LoD2BIM: A New Workflow for reconstructing and converting LoD2 Model to Information-rich IFC Model for Existing Building

Zhongxin Xia, Uwe Rüppel

Institute of Numerical Methods and Informatics in Civil Engineering, Technical University of Darmstadt, Germany – (xia, rueppel)@iib.tu-darmstadt.de

Keywords: CityGML, IFC, BaseX, Blender, Blender Add-on, Reconstruction

Abstract

Numerous existing buildings in Germany lack corresponding BIM models. Advanced analysis and visualization purposes, including indoor navigation, energy demand simulations, and building materials recycling on these buildings presents significant challenges, as only Level of Detail 2 (LoD2) models provided by local housing authorities, along with architectural drawings and textual descriptions of the internal structures, are available. This paper proposes a semi-manual workflow, LoD2BIM, to facilitate the efficient reconstruction and conversion of existing building models, particularly transforming Germany's official 3D Building Models in LoD2 into comprehensive, information-rich IFC models using open-source software. Through the integration of BaseX and CesiumJS, the visualization, querying and extracting of one LoD2 model from CityGML file become easy and feasible. Central to the LoD2BIM workflow is Blender, it acts as the primary platform for model editing, reconstruction and conversion. Users can expediently rebuild and refine LoD2 models utilizing Blender's visual interface and the tool floorplan to blender3d, based on floorplans or detailed written descriptions. New Blender add-ons are developed to streamline the model import and conversion process. The BlenderBIM add-on is also modified to facilitate the automatic batch assignment of mesh as IFC classes. This workflow not only showcases the reconstruction and subsequent conversion of CityGML models into IFC models but also illustrates the potential to act as an alternative to commercial software in Building Information Modeling (BIM) editing.

1. Introduction

Advanced analysis and visualization purposes, including energy demand simulations, and building materials recycling are crucial for the sustainable development of cities. However, the majority of existing buildings is not maintained, refurbished or deconstructed with BIM yet. BIM-based Facility Management is currently limited to less than 1% of all buildings internationally (Vorbeck and Wills, 2022). These existing building are typically documented through architectural floorplans, textual descriptions, and LoD2 model in the CityGML file of entire city. According to the version 2.0 of the CityGML standard, LoD2 just further refines LoD1 model by incorporating simplified roof structures. Many publicly accessible CityGML models predominantly focus on LoD1 and LoD2. These levels generally suffice for basic urban planning and analytical tasks. Some examples of open-source CityGML models in LoD1 and LoD2 are available for download from the websites of different German states, such as Niedersachsen and Sachsen (LGLN, 2024, GeoSN, 2024). Fig.1 shows the LoD2 models of the city Idstein in Germany, the models are freely accessible by Hessian Administration for Land Management and Geoinformation (2024).

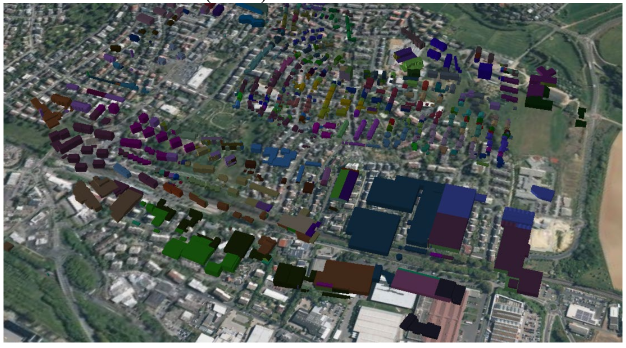


Figure 1. CityGML of the City Idstein in Germany

The CityGML file for a bigger city or region can have from tens over hundreds of gigabytes in size. If analysis of an individual building is required, applications and users need efficient tools to query, visualize, and extract building model from CityGML file.

IFC and CityGML stand as distinct standards for 3D semantic modeling of buildings and urban landscapes, each with its unique focus, scope, and detail level. IFC is highly regarded within the building industry for its comprehensive semantic framework. It facilitates detailed 3D representations of buildings using constructive elements and supports the entire building lifecycle from design through construction to maintenance. Primarily employed in the architecture, engineering, and construction (AEC) sector, IFC caters to the nuanced requirements of professionals engaged in building design, construction, and management. To conduct in-depth analyses of urban structures, CityGML models often require reconstruction and conversion into IFC format, which enriches them with detailed constructive elements and additional information. Figure 2 illustrates how the information-rich IFC model offers a more detailed perspective on existing buildings, underscoring the model's enhanced informational content. This implies that users can extract intricate details about a building's construction and components from the IFC model for specific applications. And the LoD2 model can act as a precursor for the IFC model's reconstruction.

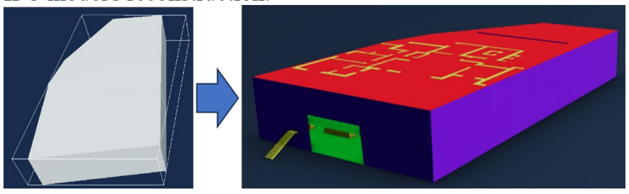


Figure 2. Reconstructing LoD2 model with more structures

While *Autodesk Revit* is excellent for creating detailed architectural and structural models, it is not the ideal tool for

directly reconstructing or refining CityGML models. Integrating CityGML data into *Revit* needs conversion into a compatible format, possibly via third-party plugins or converters. Despite comprehensive features and extensibility, its high cost may deter individual users, small firms, or startups. Several commercial software support CityGML, such as *CityEditor*, *Bentley Map*, and *SketchUp Pro* (Citygml Wiki, 2020). However, these solutions are not only costly but also proprietary, limiting further development.
In contrast, this paper introduces a novel workflow enabling users to create information-rich IFC model for their individual existing building model based on floorplans and CityGML file. The focus of this paper is not on the conversion of model formats, but rather on the reconstruction of the model's structure. This approach, grounded in an open-source ecosystem, ensures all required software and frameworks are freely available online, offering a cost-effective and flexible solution for detailed 3D modeling.

2. Related work

This chapter provides an overview of the status and developments concerning the extraction, reconstruction, and conversion of LoD2 model.

2.1 CityGML management through 3DCityDatabase

The *3DCityDatabase* provides a relational database schema designed for storing CityGML data in Oracle or PostgreSQL. It contains an Importer-Exporter tool for loading CityGML data into the database and exporting it back to the CityGML format. The process involves an object-relational mapping to transfer CityGML features into a relational database (Yao et al., 2018). However, Koch and Löwner (2016) found that *BaseX* outperforms the *3DCityDatabase* regarding the import of data and export of databases and features well as the database size for different input dataset. The primary factor behind this is the time-consuming process of converting data from XML to the relational schema within the *3DCityDatabase*, which is not required in BaseX. The more complex, hierarchical and irregular the XML data are, the more difficult is the mapping to a relational schema and the storage in an XML-enabled database (Pavlovic-Lažetic, 2006, Harms, 2008, Zhang, et al., 2008). Due to the object-relational impedance mismatch, this mapping process is also often connected with information loss (Nagel and Stadler, 2008, Zhang, et al., 2008, Kudrass and Conrad, 2002). The object-relational mapping and its drawbacks can be avoided by using native XML database systems to manage complex XML data.
And although the *3DCityDatabase* has a web-based front-end called *3DCityDB-Web-Map-Client* for 3D visualization of city models based on *CesiumJS* viewer, it only supports the display, dynamic loading, and unloading of large 3D models in the form of tiled KML/glTF datasets exported from the *3DCityDatabase* (Yao et al., 2018).

2.2 Model Reconstruction und Conversion

OpenStreetMap (OSM) and CityGML are both used for representing geospatial information, particularly related to urban environments. The 3D building models in OSM are created by the community of mappers, and the level of detail can vary based on individual contributions. In general, the building models in OSM and CityGML LoD2 may have some similarities when representing urban structures. Both OSM and CityGML LoD2 can include basic geometric structures, representing the outlines and shapes of buildings. Götz and Zipf (2012) presented a framework for creating 3D building models from OSM, while Götz (2013) focused on the generation of highly detailed CityGML LoD4 models with interior structures. Wang et al. (2017) are inspired by the work of (Goetz, 2013), they presented an algorithm which is developed for generating the 3D models of buildings, based on the data schema IndoorOSM. They focused on constructing the inner parts of buildings, such as rooms, windows, and corridors (Fig.3).

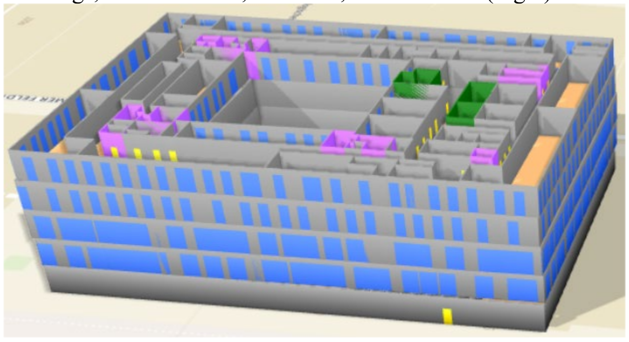


Figure 3. Generating the 3D models of buildings based on IndoorOSM (Wang et al., 2017).

IndoorOSM is defined based on the definition of a 3D Building Ontology, following a hierarchically organized structure. However, IndoorOSM is now defunct and has been replaced by other proposals that do not have the technical problems (tag collisions, massive use of relations and direct use of osm element IDs) it had (OSM Wiki, 2023).
Biljecki et al. (2016) developed *RANDOM3DCITY*, an experimental procedural modelling engine for generating synthetic datasets of buildings and other urban features. The engine is designed to produce models in CityGML and does so in multiple LoDs. As a result of their permutations, each building can be generated in 392 different CityGML representations, an unprecedented number of modelling variants of the same feature. The datasets produced by *RANDOM3DCITY* are suited for several applications. However, the new structure is randomly generated by *RANDOM3DCITY* and does not match the appearance of existing buildings in reality.
Hakim et al. (2024) developed CityJSON Importer Plugin for *Autodesk Revit*. However, when CityJSON models are imported into *Revit*, they are in mesh format, where the surfaces of the model lack depth. This lack of depth complicates any modifications or enhancements within *Revit*, such as directly inserting doors and windows into walls. Additionally, the cost of using *Revit* is prohibitively high. Salheb et al. (2020) developed and implemented a methodology to convert the CityGML model into IFC model in Python 3. The algorithm does not incorporate the capability for structural reconstruction.
Machine learning enables computers to learn from data and perform tasks that are difficult to program explicitly. Hensel et al. (2019) described a workflow for generating LoD3 CityGML models based on textured LoD2 CityGML models by adding window and door objects. This research focused on facades reconstruction using object detection and doing additional tests using a modified Faster R-CNN which is capable of predicting depth information for more detailed facade modeling. Research on converting 2D raster drawings into 3D vector data has a long history. It is a process of transforming a flat image into a 3D model. This can be useful for applications such as architecture, design, and virtual reality. As for the rapid reconstruction of internal structures, there have been studies that recognize 2D floorplan using deep learning based on semantic segmentation.

Vidanapathirana et al. (2017) presented Plan2Scene, a system that converts floorplan and set of sparse photos to a textured 3D mesh. This work focused on the texture generation stages, defining a suite of texture quality metrics, and proposing a texture synthesis approach with GNN-based propagation to unobserved surfaces. Park et al. (2021) proposed 3DPlanNet Ensemble methods that combine data-based models learned with 30 training data and rule-based heuristic methods. Through this study, the remarkable achievement of restoring the wall with an accuracy of 95% or more, and a drawing without dimensions was created with a size accuracy of 97% or more.
Machine learning algorithms can interpret and reconstruct 2D drawings into 3D structures, but they might not always capture the intricacies or the specific architectural intentions behind the original designs. Details such as textures, materials, and finer architectural elements might not be fully represented in the automatically generated 3D model. The automatically generated 3D models might contain errors or inaccuracies that need correction. Architects and 3D modelers often need to manually adjust these models, using 3D modeling software, to ensure they meet the required standards and accurately represent the intended design.
This paper will explore a method for internal and external reconstruction of existing LoD2 models using a powerful and versatile open-source 3D computer graphics software *Blender*.

3. Methodology

The Fig.4 shows the schematic diagram. CityGML file of the city is visualized using *CesiumJS* and stored within the *BaseX* database. The models are interconnected via the model's ID within the *Node.js* application. In the 3D geospatial map of the *Node.js* application, user can intuitively view, select, and download any building LoD2 model to local device. With the new created add-on, LoD2 model can be imported into *Blender*. Subsequently, with the aid of modified *BlenderBIM*, some new add-ons and tool *floorplan to blender3d*, the LoD2 model can be reconstructed into a new model and exported in IFC format.

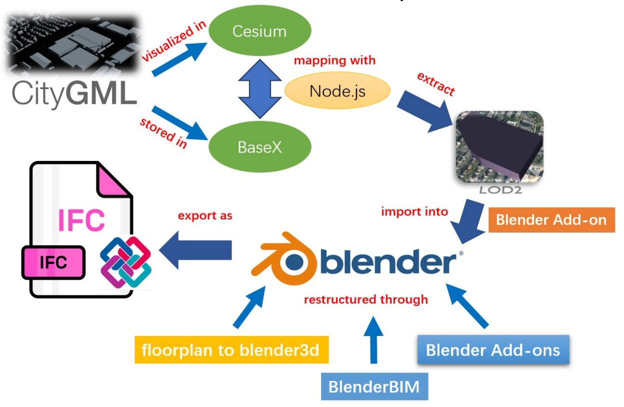


Figure 4. Overview of the proposed concept and its software components

3.1 Linking CesiumJS viewer with BaseX databse

This paper will develop a web-based interface and integrate it with *BaseX* to achieve fast retrieval and extraction of CityGML models. The software *Feature Manipulation Engine* and the cloud-based geospatial platform *Cesium ion* can easily convert CityGML files into 3D tiles format supported by *CesiumJS* viewer, enabling visualization in web (Fig.5). In Fig.5, the map situated beneath the CityGML model represents the city of Idstein, and the map is freely provided by *Cesium ion*. To facilitate the inspection of the model's underside and ensure alignment with the actual building, the model's elevation has been increased.

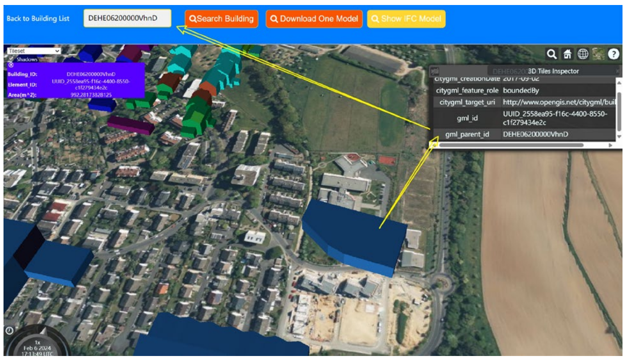
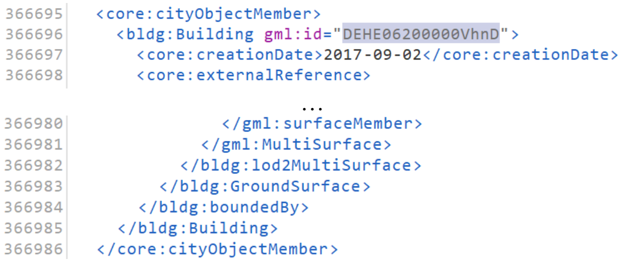

Back to Building List
DEHE06200000VhnD
Search Building
Download One Model
Show IFC Model
Tileset
Shadows
Building_ID: DEHE06200000VhnD
Element_ID: UUID_2558ea95-f16c-4400-8550-c1f279434e2c
Area(m^2): 992.28173828125
3D Tiles Inspector
citygml_creationDate 2017-09-02
citygml_feature_role boundedBy
citygml_target_uri http://www.opengis.net/citygml/bui
gml_id UUID_2558ea95-f16c-4400-8550-c1f279434e2c
gml_parent_id DEHE06200000VhnD

Figure 5. Visualizing CityGML in local *CesiumJS* viewer

The *CesiumJS* version used in this paper is 1.98. When the user clicks on the LoD2 model, the *InfoBox* of *CesiumJS* will display the details of that model, including the attribute *gml_parent_id* that is the ID of this building (Fig.5). The XML file in the *BaseX* database is mapped to the LoD2 model in *CesiumJS* using this *gml_parent_id*.
BaseX is a robust, high-performance XML database engine and a highly compliant XQuery 3.1 processor (BaseX Development Team, 2024). It offers various functionalities to handle XML data, including querying, manipulating and exporting XML files. *BaseX* does not support gml file natively. In many cases, converting a GML file to an XML file can be done manually by simply changing the file extension from ".gml" to ".xml". GML (Geography Markup Language) is an application of XML (eXtensible Markup Language), so GML files inherently adhere to XML syntax. Therefore, changing the file extension is often enough to make the GML file readable and usable by software that accepts XML files.
As shown in Figure 6, each building represented in the XML file starts as a module with "core:cityObjectMember," followed by the building's ID, and then ending with "core:cityObjectMember". Based on this structure, it's possible to extract all the content of the corresponding LoD2 model by using the building's ID as a reference through *XQuery*. *XQuery* is a language for finding and extracting elements and attributes from XML documents.



```
<core:cityObjectMember>
  <bldg:Building gml:id="DEHE06200000VhnD">
    <core:creationDate>2017-09-02</core:creationDate>
    <core:externalReference>
                     …
            </gml:surfaceMember>
          </gml:MultiSurface>
        </bldg:lod2MultiSurface>
      </bldg:GroundSurface>
    </bldg:boundedBy>
  </bldg:Building>
</core:cityObjectMember>
```

Figure 6. Fragment of XML file

The *CesiumJS* viewer is deployed in *Node.js*, which is an open-source, server-side JavaScript runtime environment that allows developers to execute JavaScript code on the server rather than just in web browsers. *jQuery* is a popular JavaScript library that simplifies the process of interacting with HTML documents, handling events, and making asynchronous requests (AJAX) to the server. *jQuery* library is used in this *Node.js* application to capture building's ID from input field. When users click the "Download One Model" button (see Fig.5), the building's ID, for example *DEHE06200000VhnD*, will be passed to the URL address *http://127.0.0.1:8089/building*. When the *HTTP GET* method in the file of *XQuery* accesses the constructed URL address, it can get the ID *DEHE06200000VhnD*. Then, the *XQuery* can extract all the corresponding elements and attributes of the building and export an XML file with the extracted content to local device.

3.2 New Blender add-ons and modified BlenderBIM

Blender is a powerful and versatile open-source 3D software. It encompasses a wide range of tools and features for 3D modeling, animation, rendering, compositing, motion tracking, video editing, and more. It's a free-to-use software, making it highly accessible to artists, designers, animators, and hobbyists. *Blender* and *Revit* both support secondary development. *Revit* plugins are typically developed using the *Revit* API, which is based on the .NET framework (Autodesk, 2024). Developers commonly use programming languages such as C# (C-Sharp) or VB.NET to create custom tools, features, and extensions for *Revit*. Compared with *Revit*, *Blender* add-ons are typically developed using Python, which is known for its simplicity, readability, and versatility, making it a popular choice for scripting and automation tasks in various applications.
The *Blender* version used in this work is 4.0. *Blender* doesn't have a direct native feature to import XML files as 3D models because XML is a markup language for storing and transporting data, not specifically for 3D geometry or models. Therefore, a *Blender* add-on is required to extend *Blender*'s functionality beyond its core features. *Blender* add-on *Import_CityGML* can imports geometry from CityGML file (or files) as one mesh/object (per file) to *Blender* (Dealga, 2020). However, this add-on has some disadvantages. Imported object may be located very far from the *Blender World Origin* (0,0,0). User must manually overwrite imported geometry origin point, to be closer to *Blender World Origin* (0,0,0) (Fig.7).

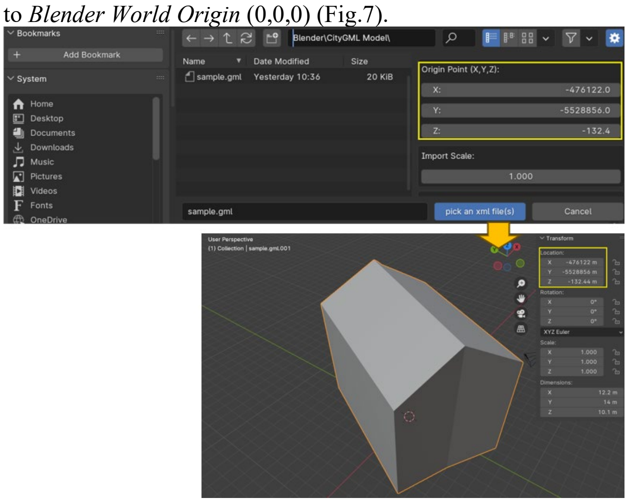

Origin Point (X,Y,Z):
X: -476122.0
Y: -5528856.0
Z: -132.4
Import Scale: 1.000
Location: X -476122 m, Y -5528856 m, Z -132.44 m
Rotation: X 0°, Y 0°, Z 0°
Scale: X 1.000, Y 1.000, Z 1.000
Dimensions: X 12.2 m, Y 14 m, Z 10.1 m

Figure 7. Manually adjusting the geometry origin point

BlenderBIM emphasizes interoperability by facilitating the import and export of IFC files, allowing users to collaborate with other BIM software seamlessly. It offers tools and workflows for creating 3D architectural models, generating construction documentation, and managing building elements' properties such materials, relationships, and metadata. In *Blender*, the imported CityGML model by *Import_CityGML* add-on cannot be visualized within an IFC project created by *BlenderBIM*. The models need to be converted to glb or gltf format, and then import it into *Blender* again (Fig.8). The format gltf works as the medium for reconstruction in *Blender*. Such operation is too complicated.

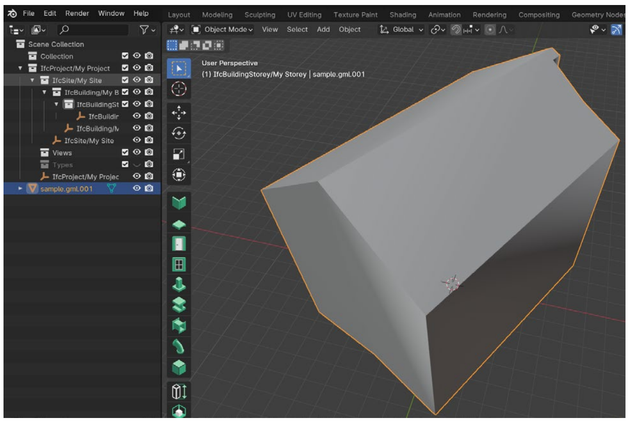


Figure 8. After conversion, the new model in glb format is now available for IfcProject.

In this work, to facilitate the import of CityGML models, a new add-on has been developed. In this new add-on, the *xml.etree.ElementTree* module is used to parse XML file, navigate through the elements of the XML tree, and manipulate the content. This module is commonly used in Python for tasks related to XML processing, such as reading and writing XML files, extracting information from XML documents, and building XML structures (Python Software Foundation, 2024). The new add-on extracts only vertex coordinates from polygon elements to construct a 3D model. The CityGML model imported by this new add-on can be directly visualized within a new IFC project established by *BlenderBIM* (Fig.9).

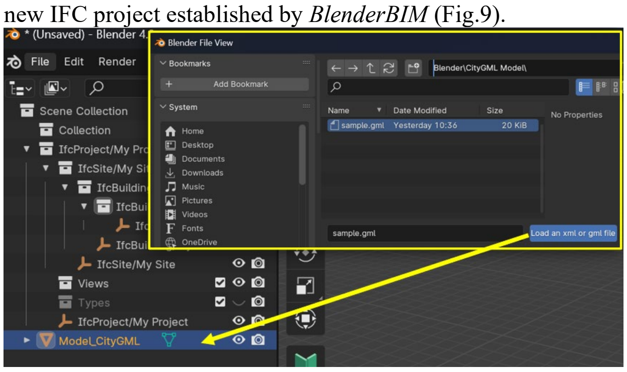


Figure 9. Loading an xml or gml file to a new IFC project

But the new add-on cannot adjust the geometry origin to the Blender World Origin (0,0,0). The other add-on is created to solve this problem. As shown in Fig.10, the panel *My Custom Panel label* contains six buttons. The add-on *Move Building to Origin* can move the geometry origin of selected object to the *Blender World Origin* (0,0,0) using *Blender* API *bpy.ops.object.origin_set*.

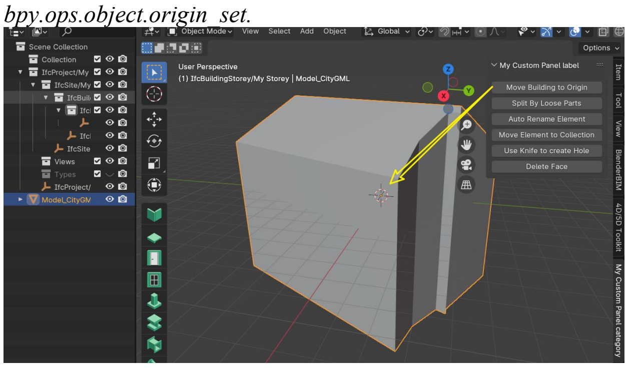


Figure 10. Moving CityGML model to *Blender World Origin*

IFC specifications define various classes of objects, representing different entities and components involved in the construction process. IFC classes cover a wide range of elements, such as Building Elements (Walls, floors, roofs, doors, windows, etc.), Spatial Elements (Spaces, zones, building storeys, etc.) and Structural Elements (Beams, columns, slabs, etc.). For example, a wall in the IFC schema is represented by the IfcWall class and the roof is represented by the IfcWall class. Therefore, we need to splite the imported CityGML model to independent fragments. The 3D geometry in a CityGML LoD2 model is often described using triangular or polygonal faces. Each face represents a flat surface and is defined by a set of vertices (points in 3D space) and associated properties like texture, color, or material. The add-on *Split By Loose Parts* can splite one object to independent (disconnected) fragment of the original mesh (Fig.11). This add-on is based on *Blender* API *bpy.ops.mesh.separate(type='LOOSE')*.

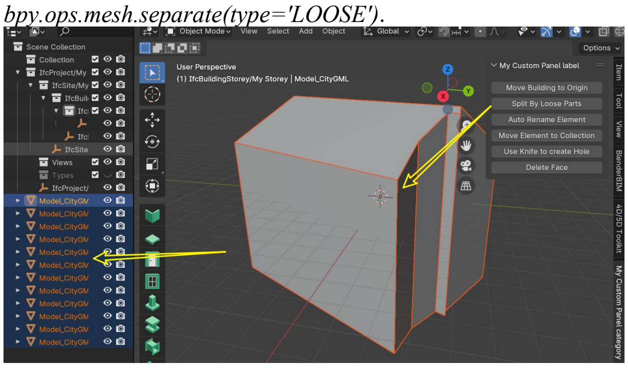


Figure 11. Spliting CityGML model to independent fragments

Through *BlenderBIM*, users have the capability to map individual components to the relevant IFC classes. However, certain CityGML LoD2 models are comprised of numerous discrete elements, making the task of manually categorizing each one into IFC classes a labor-intensive process. Consequently, the implementation of an automated renaming system becomes essential. By renaming each fragment in accordance with its spatial and geometric characteristics, we can generate descriptive names that serve as a basis for the subsequent automated classification process. As shown in Fig.12, the geometry origin of the moved CityGML LoD2 model is located at base point (0,0,0), the ground floor is located below the origin point (0,0,0).

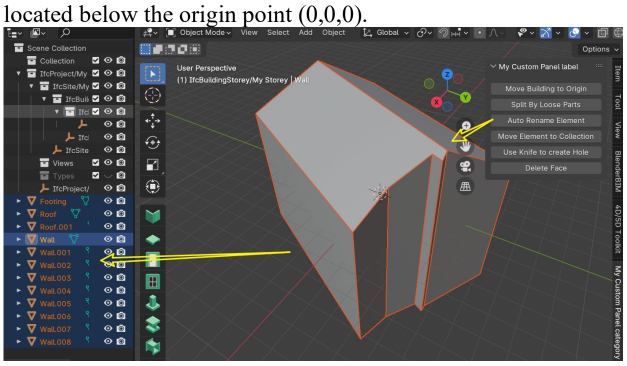


Figure 12. Automated renaming the independent fragments

Based on the construction of the CityGML LoD2 model, the algorithm in the add-on *Auto Rename Element* can classify each fragment according to the following principles: first, identify fragment located in the negative side of Z-axis and with a normal vector parallel to the Z-axis; these fragments are classified as Footing. Subsequently, for other fragments, if the angle between the fragment's normal vector and the Z-axis is greater than 80 degrees, the fragments is classified as Wall. In general, the normals of walls are typically perpendicular to the Z-axis, this means the angle between the normals and the Z-axis is 90 degrees. All remaining fragments can be categorized as Roof.

The next step is to place the renamed fragments under the Collection named "*IfcBuildingStorey/My Storey*". This Collection created by *BlenderBIM* contains all the necessary elements for the IFC model. In other words, only the meshes within this Collection can be successfully exported as IFC format elements. The add-on *Move Element to Collection* uses the *collection.objects.unlink(obj)* and *collection.objects.link(obj) Blender* API to achieve the relocation (Fig.13).

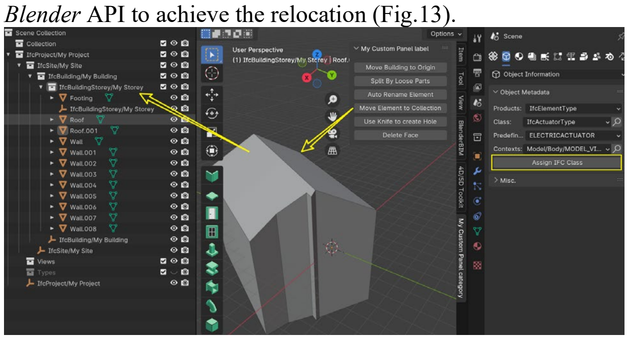


Figure 13. Moving renamed fragments to the Collection named "IfcBuildingStorey/My Storey".

In this work, the add-on *BlenderBIM* is modified to enable the rapid batch assignment of selected models as IFC classes. The class *AssignClass* of *BlenderBIM* contains the corresponding function for the button "*Assign IFC Class*". The installed *BlenderBIM* is generally located in the following local path:

AppData\Roaming\Blender Foundation\Blender\4.0\scripts\addons

We can directly open the corresponding *Python* file *operator.py* of the installed *BlenderBIM* and add if statements within the class *AssignClass* to achieve batch model assignment based on the name of mesh. When users click on button "*Assign IFC Class*" on the panel *Object Metadata* (see right side of the Fig.13), the meshes in the Collection "*IfcBuildingStorey/My Storey*" are automatically converted to corresponding IFC classes (Fig.14). After convertion, *BlenderBIM* provides users with the ability to work with a variety of IFC elements within the *Blender* interface. Users can add walls, floors, doors, windows, and other building components to to the previously converted IFC classes.

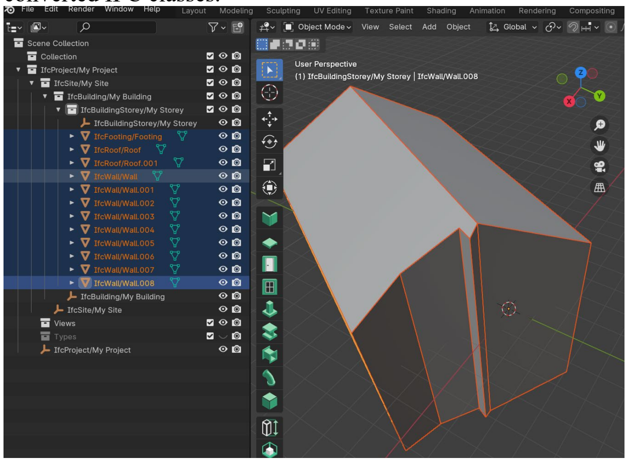


Figure 14. Automatically convert fragments to corresponding IFC classes

In CityGML LoD2 models, detailed openings such as doors and windows are typically not represented. However, the presence of openings results in a reduction of the surface area, which significantly impacts both material quantification and energy simulation during building performance analysis. Therefore, the add-on *Use Knife to create Hole* and *Delete Face* are developed to interactively subdivide (cut up) geometry by drawing lines or closed loops to create openings in the mesh for various purposes, including visualization, accurate modeling, and simulations, especially when dealing with architectural or structural representations. It's important to note that cutting up openings must be done before the mesh is converted into an IFC class. As shown in Fig.15, Cutting openings does not affect objects that have already been converted into IFC classes. After exporting the model to IFC format, the holes cut in the roof that have been converted to IFC Class cannot be retained.

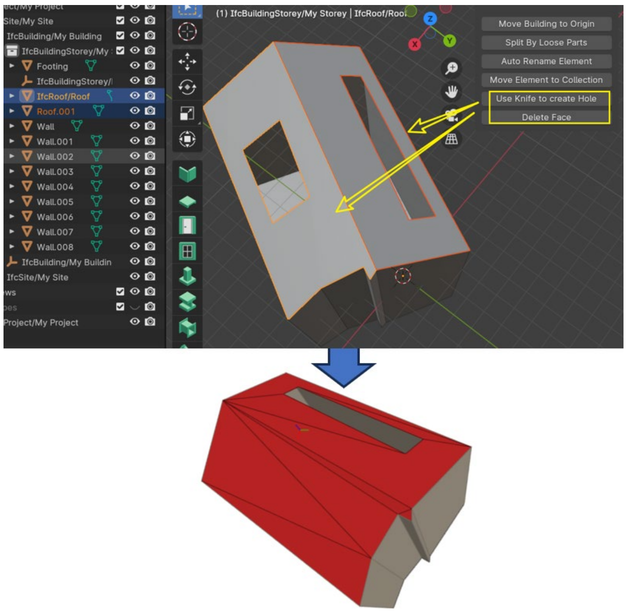


Figure 15. Knife Tool creates opening on mesh, but it does not work on the IFC element.

In this paper, the reconstructed LoD2 models are targeted towards existing buildings. To reconstruct the internal structure of the buildings, the basis relies on floorplans. The existing building plans are typically in the form of images or PDF formats. The open-source tool *floorplan to blender3d* can convert a floorplan to a 3d model, and this 3D model can be directly imported and used in *Blender* (Grebtsew, 2021). The core algorithm of this tool relies on Open Source Computer Vision Library (OpenCV), which is an open-source computer vision and machine learning software library. This tool is primarily deployed on Docker, making it very convenient to use. It will eventually generate a file in blend format and save it in a directory named "target" (Fig.16).

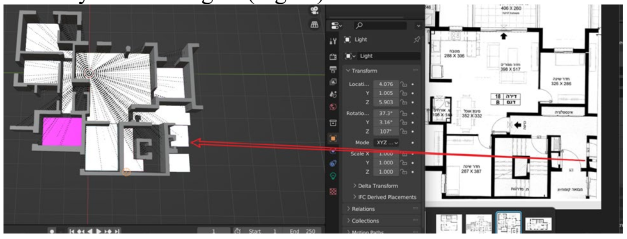
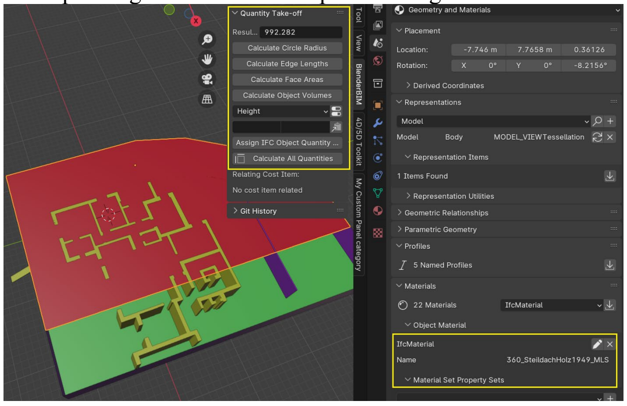


Figure 16. Automated converting floorplan to the 3D model using *Floorplan to Blender3d*

Next, we can combine this internal structure generated by *floorplan to blender3d* with previously 3D model within *Blender*. As shown in Fig.17, To facilitate the display of the internal structure, we intentionally offset the roof slightly. The corresponding LoD2 model is depicted in Fig.5.

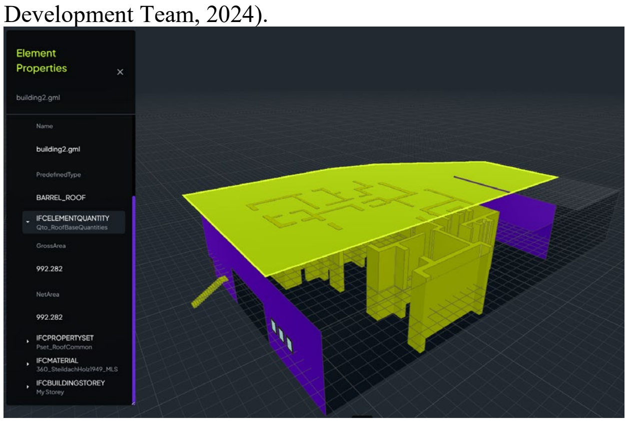


Resul... 992.282
Height
Relating Cost Item:
No cost item related

Placement
Location: -7.746 m 7.7658 m 0.36126
Rotation: X 0° Y 0° -8.2156°
Representations
Model
Model Body MODEL_VIEWTessellation
1 Items Found
Profiles
5 Named Profiles
Materials
22 Materials IfcMaterial
IfcMaterial
Name 360_SteildachHolz1949_MLS

Figure 17. The ultimately reconstructed IFC model with door, windows, hole, stair, slab, and interior walls.

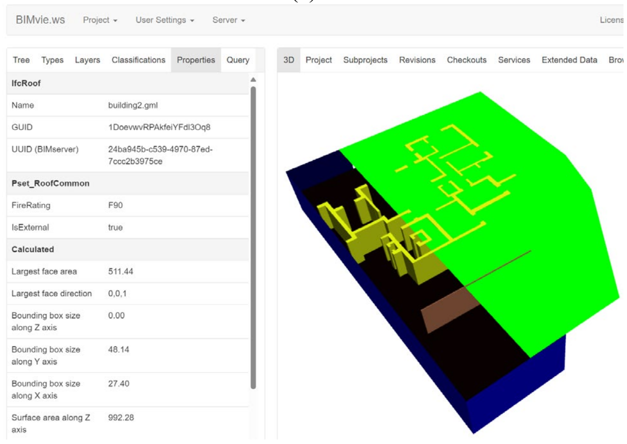
With *BlenderBIM*, users can also define the materials of building element through *IFCMaterial* tool, and the tool *Quantity Take off* has the capability to automatically calculate the area and volume of structures and assign this information to the structures (Fig.17). Subsequently, user can visualize this information through an open-source IFC viewer, such as *BIM Fragment* of *IFC.js* (BIM Fragments Development Team, 2024) or *Building Information Model server* (*BIMserver*) (BIMserver Development Team, 2024).



Element Properties
building2.gml

Name
building2.gml
PredefinedType
BARREL_ROOF
IFCELEMENTQUANTITY
Qto_RoofBaseQuantities
GrossArea
992.282
NetArea
992.282
IFCPROPERTYSET
Pset_RoofCommon
IFCMATERIAL
360_SteildachHolz1949_MLS
IFCBUILDINGSTOREY
My Storey

(a)



IfcRoof	
Name	building2.gml
GUID	1DoevwvRPAkfeiYFdl3Oq8
UUID (BIMserver)	24ba945b-c539-4970-87ed-7ccc2b3975ce
Pset_RoofCommon	
FireRating	F90
IsExternal	true
Calculated	
Largest face area	511.44
Largest face direction	0,0,1
Bounding box size along Z axis	0.00
Bounding box size along Y axis	48.14
Bounding box size along X axis	27.40
Surface area along Z axis	992.28

(b)

Figure 18. Displaying the reconstructed IFC model with properties through *BIM Fragment* (a) and BIMserver (b)

BIMserver can act as an IFC database, with special extra features like model checking, versioning, project structures, merging, etc. The main advantage of this approach is the ability to query, merge and filter the BIM-model and generate IFC output. *BIMserver* also contains function for multi-user support, multiple people can work on their own part of the dataset.

4. Discussion

Revit, *SketchUp*, and similar 3D model editors are often priced high due to several factors, and their popularity in the market can be attributed to a combination of features and usability. On the other hand, *Blender* provides an alternative for developers and users seeking a powerful 3D modeling tool with an open-source model. *Blender*'s strengths lie in its versatility, extensive feature set, and a community of developers contributing to its improvement. While it may not have the same industry-specific focus as some commercial tools, its accessibility and cost-free nature make it an attractive option for a broad range of users, particularly those in indie game development, animation, and hobbyist 3D modeling.
During the conversion process, we discovered that the clarity of the floorplan had an impact on the quality of the models generated by the tool *floorplan to blender3d*. If the floorplan is not clear, some structure will be lost, and manual modification will be required. Despite this minor issue, the tool does quickly convert 2D floorplan into usable 3D models.
In the work, some add-on functionalities could actually be consolidated into a single button. However, for the sake of convenience in demonstration, each functionality was assigned to its respective button.

5. Conclusion and Outlook

The paper demonstrates a new method for extracting, reconstructing and converting a simple LoD2 model into a complex IFC model using open-source software. The focus of this paper is not on the conversion of model formats, but rather on the reconstruction of the model's structure. To achieve this, a *Node.js* application was developed to visualize CityGML models on *CesiumJS* viewer and facilitate quick selection and extraction of CityGML model through association with the *BaseX* database. Subsequently, the open-source software *Blender* was considered the core editor for 3D model reconstruction, and corresponding add-ons were developed to ensure accurate import, editing, and format conversion of the model. In this study, *BlenderBIM* is not merely utilized in its standard form; rather, following its installation, modifications are made to the pre-installed *BlenderBIM* software: Additional code has been implemented to facilitate the automatic batch assignment of selected models as IFC classes based on the name of mesh. The paper also highlights the utilization of the open-source tool *floorplan to blender3d* to swiftly convert architectural floorplan into 3D structures, thereby supplementing the internal structures missing in the LoD2 model. The workflow described in this paper can solve the intended task described in the introduction. This workflow illustrates that *Blender*, as open-source software, possesses a highly powerful extensibility.
Artificial intelligence plays an increasingly important role in 3D modeling nowadays. *Blender* has a flexible Python controlled interface. And it supports add-on development, particularly in the widely popular programming language Python, our plan for future work involves integrating artificial intelligence models into *Blender*, enabling users to rapidly create 3D models.
Moreover, we plan to integrate the *BIM Fragment* and *BIMserver* to our *Node.js* application to link the CityGML model with the reconstructed IFC model. This makes it easier for users to view and manage the models in the same platform.

References

Autodesk, 2024. Revit APIs. revitapidocs.com/2024/news (1 February 2024).

BaseX Development Team, 2024. BaseX Software, Version 10.7. basex.org (1 February 2024).

BIM Fragment Development Team, 2024. BIM Fragment Software, Version 2.0.0. github.com/IFCjs/fragment (1 February 2024).

BIMserver Development Team, 2024. BIMserver Software, Version 1.5.184. bimserver.org (1 February 2024).

Biljecki, F., Ledoux, H., and Stoter, J., 2016: Generation of multi-LoD 3D city models in CityGML with the procedural modeling engine Random3DCity, *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, IV-4/W1, 51–59, doi.org/10.5194/isprs-annals-IV-4-W1-51-2016

Citygml Wiki, Commercial Software, 2020. citygmlwiki.org/index.php/Commercial_Software (1 February 2024).

Dealga M., 2020. Import CityGML. github.com/ppaawweeuu/Import_CityGML (1 February 2024).

Goetz, M. and Zipf, A., 2012. Towards Defining a Framework for the Automatic Derivation of 3D CityGML Models from Volunteered Geographic Information. *International Journal of 3-D Information Modeling (IJ3DIM)*, 1(2), 1-16. doi.org/10.4018/ij3dim.2012040101

Goetz, M. 2013. Towards generating highly detailed 3D CityGML models from OpenStreetMap, *International Journal of Geographical Information Science*. 27 (5), 845-865.

Grebtsew, 2021. floorplan to blender. github.com/grebtsew/FloorplanToBlender3d (1 February 2024).

Harms, J., 2008: Räumliche Datenbanken und GML. Bachelorarbeit an der Technischen Universität Wien. Ausgeführt am Institut für Rechnergestützte Automation Forschungsgruppe Industrial Software. Wien, Februar 2008.

Hakim, A., der Vaart, J.v., Ohori, K.A., Stoter, J., 2024. Development of a Geo to BIM Converter: CityJSON Importer Plugin for Autodesk Revit. In: Kolbe, T.H., Donaubauer, A., Beil, C. (eds) Recent Advances in 3D Geoinformation Science. 3DGeoInfo 2023. Lecture Notes in Geoinformation and Cartography. Springer, Cham. doi.org/10.1007/978-3-031-43699-4_1

Hensel, S., Goebbels, S., and Kada, M., 2019: Facade reconstruction for textured LoD2 CityGML models based on deep learning and mixed integer linear programming, *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, IV-2/W5, 37–44, doi.org/10.5194/isprs-annals-IV-2-W5-37-2019.

Hessische Verwaltung für Bodenmanagement und Geoinformation, 3D-Gebäudemodelle. hvbg.hessen.de/landesvermessung/geotopographie/3d-daten/3d-gebaeudemodelle (1 February 2024).

Kudrass, T., Conrad, M., 2002: XML-Based Data Management and Multimedia Engineering — EDBT 2002 Workshops Lecture Notes in Computer Science Volume 2490, 2002, pp. 210-227.

Landesamt für Geoinformation und Landesvermessung Niedersachsen (LGLN), 3D-Gebäudemodelle (LoD1 und LoD2). lgln.niedersachsen.de/startseite/geodaten_karten/3d_geobasisdaten/3d_gebaudemodelle/3d-gebaeudemodelle-142891.html (1 February 2024).

Landesamt für Geobasisinformation Sachsen [GeoSN], Downloadbereich Digitale 3D-Stadtmodelle. geodaten.sachsen.de/downloadbereich-digitale-3d-stadtmodelle-4875.html (1 February 2024).

Nagel C., Stadler, A., 2008: Die Oracle-Schnittstelle des Berliner 3D-Stadtmodells. In: Entwicklerforum Geoinformationstechnik 2008.

OpenStreetMap Wiki, Proposal:IndoorOSM. wiki.openstreetmap.org/wiki/Proposal:IndoorOSM (1 February 2024).

Park S, Kim H., 2021: 3DPlanNet: Generating 3D Models from 2D Floor Plan Images Using Ensemble Methods. Electronics. 10(22):2729. doi.org/10.3390/electronics10222729

Pavlovic-Lažetic, G., 2006: Native XML databases vs. relational databases in dealing with XML documents. In: Kragujevac J. Math. 30 (2007), pp. 181-199

Python Software Foundation, 2024. xml.etree.ElementTree - The ElementTree XML API. docs.python.org/3/library/xml.etree.elementtree.html (1 February 2024).

Salheb, N., Arroyo Ohori, K., and Stoter, J., 2020: Automatic conversion of CityGML to IFC, *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLIV-4/W1-2020, 127–134, doi.org/10.5194/isprs-archives-XLIV-4-W1-2020-127-2020.

Vidanapathirana M., Wu Q., Furukawa Y., Chang A. and Savva M., 2021. Plan2Scene: Converting Floorplans to 3D Scenes, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021 pp. 10728-10737

Vorbeck T. and Wills N., 2022: The Current State of BIM on Existing Buildings: The Case of Germany.

Wang, Z. and Zipf, A., 2017: Using OpenStreetMap data to generate building models with their inner structures for 3D maps, *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, IV-2/W4, 411–416, doi.org/10.5194/isprs-annals-IV-2-W4-411-2017.

Yao, Z., Nagel, C., Kunde, F. et al., 2018: 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, 5 (2018). doi.org/10.1186/s40965-018-0046-7

Zhang, S., Gan, J., Xu, J., Lv, G., 2008: Study on NXD based GML storage model. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*. Volume 37. Part B4. Beijing 2008