Development of a QGIS plugin for the CityGML 3D City Database

Konstantinos Pantelios
student #5374367
K.PANTELIOS@student.tudelft.nl

1st supervisor: Camilo León Sánchez
2nd supervisor: Giorgio Agugiaro
external supervisor: Claus Nagel

January, 2022

Contents

List of Figures

Acronyms

3DCityDB 3D City Database. 3, 4

ADE Application Domain Extension. 3, 4

BAG Basisregistratie Adressen en Gebouwen / Dutch Cadastre. 3, 16

CVS Comma Separated Values. 3, 4

ESRI Environmental Systems Research Institute. 3, 5

GIS Geographical Information System. 3, 5

GML Geography Markup Language. 3, 4

GUI Graphical User Interface. 3, 11

JSON JavaScript Object Notation. 3, 4

KLM Keyhole Markup Language. 3, 4

LOD Level Of Detail. 3, 5

LTR Long Term Release. 3, 5

LTS Long Term Support. 3, 11

OGC Open Geospatial Consortium. 3, 4

QGIS Quantum Geographical Information System. 3, 5

RDBMS Relational Database Management System. 3, 4

SFM Simple Feature Model. 3, 13

SQL Structured Query Language. 3, 4

TOC Table Of Contents. 3, 13

UI User Interface. 3, 6

UML Unified Modeling Language. 3, 4

UX User eXperience. 3, 6

XML Extensible Markup Language. 3, 4

1 Introduction

With the advancements in technology and especially computing power and software abundance, 3D city model functionality moved from only data visualisation to well refined data analysis applications. In practice city planners and other relevant actors can use city models to create simulations regarding energy consumption, traffic, growth, disaster management or any other metric that can felicitate urban management and planning (Biljecki et al., 2015).

The issue that often arises with city models is the complexity of the city as an entity. Cities are composed of many different objects which are not guaranteed to be found in every city and at the same type. Additionally, a particular type of a city model that is structured to accommodate the energy sector might not be able to be used in other applications. Lastly, there could be many different data structure formats used by different people making data communication and interoperability difficult (Stadler et al., 2009).

These are some of the motivations that led the OGC towards the adoption of the CityGML. CityGML is a data model that is openly available for use allowing city models to be shared between different people and organisations with ease. Moreover, it solves the interoperability issue as its XML structure can be used universally. Regarding the data model, CityGML tries to accommodate for as much information as possible which means that main classes are hierarchically linked to more detailed features representing the city as best as possible (Gröger and Plümer, 2012). An example of this, that is also the stepping stone of this research, is that buildings can have building parts or other, relative to the building, installations with particular attributes. It quickly becomes apparent that while this model is really comprehensive, its XML based hierarchical structure between the features, attributes and geometries is bound to produce enormous GML files that can be hard to work with. A city center of approximately 3000 buildings, depending on the level and type of detail, can take up to 1 GB of storage space.

This complexity, hints towards the usage and development of different encodings. One such encoding is based on JSON, namely CityJSON which was developed by TU Delft (Ledoux et al., 2019). Another one is based on SQL with the city model data stored exclusively in a geo-spatial database. This approached was followed by a team at TU Berlin who originally developed the open '3D City Database' software (Yao et al., 2018). This research is going to be relevant to the latter approach and solution.

3DCityDB, as the software suite is called, operates with relational geo-spatial databases like PostgreSQL and Oracle and follows the CityGML OGC standard. In practice CityGML files are imported into the database based on a set of mapping rules storing data into a set of predefined tables. Having city models stored in this way gives the benefit of database operations that help in data manipulation. For example the use of spatial indices can speed up spatial queries or with the use of pre-defined functions, entries can be handled efficiently. Moreover, users are able to create their own complex functions and queries to analyse data more effectively. Additionally, 3DCityDB with the use of its 'Importer/Exporter' application provide a high level way of creating queries based on location, detail, feature etc. and allows exporting the database's contents to KLM, XML, GML, JSON, CVS formats. Lastly, it allows the data model to be enhanced with different types of ADEs using specific extension rules (Yao et al., 2018).

The caveat with this approach is again, the complexity. People that need to use this system effectively must have sufficient knowledge of RDBMSs, SQL and programming skills, CityGML and UML comprehension. This slow learning curve means that people may not be able to use the software immediately and effectively without the active help of an expert in the field. Thus, the following question arises. How can this application be introduced to the wide audience of city planners and other users that don't have the required technical knowledge?

Nowadays, people working with geographical data use a specific type application called

GIS. Two of the most popular are the proprietary ArcGIS from ESRI and the open source QGIS. These example software fall into the no-code category. This in practice means that even though they are equipped to handle a plethora of different geographical operations using many techniques and algorithms (ranging form data management and analytics to image and 3D processing), they do not require any programming knowledge or advanced technical skills. This is an important benefit as the offered convenience allow people with no previous experience to enter the field of geo-information fast and effectively. Their popularity caused an increased demand and offer of tutorials and courses and even certifications to be obtained from various institutions. Consequently, people in this field (like city planners) are proficient or accustomed in the use of GIS software (Steiniger and Hunter, 2012).

This research is going to alleviate the aforementioned limitation of 3DCityDB by developing a plugin within the QGIS software. QGIS is mainly selected due to its open source nature, meaning that the plugin, as it is open source it self, is not obstructed by pay walls and could be reached by as many people as possible. In short, QGIS can provide a recognisable user-friendly environment, while the plugin can handle back-end operations of setting up and querying the database.

The plugin focuses mainly on the Building features, but that will be open to further extension for the remaining CityGML modules (bridges, waterbodies, etc.). More specifically, it will be based on CityGML v.2.0, the 3DCityDB for PostgreSQL/PostGIS and QGIS LTR 3.22. Lastly, its functionalities are to allow the user to connect, load, edit features attributes, and update the database.

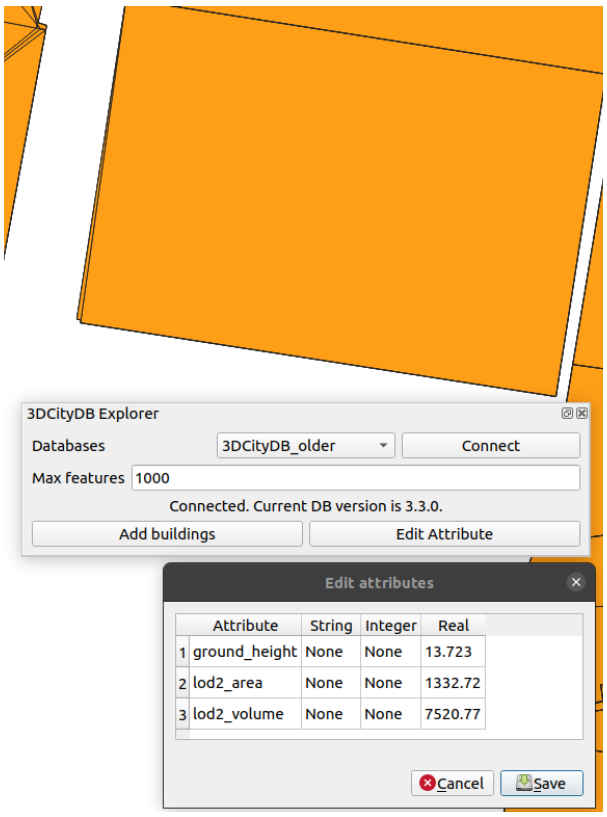
2 Related work

2.1 Existing similar plugins

Currently, two other experimental plugins exist that seemingly try to tackle the question of this research.

2.1.1 3DCityDB Explorer

The first plugin is called '3DCityDB Explorer' and is openly available through GitHub (`https://github.com/3dcitydb/3dcitydb-qgis-explorer`)(Figure 1). According to the authors, it is currently in a development phase with the last update pushed on 8th of March, 2021. In short its functionality is to load data from a 3D City Database and modify the generic attributes of the underlying geometry. The limitations are that it only works for building features represented by LOD2 geometries excluding any composing classes like building parts. Moreover, it is not compatible for user defined schema or multiple schemas, as only the default one (citydb) is hard-coded to work with.

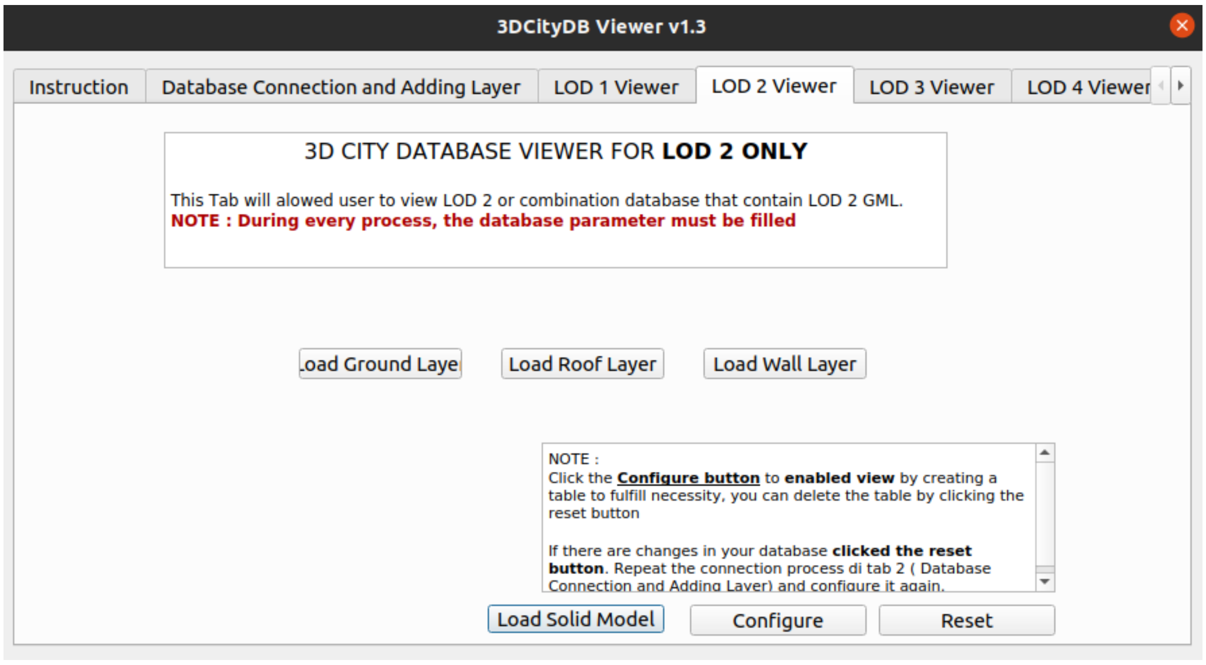


	Attribute	String	Integer	Real
1	ground_height	None	None	13.723
2	lod2_area	None	None	1332.72
3	lod2_volume	None	None	7520.77

Figure 1: 3DCityDB Explorer

2.1.2 3DCityDB Viewer

The second plugin is called '3DCityDB Viewer' and can also be found in GitHub (`https://github.com/aberhamchristomus/3DCityDB-Viewer`)(Figure 2). This plugin lacks any meta-data related to its development phase (no guarantee whether is active or not), yet the last update was pushed on 16th of June, 2021. The functionality for this plugin is to load data from a 3D City Database. Users can select to load building features based on all geometry levels and types excluding any composing classes like building parts. Here, the approach is much more complicated meaning that users may find it difficult to operate it. There is a noticeable lack of UI/UX quality which hinders the overall goal of this research. Additionally, like the previous plugin, it cannot accommodate for multiple schemas or any other than the default citydb schema.



3D CITY DATABASE VIEWER FOR **LOD 2 ONLY**

This Tab will alowed user to view LOD 2 or combination database that contain LOD 2 GML.
NOTE : During every process, the database parameter must be filled

NOTE :
Click the **Configure button** to **enabled view** by creating a table to fulfill necessity, you can delete the table by clicking the reset button

If there are changes in your database **clicked the reset button**. Repeat the connection process di tab 2 (Database Connection and Adding Layer) and configure it again.

Figure 2: 3DCityDB Viewer

In summary, both plugins deviate from the fundamental goad of this research, that is to facilitate operations in 3DCityDB models and increase efficiency and productivity for inexperienced users and experts alike.

2.2 CityGML

The plugin related to this research is based on the 3DCityDB which by itself is based on the CityGML OGC standard. To begin with, only the Building feature is going to be considered in this thesis as is the most important element of a city model. CityGML allows buildings to be represented in multiple LODs i.e. from LOD0 to LOD4. Moreover, in addition to the building's attributes (inherited from the _AbstractBuilding class), it can contain more generic attributes (from Generics module). These attributes and geometries are important as they consist of those element that the plugin is going to be able to operate upon. Another important aspect concerning the representation of the feature is that LOD2 can be also constructed by aggregates of thematic surfaces. Such surfaces often are the ground, walls and roof that together compose the shell of a building. All of the aforementioned characteristics are also applied to the aggregate building classes of 'BuildingPart' and 'BuildingInstallation' which are important as they too are included in this research and are described in figure 3, (Gröger and Plümer, 2012).

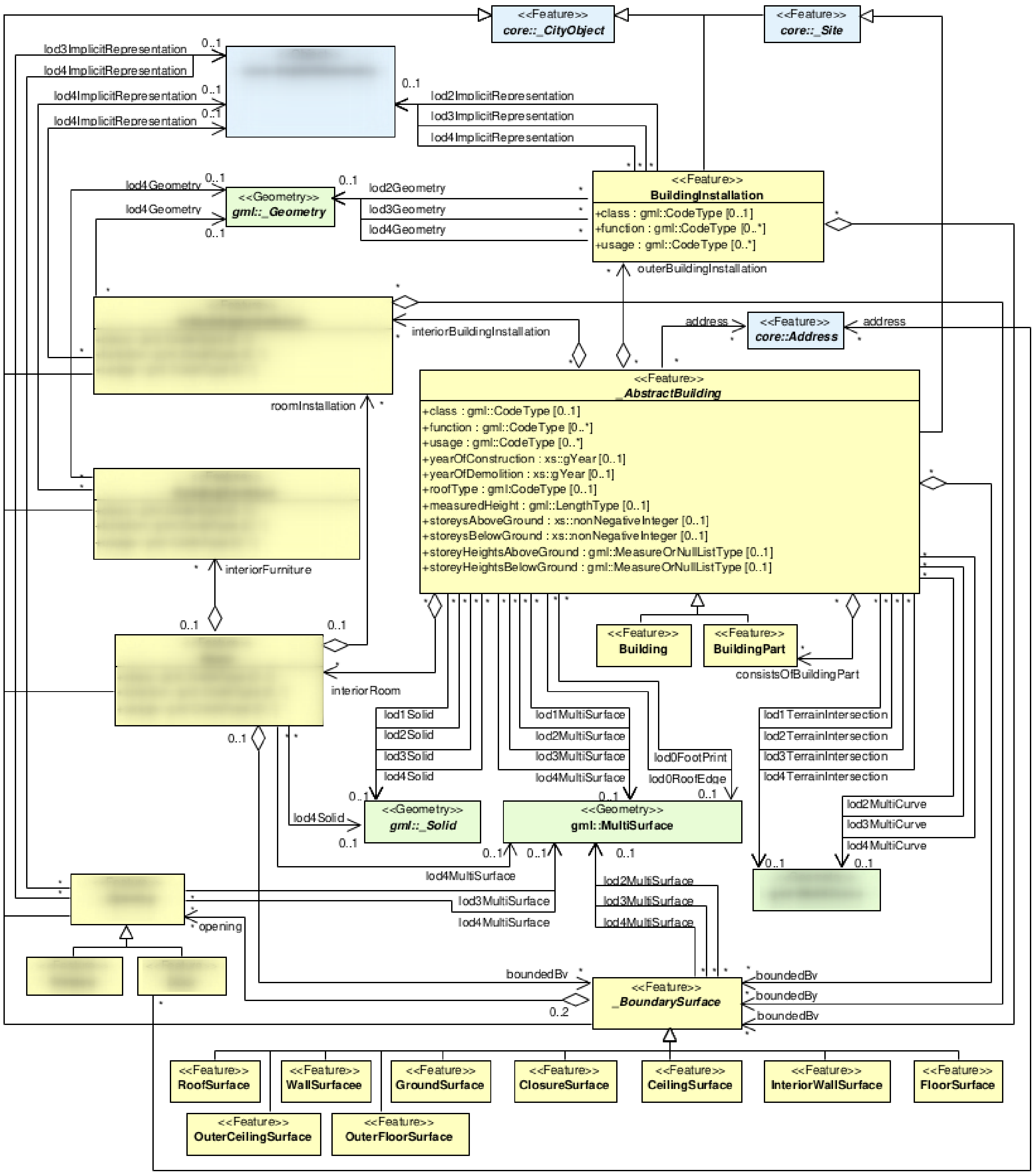


Figure 3: CityGML UML diagram of Building feature and relevant elements (Figure from Gröger and Plümer (2012))

2.3 3DCityDB

CityGML is the standard upon which the 3DCityDB is built. However, in order to convert the conceptual model into relational tables some restrictions are set. In order to keep good server performance the hierarchical classes are mapped directly, in a way that can be joined only once (Figure 4). This translate to the limitation that "*the super class shall be an abstract class that holds all attributes and associations which will be inherited by the concrete sub- classes and every of the sub- classes shall not have any further attributes or associated with other classes*" (Yao et al., 2018). So to accomplice this, many tables are introduced to the schema (66 for v.4.3) and followed by their corresponding primary and foreign keys to set the connection between them. The building feature which is relevant for this research is represented in figure 5.

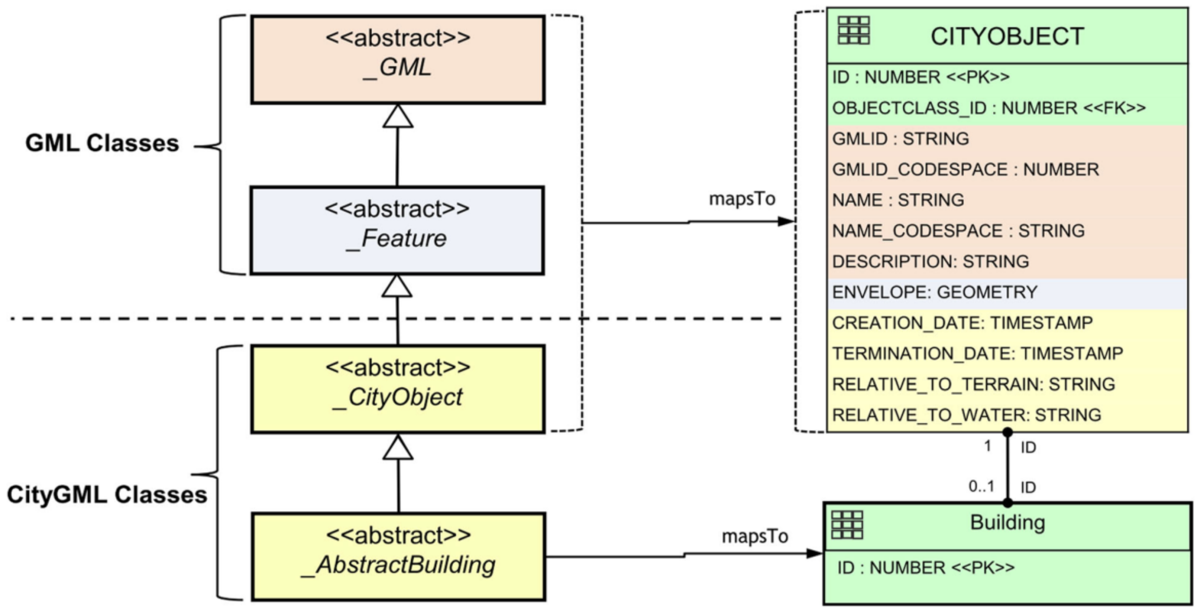


Figure 4: 3DCityDB Inheritance of building feature mapping. (Figure from Yao et al. (2018))

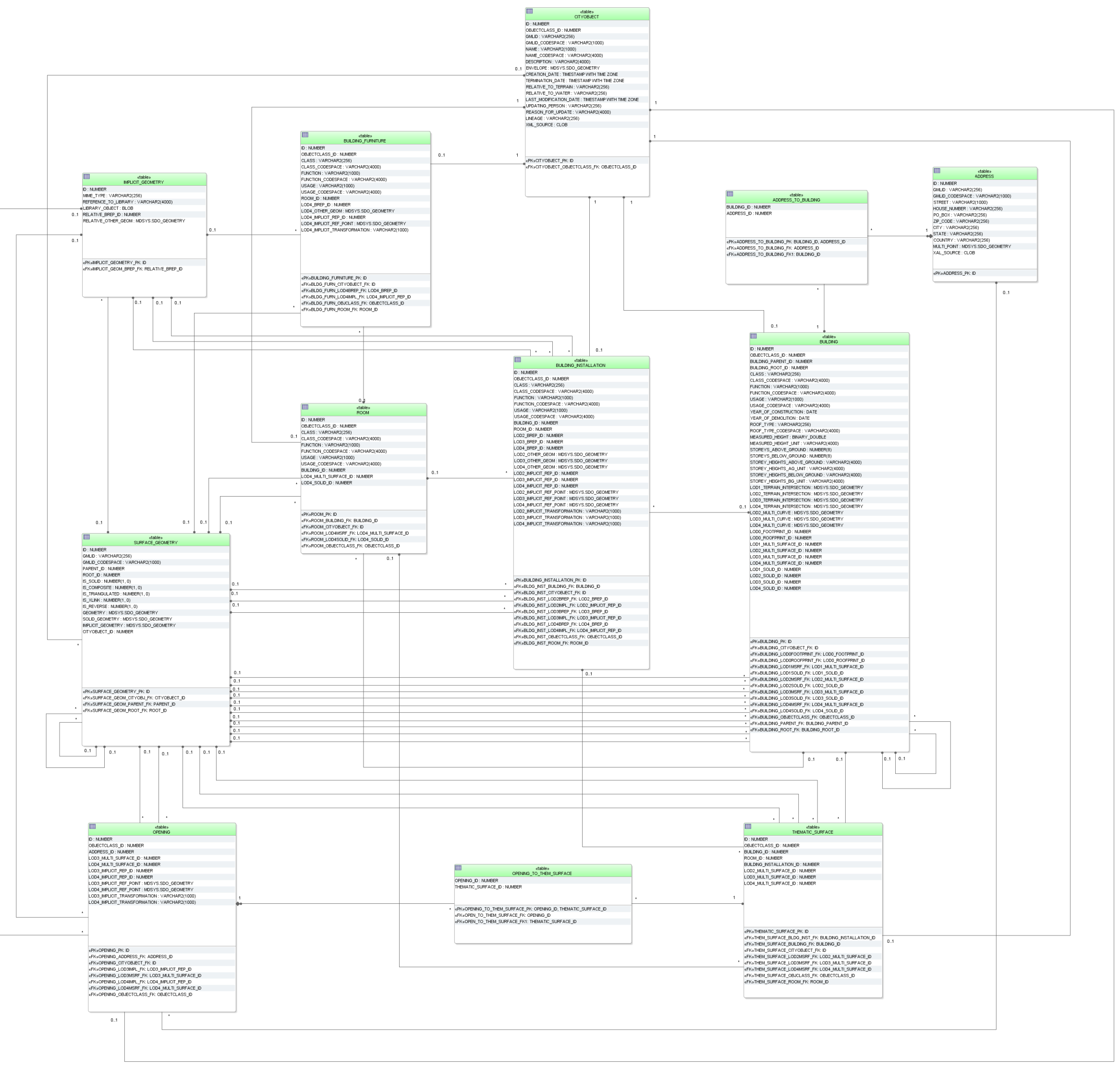


Figure 5: 3DCityDB Building database schema. (Figure from Yao et al. (2018))

3 Research questions

As already mentioned this research aims to use the popularity and accessibility of GIS in order to introduce and facilitate operations of database stored 3D city models. To achieve this goal, the following research questions need to be answered.

- How can GIS software (like QGIS) be used to achieve proper synergy with 3D city models stored in geo-spatial databases?
- How 3DCityDB (database) must be structured, if at all, to accommodate for QGIS operations?
- What QGIS capabilities are considered both user-friendly and practical enough to be appreciated by both inexperienced and expert users?
- How to balance between the complexity of CityGML's database model and best UI/UX?
- What new methods, if any, must be introduced to keep and transfer the benefits of database stored data into the QGIS environment?

The scope of the research is set around the thematic feature of 'Building' including sub-elements like 'BuildingPart', 'BuildingInstallation', 'Address', Appearance' and their attributes, detail level and types of geometries. As buildings are the most frequent and important objects of a city, being able to work with them adds value to the plugin even in its current experimental form. If this endeavour proves successful then it is going to scale easily to include the rest of the CityGML features and even migrate to newer versions. For this project the development is going to be bound on QGIS 3.22 LTR, CityGML 2.0 and 3DCityDB 3.3 (and 4.x). These versions were chosen based on their stability and time in circulation (widely adopted). Lastly, the operating system upon which all of the above tools are installed and the plugin is developed is Ubuntu 20.04.3 LTS. Nevertheless, the plugin is set to be platform-independent.

4 Methodology

To answer the above questions, in this research a QGIS python plugin is going to be developed for 3DCityDB named '3DCityDB Loader'.

4.1 Structure

The GUI is going to be of tab form where each tab corresponds to each step of the pipeline in figure 6. In addition, an 'About' tab is going to include metadata information about the plugin and a 'Settings' tab is going to include any operations relating to the technical build of the plugin (Figure 9). Its GUI is going be a dialogue window instead of a panel widget. The decision for this is made based on practicality, as the plugin itself works like a form that the user fills in order to query specific data and create an equivalent QGIS layer from them. After the layer loads into the QGIS project, all plugin operations are complete, so there is no reason for the plugin to occupy any more screen space. Its position in the QGIS environment is going to be set automatically both under under the 'Database' menu and the 'Database' toolbar which are reserved for database related plugins and processes.

4.2 Functionality

In short the plugin is going to let the user to connect into a database and based on his/her criteria a QGIS layer is going to be created from a corresponding database view. The layer is going to be available for both viewing and editing (Figure 6).

4.2.1 Connection

In more detail, regarding the database connection, the plugin will immediately be able to identify all connections that the user has already stored in his/her local QGIS environment. These connections are going to be presented to the user to choose from. For the case that there aren't any connections or the user wants to create a new one, the plugin is going to have the capability to create it based on the user's database credentials. Moreover, users can opt to store these credentials into their own local environment (Figure 9a).

After selecting a connection the user can connect to its underlying database. At this step, in the background, some test are going to be conducted to assess a valid and successful connection. Additionally, more test are going to be conducted to assess whether a particular database has the necessary 3DCityDB structure. After this step, the database is going to be enhanced by installing necessary for the methodology tools. However, the user is going to be queried before installation and be able to clear his/her database from all added elements. These elements are most likely going to refer to updatable views with their corresponding trigger functions and rules to which the QGIS layers are connected. Lastly, to aid the user visually, these steps are going to be followed by status information like currently connected to database, database version and server version.

4.2.2 Import

Following successfully the connection steps, user will be redirected into the Import tab where he is called to set the parameters that will define the imported layer. At first, users are able to select between multiple schemas. In the background, the plugin will fetch all available schemas and display them to the user. However, there are going to be event handlers that will assess the validity of the schema on-the-fly. In the next step users select between thematic features. As mentioned in the scope of this research, only buildings are going to be included, however the functionality to choose from a list, is going to be implemented to facilitate future development. After that, users will be asked to choose the geographical extents from which features are going to be imported. This decision was made to avoid performance and stability issues in cases of extremely large data-sets. However, to avoid unnecessary user freedom restrictions, it acts solely as a warning. Next, the plugin will give geometry level and types options. In particular, firstly, users will be able to choose between LOD0, LOD1 and LOD2 and secondly, they will be able to choose between 'Footprint'/'Roofprint', 'Multi-surface'/'Solid', 'Multi-surface'/'Solid'/'Thematic-surface' types respectively. Lastly, based on the above parameters, a corresponding view from the database is going to be loaded in the current QGIS project. (Figure 9b)

4.2.3 User Experience from User Interface

It is important to note that to enhance the UX the plugin is going to be developed in a way to guide the user without the need to follow any extensive documentation or frequent tool-tips.

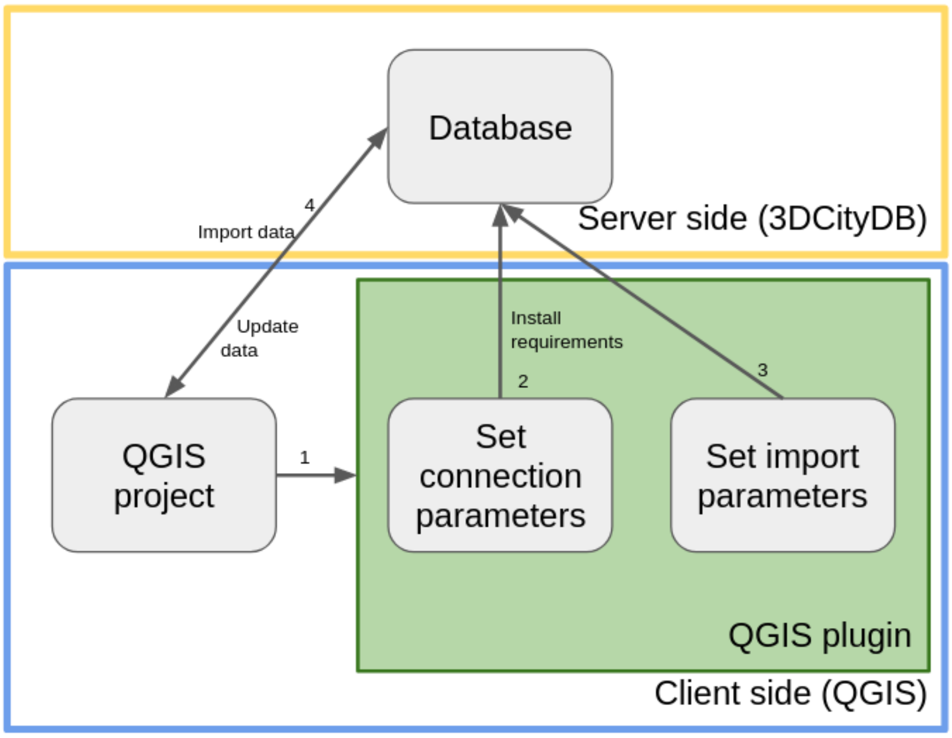


Figure 6: 3DCityDB Loader pipeline

To do this, the plugin will adhere to the order illustrated in figure 6. In practice this means that tabs and buttons are not going to be available to the users until necessary criteria are met. This criteria is as simple as handling the slot before the unavailable one. For example if there are no buildings inside the selected extent, then the users will not be able to select any geometry level/type or import any layer as the layer is going to be empty. The same applies for all the background checks about the 3DCityDB structure. If the connection tab is not properly handled by the user, then the import tab is going to be unavailable. All in all, these restriction are going to act as a convenient guide that users can follow intuitively.

4.2.4 Database - 3DCityDB

For 3DCityDB features to be properly imported as QGIS layers, the database needs to be enhanced with additional contents. The complexity issue of 3DCityDB and CityGML, as mentioned before, is that the model has many different tables and that attributes of distinct features are stored over multiple tables. The plugin will be responsible to convert features whose data span over multiple table into one concise table containing everything relevant. To do this, the plugin is going to handle the installation of a set of views that are going to correspond to every parameter combination that the user could make. In the end these views are going to contain all feature attributes and geometries including, children classes with their own attributes and geometries. In addition, the implementation in going to allow these views to be updatable, meaning that changes happening on the view (QGIS layer) are going to cascade to the underlying tables inside the database. These views are inspired from the 3D geoinfromation's group '3DCityDB plus'. Here, it is important to note that for this experimental plugin is, only the attributes are going to be updateable. In addition to updating, the views are going to allow entries to be deleted as well.

4.2.5 QGIS

In QGIS, at the moment, the proposal is to name the imported layers based on the syntax: schema_feature_lod_geometryType. Each layer corresponds to a view, and simplifies the complex CityGML model to a set of layers that are compatible with the SFM. Additionally, a specific TOC structure (Figure 7) is going to be created in order to insert each layer to a corresponding location.

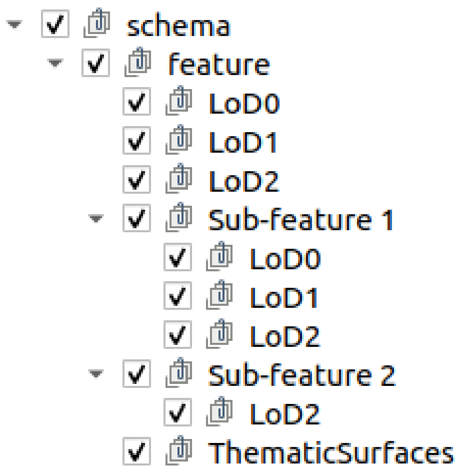
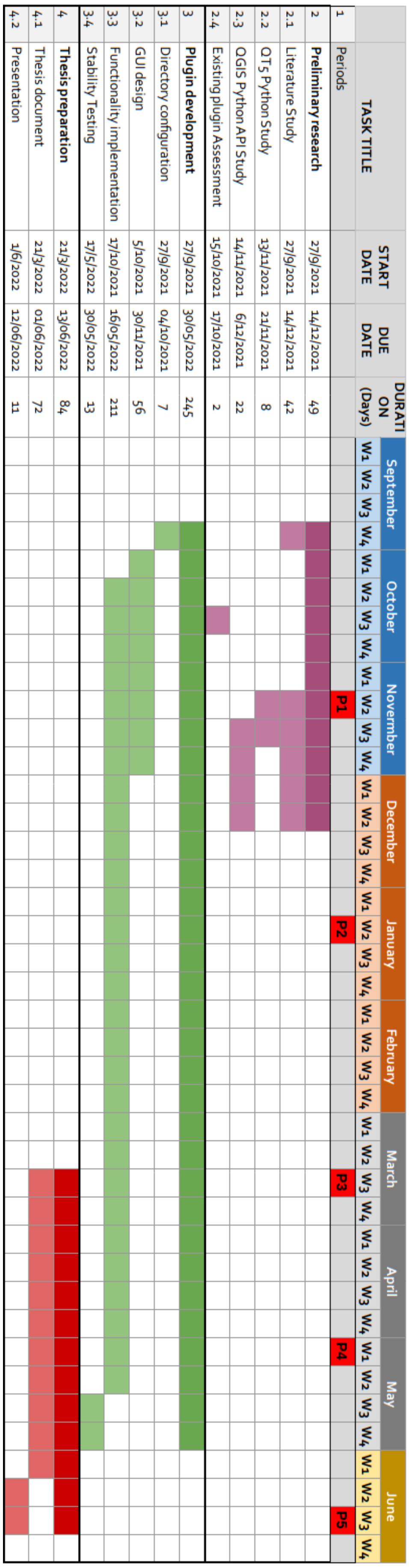


Figure 7: Proposed hierarchical structure of the QGIS table of contents for the layers from the 3DCityDB

5 Time planning

The schedule for the tasks related to this thesis is presented with a Gantt diagram in figure 8.



	TASK TITLE	START DATE	DUE DATE	DURATION (Days)
1	Periods			
2	**Preliminary research**	27/9/2021	14/12/2021	49
2.1	Literature Study	27/9/2021	14/12/2021	42
2.2	QT5 Python Study	13/11/2021	21/11/2021	8
2.3	QGIS Python API Study	14/11/2021	6/12/2021	22
2.4	Existing plugin Assessment	15/10/2021	17/10/2021	2
3	**Plugin development**	27/9/2021	30/05/2022	245
3.1	Directory configuration	27/9/2021	04/10/2021	7
3.2	GUI design	5/10/2021	30/11/2021	56
3.3	Functionality implementation	17/10/2021	16/05/2022	211
3.4	Stability Testing	17/5/2022	30/05/2022	13
4	**Thesis preparation**	21/3/2022	13/06/2022	84
4.1	Thesis document	21/3/2022	01/06/2022	72
4.2	Presentation	1/6/2022	12/06/2022	11

Figure 8: Gannt diagram of thesis

6 Tools and datasets used

6.1 Plugin

This plugin is going to be developed for the QGIS environment and in particular for the LTR version of 3.22. Moreover, it will be developed with the Python embedded programming language of version 3.8.10 within QGIS. The database server is going to be PostgreSQL of version 12.9 on which the 3DCityDB versions 3.3 and 4.1 will be installed. The GUI of the plugin is going to be constructed using the QT toolkit of version 5.12.8 that QGIS is based upon.

6.2 Test Data

Regarding the datasets that are going to be used, a complete collection of buildings in the wider area of Rijssen-Holten (derived from 3D BAG) is provided as a 3D city model. This comes in gml format and imported into both local and remote 3D City Databases using its Imported/Exporter software (v.4.3) A subset of this dataset or other is going to be provided from a public GitHub repository with the rest of the plugin code. Lastly, other random open datasets are going to be used for test purposes. The datasets that are going to be used for test must follow the criteria below:

- Contain at least one building feature
- The building feature geometry is at least at one of the possible LODs (0 to 2)

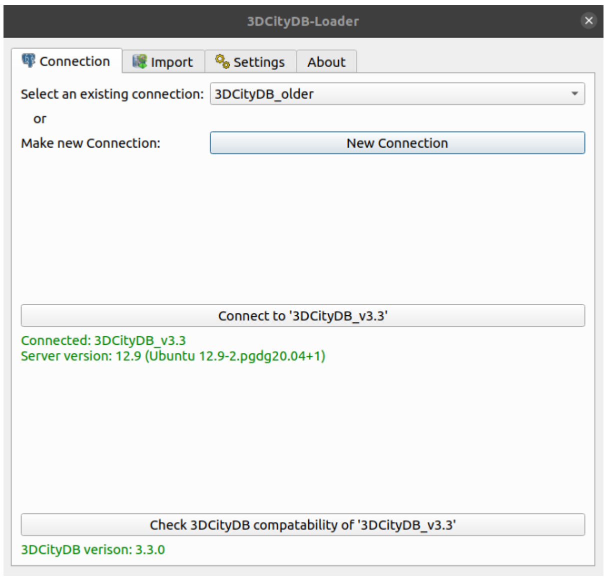
7 Preliminary results

Currently, the plugin's development is at a point where users can do the following:

- Select from existing connections
- Create new connection
- Opt to store credentials of new connections
- Connect to database of specific connection
- Enhance database by installing requirements
- Uninstall said requirements
- Select from existing schemas
- Select from existing features (only buidlings)
- Select bounding box either from a layer, from current map extents or from drawing a square
- Select from existing LODs
- Select from existing geometry types
- Import data layer (currently only for viewing)

Correspondingly, the plugin handles the following operations in the background:

- Reads the user's global settings to find existing connections
- Connects to database using existing parameters
- Can store new parameters into the user's global settings
- Executes installation script to enhance the database
- Executes un-installation script to clear the database
- Checks for valid/successful connection
- Checks for valid 3DCityDB structure
- Reads existing schemas and guards against invalid ones
- Reads existing features (only buildings)
- Reads selected bounding box which is used to import specific entries based on spatial filter
- Reads existing LODs
- Reads existing geometry types
- Creates QGIS layer in the project and connects it with a corresponding view

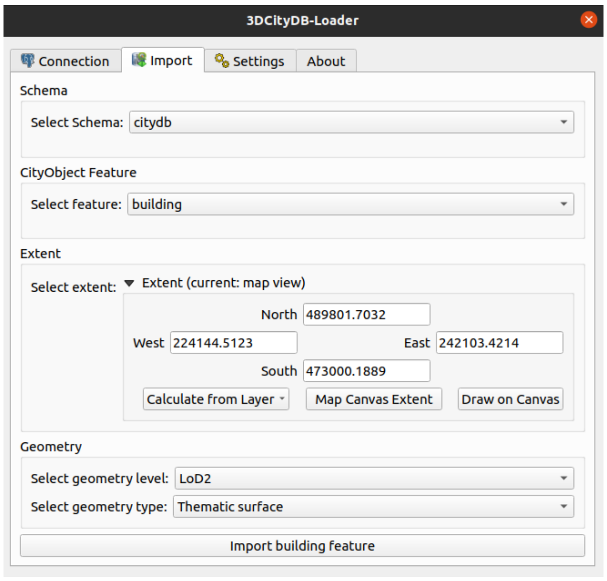

3DCityDB-Loader
Connection
Import
Settings
About
New Connection
Connect to '3DCityDB_v3.3'
Check 3DCityDB compatability of '3DCityDB_v3.3'

Select an existing connection: 3DCityDB_older
or
Make new Connection:

Connected: 3DCityDB_v3.3
Server version: 12.9 (Ubuntu 12.9-2.pgdg20.04+1)

3DCityDB verison: 3.3.0

(a) Connection Tab



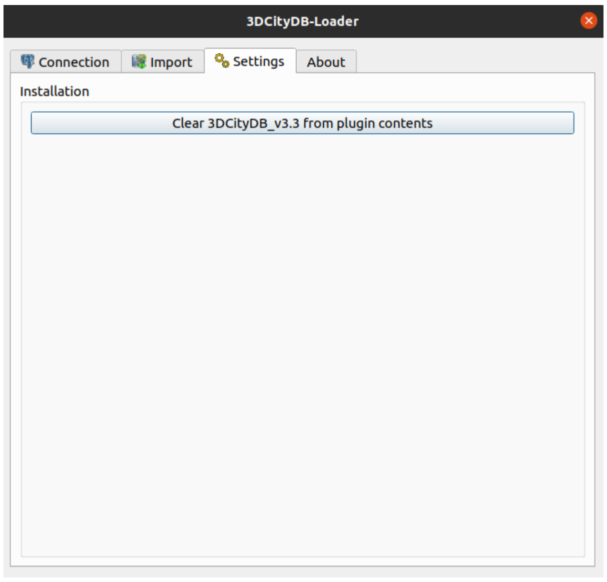
Schema
Select Schema: citydb

CityObject Feature
Select feature: building

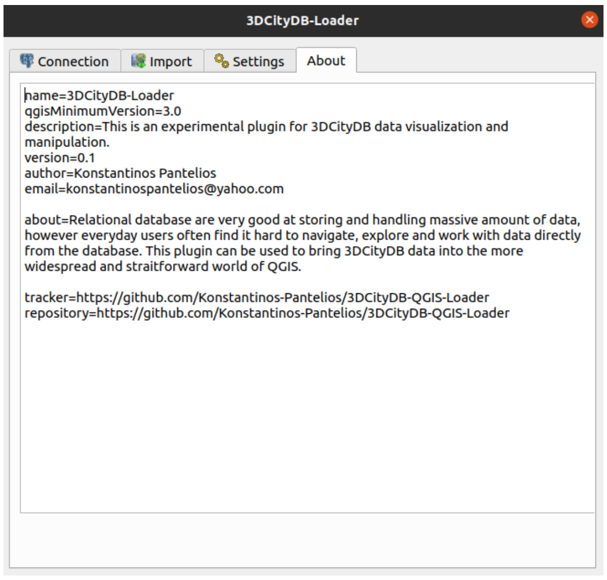
Extent
Select extent: Extent (current: map view)
North 489801.7032
West 224144.5123
East 242103.4214
South 473000.1889

Geometry
Select geometry level: LoD2
Select geometry type: Thematic surface

(b) Import tab



(c) Settings tab



```
name=3DCityDB-Loader
qgisMinimumVersion=3.0
description=This is an experimental plugin for 3DCityDB data visualization and
manipulation.
version=0.1
author=Konstantinos Pantelios
email=konstantinospantelios@yahoo.com

about=Relational database are very good at storing and handling massive amount of data,
however everyday users often find it hard to navigate, explore and work with data directly
from the database. This plugin can be used to bring 3DCityDB data into the more
widespread and straitforward world of QGIS.

tracker=https://github.com/Konstantinos-Pantelios/3DCityDB-QGIS-Loader
repository=https://github.com/Konstantinos-Pantelios/3DCityDB-QGIS-Loader
```

(d) About tab

Figure 9: 3DCityDB Loader plugin (as of 05/01/2021)

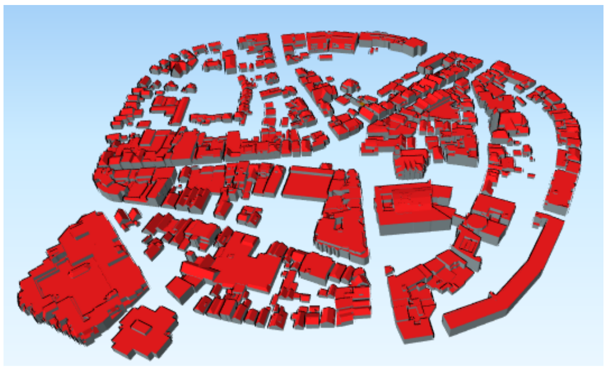


Figure 10: 3D City Model of the city-center of Rijssen. Imported from 3DCityDB into QGIS using 3DCityDB Loader. LoD2 thematic surface representation.

References

F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015. ISSN 2220-9964. doi: 10.3390/ijgi4042842. URL `https://www.mdpi.com/2220-9964/4/4/2842`.

G. Gröger and L. Plümer. Citygml–interoperable semantic 3d city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, 2012.

G. Gröger, T. H. Kolbe, C. Nagel, and K.-H. Häfele. *OGC City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium, 2.0.0 edition, 2012.

J. Herring et al. Opengis® implementation standard for geographic information-simple feature access-part 1: Common architecture [corrigendum]. 2011.

T. H. Kolbe. *Representing and Exchanging 3D City Models with CityGML*, pages 15–31. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-87395-2. doi: 10.1007/978-3-540-87395-2_2. URL `https://doi.org/10.1007/978-3-540-87395-2_2`.

H. Ledoux, K. A. Ohori, K. Kumar, B. Dukai, A. Labetski, and S. Vitalis. Cityjson: A compact and easy-to-use encoding of the citygml data model. *Open Geospatial Data, Software and Standards*, 4(1):1–12, 2019.

J. Malczewski. Gis-based land-use suitability analysis: a critical overview. *Progress in Planning*, 62(1):3–65, 2004. ISSN 0305-9006. doi: https://doi.org/10.1016/j.progress.2003.09.002. URL `https://www.sciencedirect.com/science/article/pii/S0305900603000801`.

A. Stadler, C. Nagel, G. König, and T. H. Kolbe. *Making Interoperability Persistent: A 3D Geo Database Based on CityGML*, pages 175–192. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-87395-2. doi: 10.1007/978-3-540-87395-2_11. URL `https://doi.org/10.1007/978-3-540-87395-2_11`.

S. Steiniger and A. J. S. Hunter. *Free and Open Source GIS Software for Building a Spatial Data Infrastructure*, pages 247–261. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-10595-1. doi: 10.1007/978-3-642-10595-1_15. URL `https://doi.org/10.1007/978-3-642-10595-1_15`.

S. Vitalis, K. Arroyo Ohori, and J. Stoter. Cityjson in qgis: Development of an open-source plugin. *Transactions in GIS*, 24(5):1147–1164, 2020. doi: https://doi.org/10.1111/tgis.12657. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12657`.

Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi, and T. H. Kolbe. 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):1–26, 2018.