978-3-030-86973-1_16

modelled in five different Levels of Detail (LOD) with respect to their geometry and spatial accuracy [18]. As a widely adopted standard, CityGML enables reusability and interoperability of 3D models over different applications. Till now, it is mostly used for modelling of the buildings due to their dominant role within the urban environment and lack of data for other thematic objects such as road infrastructure, underground networks and utilities, water bodies, etc. [19]. CityGML influenced the INSPIRE Directive of the European Commission (EC), which aims at the creation of a European Union spatial data infrastructure providing public sector data in an interoperable way [20]. In addition, a lot of commercial and open-source tools provide support for CityGML standard. For example, citygml4j is a Java class library for processing CityGML models developed by the Technische Universität Berlin. The 3D City Database is a free 3D geo-database to store and manage 3D city models on top of Oracle Spatial or PostgreSQL/PostGIS database management systems [21]. FME (Feature Manipulation Engine) from Safe Software Inc. provides a rich functionality for transformation of city data in a variety of formats in CityGML [22]. Bentley Map combines the strengths of CAD and traditional GIS providing capabilities for 2D or 3D analysis and visualizations [23]. Solar3Dcity, a simple tool developed by TU Delft, assesses the yearly solar irradiation of buildings modeled in CityGML [24]. Cesium Ion and CesiumJS enable a CityGML upload and visualization in the Cesium Globe. Another tool for visualization of 3D models is FZKViewer from Karlsruhe Institute of Technology, supporting both for IFC and CityGML standards [25]. The number of such tools is steadily growing, especially in the context of rapid development of city digital twin platforms.

This paper presents the preliminary results from creating a CityGML-compliant semantic 3D model of the city of Sofia. It explains the 3D transformation of proprietary geospatial data into CityGML schema and its further visualization in a web application. First, a semantic 3D model of buildings and terrain at district scale is created. It covers the territory of District "Vazrajdane" of Sofia, which almost entirely belongs to the city centre. Next, a simple web application is developed for visualisation and user interaction with the 3D model of the buildings. The semantic 3D city model and its visual representation are the first step towards building of a digital twin platform, which will be used for planning, design, exploration, experimentation and optimization of urban processes and services.

The rest of the paper is organised as follows. Section 2 points out data used and describes the process of model creation in the current use case. Section 3 explains in detail workflow and implementation for visualization. Section 4 summarizes the paper and extends on future work.

2 3D Modelling

This section describes geospatial data transformation with the help of FME software, which is a widely used tool for data elevation by extruding 2D geometry to 3D models [26]. It provides pre-built transformation components that accelerate the process of workflow creation. The goal of the transformation is to produce 3D city model compliant with CityGML standard in LOD1.

2.1 Data Sources

The geospatial data were provided for research purposes by Sofiaplan, which is a municipal enterprise responsible for the spatial and strategic planning of Sofia Municipality. Table 1 provides description of the data sources.

Table 1. Description of data sources

Type object	File type	File size	Number of objects	Description
Buildings	.SHP	1063 KB	5090	Building footprints, floor count and function
Digital terrain	.TIF	953 KB	1	Digital terrain

The buildings' data is stored in PostGIS database and exported in.shp format. In addition, a digital terrain model was provided in.tif format. Figure 1 shows the visualisation of source data in QGIS. The relief is presented with grey colour, while the buildings' footprints are coloured in blue. The coordinate reference system of source data is BGS2005/CCS2005 (Bulgaria Geodetic System 2005, EPSG: 7801).

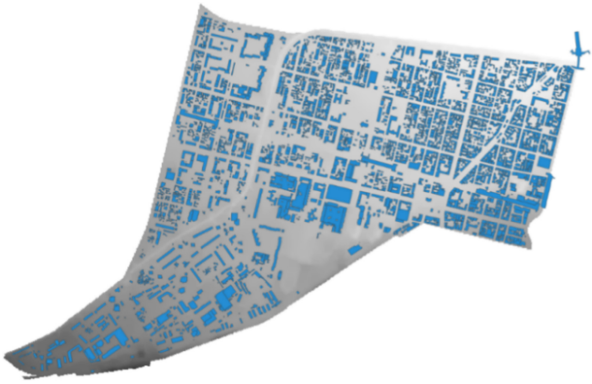

Fig. 1. Visualisation of source data.

2.2 Building Modelling

Each LOD defined in CityGML standard is characterized by differing accuracies and minimal dimensions of the corresponding objects. The buildings are modelled based on the thematic module Building. In contrast to higher level of details, in LOD1 buildings are represented as a general geometry of the outer shell where volume and surface are identical. Furthermore, different structural parts are combined into a simple block [18], including roofs which receive a flat representation. The absolute 3D point accuracy is

required to be 5/5 m. The interior structure of the buildings is not part of the LOD1 model.

The geometric representation can be implemented by only one of the properties *lod1Solid* and *lod1MultiSurface*. For the current model lod1Solid property was selected to be implemented, including the following building's attributes: *class, function, measuredHeight* and *storeysAboveGround*. The CityGML class and function attributes of the buildings are obtained from their function property in the dataset. The function code types of the building in the dataset are mapped to the function and class code types in the CityGML standard. As a result, an additional.xlsx file is created, which contains the CityGML function and class attributes of the buildings.

The CityGML standard does not provide specifications and instructions how objects can be geometrically modelled [27]. The *measuredHeight* attribute of the building is the measured or computed difference between the lowest terrain intersection point and the highest roof point. Figure 2 shows different valid heights for different roof overhangs. Since the roof could overhang, roof edges could have a lower elevation than the highest point of the walls. The current implementation of the model considers heights at the roof eaves.



Fig. 2. Different valid representations of building height in LOD1 [28].

The heights of the buildings are not presented in the dataset and therefore the footprints are extruded to their respective heights from the number of floors, depending on the class attribute. Based on this logic *measuredHeight* attribute is preliminary calculated and included in.xlsx file. For example, a building having three floors and a CityGML class "habitation" has received a value of nine meters for *measuredHeight* attribute. Consequently, the 2D footprints can be transformed into 3D solid geometries.

The input data, delivered in.shp and.xlsx files, is merged and transformed in several steps. The FME workflow is depicted in the Fig. 3.

The input data is joined using *FeatureJoiner* transformer. The join is based on the Id property, which matches both input files in one-to-one relation. Next, the Extruder transformer is used to create solid geometries, based on the approximated building heights. In this way, the 2D data is extruded into 3D. Unnecessary attributes in the input data are cleaned using *AttributeKeeper* transformer. Thus, only CityGML relevant data is passed to the next step. FME allows for simple sampling with the Sampler transformer. Within the implemented workflow, it is used to speed up testing. Further, a new attribute "*gml:id*" is added. This is accomplished using the "*AttributeCreator*" transformer, which assigns unique identifiers based on the Id attribute. Next, the buildings' geometries are

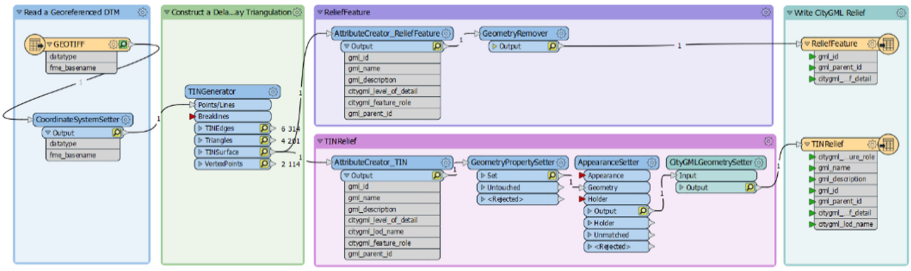

Fig. 3. FME workflow for building modelling.

set up using the *CityGMLGeometrySetter* transformer. The CityGML LOD name is set to "*lod1Solid*" and the feature role is set to "*cityObjectMember*".

The last step of the workflow is the reprojection of the geometries to the desired coordinate system. This is needed since the local footprint polygons are usually projected in a different coordinate system than the base map used for visualisation. Finally, a Writer is used to generate CityGML output for the buildings. A 3D view of the obtained result is shown in Fig. 4.



Fig. 4. A 3D view of the buildings.

2.3 Terrain Modelling

CityGML standard supports Digital Terrain Model (DTM) through the thematic module *Relief*. The terrain is modelled using *ReliefFeature* class and could be represented with different concepts – a regular raster or grid, break lines, a Triangular Irregular Network (TIN) or mass points. For the current model the TIN concept is selected, which represents the relief as triangles, implemented by the *TinRelief* class.

The FME workflow for generation of CityGML compliant relief model is shown in Fig. 5. The input data is provided in.tiff format. The transformation starts with construction of a Delaunay triangulation using *TINGenerator* transformer. Next, it is split into two chains. The first chain produces an instance of *ReliefFeature* class. The *AttributeCreator* transformer is used for generation of gml_id, gml_name, gml_description,

city_gml_level_of_detail, citygml_feature_role and gml_parent_id attributes for *ReliefFeature*. The *GeometryRemover* transformer uses the gml:id attribute of *ReliefFeature* for aggregation of all geometries from the input file.

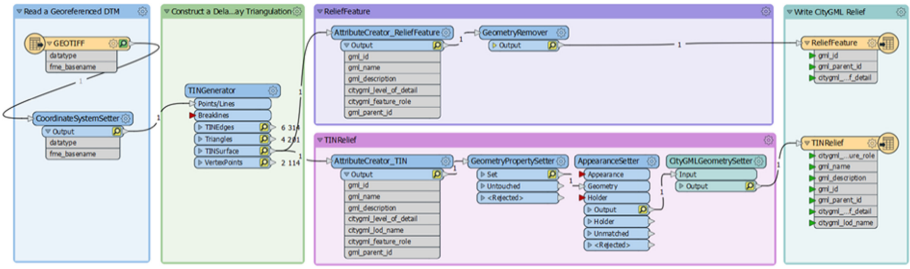

Fig. 5. FME workflow for terrain modelling

The second chain produces an instance of *TinRelief* class. The *AttributeCreator* transformer is used again to set values for gml_id, gml_name, gml_description, city_gml_level_of_detail, citygml_lod_name, city_gml_feature_role and gml_parent_id attributes. The relation between *ReliefFeature* and *TinRelief* instances is specified through gml_id and gml_parent_id attributes. The citygml_lod_name attribute is used to set a trait by *GeometryPropertySetter* transformer. The *AppearanceSetter* transformer sets appearance style of geometries. The *CityGMLGeometrySetter* transformer sets the CityGML LOD name to "tin" and the feature role to "ReliefComponent". Finally, a Writer is used to generate CityGML output for the buildings. A 3D view of the obtained result is shown in Fig. 6.

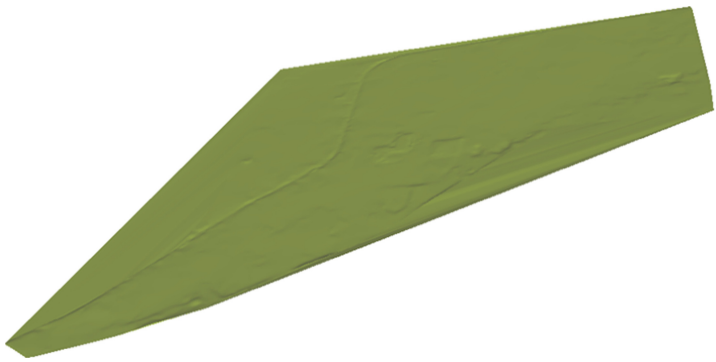

Fig. 6. A 3D view of relief.

2.4 Validation

The quality of semantic 3D models is essential, especially when they will be used for analysis and simulations. The presented semantic 3D models of the buildings and

terrain are validated by open-source tools. 3DCityDB Importer/Exporter is used to check whether the models are actually compliant with CityGML 2.0 schema and can be further imported into 3DCityDB on top of PostGIS. FZK Viewer is useful to check if the 3D models are delivered to the correct CityGML theme. In the current case, such validation is not needed, since the objects in the input files belongs to a concrete CityGML theme. Val3dity is used for validation of the geometries and semantics.

The implemented FME workflows are reusable, since they could be applied for modelling of another urban area. If the input.shp and.tif files are changed, then 3D models of terrain and buildings can be easily produced for different districts as well as for the whole city.

3 Visualisation

A web application is developed for visualizing the CityGML buildings model of district "Vazrajdane". Cesium ion platform is used for streaming and hosting the 3D content, namely the CityGML building model. The hosted 3D content is accessed through its AssetId. CesiumJS is used to visualize the 3D content that is loaded by Cesium ion platform. The web application is hosted on a local web server, which is set up with Node.js – an asynchronous event-driven JavaScript runtime [29]. It is developed as a single HTML page as follows. First, a Cesium Widget is implemented using an ordinary div element (see Listing 1).

Listing 1

```
<div id="cesiumContainer" class="fullSize"></div>
```

Next, an instance Cesium Viewer is created (see Listing 2) and attached to the Cesium Widget. Cesium Viewer provides an interactive 3D globe with rich functionality.

Listing 2

```
var viewer = new Cesium.Viewer("cesiumContainer");
```

In order to visualize the 3D content in the Cesium Viewer, an instance of Cesium3DTileset is created and initialized with the AssetId of the 3D content hosted on Cesium ion platform as it is shown in Listing 3.

Listing 3

```
var tileset = viewer.scene.primitives.add(
  new Cesium.Cesium3DTileset({
    url: Cesium.IonResource.fromAssetId(155869),
  })
)
```

The following functionality is implemented for user interaction:

- Silhouette a building on mouseover and show its class as overlay content;
- Silhouette a building on selection and show its class, function, floor count and height in an information box;

- Show shadows depending on the current time;
- Show buildings in different colours depending on their height, class and latitude;
- Show buildings in transparent style;
- Show buildings with height over 50 m.

A HTML overlay for showing building's class on mouseover is created and appended to the Cesium Viewer. The corresponding code is shown in Listing 4.

Listing 4

```
var nameOverlay = document.createElement("div");
viewer.container.appendChild(nameOverlay);
var name = pickedFeature.getProperty("citygml_class_description");
nameOverlay.textContent = name;
```

In order to silhouette a building, a post-process stage that detects edges is created. The length in pixels and colour of edge are initialised as it is shown in Listing 5.

Listing 5

```
var silhouetteBlue = Cesium.PostProcessStageLibrary.
                              createEdgeDetectionStage();
silhouetteBlue.uniforms.color =
                            Cesium.Color.MEDIUMSLATEBLUE;
silhouetteBlue.uniforms.length = 0.01;
```

The styles of the buildings are implemented through instances of the Cesium3DTileStyle. When default style is applied, the buildings are coloured in white (see Listing 6).

Listing 6

```
var defaultStyle = new Cesium.Cesium3DTileStyle({
        color : "color('white')",
        show : true
    });
```

Additional 3 instances of Cesium3DTileStyle are created for colouring the building by height, by class and by latitude. A separate instance of Cesium3DTileStyle is created for transparent style of the buildings (see Listing 7).

Listing 7

```
var transparentStyle = new Cesium.Cesium3DTileStyle({
        color : "color('white', 0.3)",
        show : true
    });
```

In order to show the buildings with heigh over 50 m, a conditional style is defined as it is shown in Listing 8.

Listing 8

```
var hideStyle = new Cesium.Cesium3DTileStyle({
        show: "${Height} > 50"
    });
```

The shadows if the buildings are shown through setting the *shadow* property of the Cesium Viewer (see Listing 9).

Listing 9

```
var shadowsElement = document.getElementById('shadows');
 shadowsElement.addEventListener('change', function (e) {
        viewer.shadows = e.target.checked;
    });
```

Figure 7 shows the web application for visualization of the buildings' model.



Fig. 7. Buildings' model visualization.

The web application is designed to work independently of the actual CityGML buildings model. If the same model is implemented for another urban area, the only the AssetId of the 3D content hosted on Cesium ion platform needs to be changed. Since

Cesium ion is built on a modern cloud architecture, large datasets can be efficiently streamed into the web application. Thus, the proposed solution could be easily scaled at city level.

4 Conclusions and Future Work

The paper presents the process of creating a semantic 3D city model and its visual representation. The transformation workflow of proprietary data into a CityGML schema using FME was described in detail. The semantic 3D city model covers the buildings and relief thematic modules of CityGML standard. It is hosted on Cesium ion platform and visualized through a simple web application, developed with CesiumJS. The web application provides functionality for user interaction, including silhouetting a building on mouse hover and mouse click and showing overlay content, displaying shadows and styling of buildings.

As a standard CityGML is applicable for areas with various size, and can represent 3D urban objects and terrain in different levels of detail simultaneously. The 3D models, based on CityGML could be simple, single scale models as those presented in the paper as well as very complex multi-scale models. Since, CityGML standardizes the representation of 3D urban objects, providing data consistency, the respective 3D city models could be used in various application domains such as urban and landscape planning, disaster management, environment simulations, vehicle and pedestrian navigation, etc. For example, the 3D city models in high level of detail deliver true picture and real scene of the urban environment, enabling urban planners explore the 3D urban objects in an intuitive and user-friendly way as well as to simulate different planning and design scenarios.

In future work, a computation of *Terrain Intersection Curve* (TIC) is considered. The most commonly used method, including buildings' footprints and DTM, will be applied. A special attention should be given on parts of the terrain without having enough points in between footprints. In such cases, the terrain could be refined through additional terrain points obtained from the original terrain or by integrating footprints of new objects such as trees and streets [30].

Acknowledgement. This work has been supported by GATE project, funded by the Horizon 2020 WIDESPREAD-2018-2020 TEAMING Phase 2 programme under grant agreement no. 857155, by Operational Programme Science and Education for Smart Growth under Grant Agreement No. BG05M2OP001-1.003-0002-C01 and by the Bulgarian National Science fund under project Big4Smart agreement no. DN12/9.

References

1. Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A.: Applications of 3D city models: state of the art review. ISPRS Int. J. Geo-Information **4**, 2842–2889 (2015)
2. Julin, A., et al.: Characterizing 3D city modeling projects: towards a harmonized interoperable system. ISPRS Int. J. Geo-Information **7**, 55 (2018)

3. Morton, P., Horne, M., Dalton, R., Thompson, E.: Virtual city models: avoidance of obsolescence. In: Proceedings of the 30th eCAADe Conference In Digital Physicality, Prague (2012)
4. Li, W., Zlatanova, S., Diakite, A., Aleksandrov, M., Yan, J.: Towards integrating heterogeneous data: a spatial DBMS solution from a CRC-LCL project in Australia. ISPRS Int. J. Geo-Information **9**, 63 (2020)
5. Arroyo Ohori, K., Biljecki, F., Kumar, K., Ledoux, H., Stoter, J.: Modeling cities and landscapes in 3D with CityGML. In: Borrmann, A., König, M., Koch, C., Beetz, J. (eds.) Building Information Modeling, pp. 199–215. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92862-3_11
6. Blaschke, T.: Object based image analysis for remote sensing. ISPRS J. Photogramm. Remote Sens. **65**, 2–16 (2010)
7. Tomljenovic, I., Höfle, B., Tiede, D., Blaschke, T.: Building extraction from airborne laser scanning data: an analysis of the state of the art. Remote Sens. **7**, 3826–3862 (2015)
8. Arroyo Ohori, K., Ledoux, H., Stoter, J.: A dimension-independent extrusion algorithm using generalised maps. Int. J. Geogr. Inf. Sci. **29**, 1166–1186 (2015)
9. Shahzad, M., Zhu, X.: Robust reconstruction of building facades for large areas using spaceborne TomoSAR point clouds. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **53**, 752–769 (2015)
10. Soon, H., Khoo, V.H.S.: Citygml modelling for singapore 3D national mapping. ISPRS – Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. **42**, 37–42 (2017)
11. Wang, Y., Cheng, Y., Zlatanova, S., Palazzo, E.: Identification of physical and visual enclosure of landscape space units with the help of point clouds. Spat. Cogn. Comput. **20**(3), 257–279 (2020)
12. Yin, X., Wonka, P., Razdan, A.: Generating 3D building models from architectural drawings: a survey. IEEE Comput. Graph. Appl. **29**, 20–30 (2009)
13. Lewis, R., Séquin, C.: Generation of 3D building models from 2D architectural plans. Comput. Aided Des. **30**, 765–779 (1998)
14. Smelik, R., Tutenel, T., Bidarra, R., Benes, B.: A Survey on procedural modelling for virtual worlds. IEEE Comput. Graph. Appl. **33**, 31–50 (2014)
15. Besuievsky, G., Patow, G.: Recent advances on LoD for procedural urban models. In: Proceedings of the 2014 Workshop on Processing Large Geospatial Data, Cardiff, UK (2014)
16. Goetz, M.: Towards generating highly detailed 3D CityGML models from OpenStreetMap. Int. J. Geogr. Inf. Sci. **27**, 845–865 (2013)
17. Over, M., Schilling, A., Neubauer, S., Zipf, A.: Generating web-based 3D city models from OpenStreetMap: the current situation in Germany. Comput. Environ. Urban Syst. **34**, 496–507 (2010)
18. Gröger, G., Kolbe, T., Nagel, C., Häfele, K.: OGC City Geography Markup Language (CityGML) Encoding Standard. Open Geospatial Consortium, Wayland, MA, USA (2012)
19. Beil, C., Kolbe, T.H.: Citygml and the streets of New York - a proposal for detailed street space modelling. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. **44**, 9–16 (2017)
20. European Commission: INSPIRE Directive. https://inspire.ec.europa.eu/about-inspire/563. Accessed 20 Mar 2021
21. Yao, Z., Nagel, C., Kunde, F., et. al.: 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. Open Geospatial Data Softw. Stand. **3**, 1–26 (2018). https://opengeospatialdata.springeropen.com/track/pdf/10.1186/s40965-018-0046-7.pdf
22. Safe Software: FME Desktop. https://www.safe.com/fme/fme-desktop/. Accessed 22 Jan 2021
23. Bentley OpenCities Map 2D/3D Desktop GIS and Mapping Software. https://www.bentley.com/en/products/product-line/asset-performance/opencities-map. Accessed 22 Jan 2021

24. TU Delft: Solar3Dcity GitHub repository. https://github.com/tudelft3d/Solar3Dcity. Accessed 22 Jan 2021
25. Karlsruhe Institute of Technology: FZKViewer details. https://www.iai.kit.edu/english/1648.php. Accessed 22 Jan 2021
26. Safe Software: FME Software. https://www.safe.com/fme/. Accessed 22 Jan 2021
27. Biljecki, F., Ledoux, H., Stoter, J., Zhao, J.: Formalisation of the level of detail in 3D city modelling. Comput. Environ. Urban Syst. **48**, 1–15 (2014)
28. F. Biljecki, H. Ledoux, Stoter, J.: Height references of CityGML LOD1 buildings and their influence on applications. In: Breunig, M., et al. (eds.) Proceedings of the ISPRS 3D GeoInfo 2014 Conference, November 2014
29. OpenJS Foundation: "Node.js main page,". https://nodejs.org/en/. Accessed 22 Jan 2021
30. Yan, J., Zlatanova, S., Aleksandrov, M., Diakite, A.A., Pettit, C.: Integration of 3D objects and terrain for 3D modelling supporting the digital twin. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. **44**, 147–154 (2019)