FACULTY OF ENGINEERING
AND ARCHITECTURE

Pre-culling geometric linked building data for lightweight viewers

Linked Data

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: architectuur

Supervisor: Prof. ir.-arch. Paulus Present

Counselors: Ir.-arch. Jeroen Werbrouck
Prof. dr. ir. arch. Ruben Verstraeten

Philippe Soubrier 01702837 philippe.soubrier@ugent.be
Academic year: 2022–2023

GHENT
UNIVERSITY

Contents

List of Figures

List of Tables

Short abstract

This is my short abstract.

Abstract

This is my abstract

Chapter 1

Introduction

From 2D, to 3D and now to BIM. The evolution of the Architecture, Engineering and Construction (AEC) industry has been a long and complex one. The introduction of 3D modeling was the first major step in the industry's evolution, as it allowed for more accurate representations of buildings. No longer solely relying on 2D drawings, a 3D model of a building can be used to create various representations, from a simple 2D floor plan to a full 3D model. Following the adoption of 3D modeling, the implementation of Building Information Modelling (BIM) emerged as another significant milestone. BIM adds an extra layer of information on top of the 3D model. As the digital representation of a building's physical and functional characteristics, BIM serves as a repository for semantics originating from various applications throughout the design and construction processes, including cost estimation, energy analysis, and production planning.
-> needs refs to support point (about infustries in AEC)

However, as mentioned in Werbrouck, 2018, the next challenge for the AEC industry is related to the ecosystem of current BIM softwares, which remains closed off to other disciplines. This data management challenge is currently being adressed by the Linked Building Data Community Group (LBD-CG) and other research entities, such as the University of Ghent, through the use of Web of Data technologies[1]. This emerging milestone will be discussed in this thesis under the term Linked Data BIM (LDBIM).

Proposal

Each of these evolutions has brought, and will continue to bring, a significant amount of data together. This volume is expected to grow exponentially in the future as the industry shifts towards a more digital approach and opens up to other stakeholders. The data graphs will not only expand in terms of semantics but also in geometry. This makes visual querying, or simply put, 3D exploration of models, an increasingly diffi-

[1]W3C, 2023.

cult task. Especially when looking at newer devices used in the industry such as mobile phones, and tablets, which are becoming more and more powerful, but still have limited computational resources in comparison to office computers.

To bring this volume of geometric data in perspective, Table 1.1 shows the size of the test-models used in Johansson et al., 2015 , a study from 2015 on the performance of BIM viewers for large models with the following description:

“ Although the Hotel model contains some structural elements they are primarily architectural models. As such, no Mechanical, Electrical or Plumbing (MEP) data is present. However, all models except the Hospital contain furniture and other interior equipment. ” (Johansson et al., 2015)

Model	# of triangles	# of objects	# of geometry batches
Library	3 685 748	7318	11 195
Student House	11 737 251	17 674	33 455
Hospital	2 344 968	18627	22 265
Hotel	7 200 901	41 893	62 624

Table 1.1: Size of test-models in Johansson et al., 2015

These models demonstrate how basic BIM models can already contain a significant amount of data. LDBIM will not only bring together new stakeholders but also have the capability to keep track of multiple geometry versions for each object, should they occur. Therefore, this thesis proposes a new approach to the visual querying of LDBIM models, wherein viewers will not have to load the entire model into memory. Instead, after filtering at the source, only the geometry needed for the visual tasks at hand will be loaded, while maintaining the original link to each resource for further processing and use cases. This filtering step is commonly referred to as culling in the computer graphics industry and is illustrated in Figures 1.1 and 1.2.

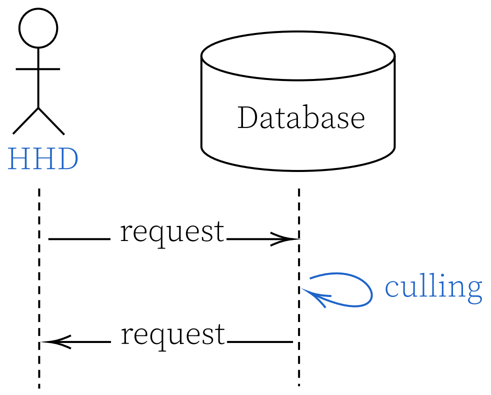

Figure 1.1: Sequence diagram - basic concept

Figure 1.1 illustrates the basic idea of this thesis, presenting an extra step in the communication between a user, represented here by a Hand Held Device (HHD), and a database storing the model. An HHD has been chosen to exemplify a low-powered device used in the field, which requires a lightweight 3D viewer to visualize and explore the digital twin of the building. The HHD is assumed to have no knowledge of the LDBIM model and only receives the geometry that needs to be displayed from the database. On the other hand, the database is assumed to possess, or have access to, all the knowledge of the model and the necessary semantics to perform the culling.

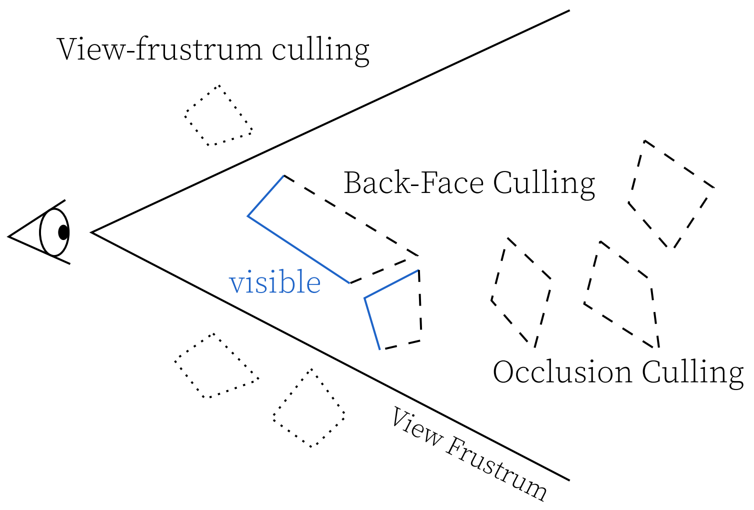

Figure 1.2: Illustration of culling principle, based on Cohen-Or et al., 2003

Figure 1.2 showcases multiple culling techniques to showcase some culling principles. The first technique, *frustum culling*, is used to determine which objects are visible to the user. The second technique, *occlusion culling*, is used to determine which objects are occluded by or behind other objects. And lastly, *back-face culling*, is used to determine which faces, and not whole objects, are facing away from the user.

1.1 Linked Data

As mentioned in the Introduction, the evolution from BIM to LDBIM is an evolution of the data *management* layer. “Linked Data”, as stated by the World Wide Web Consortium (W3C), is a collection of interrelated datasets on the Web, formatted in a standard way that is accessible and manageable by Semantic Web tools. The same applies to the relationships among them.[2] The following collection of Semantic Web technologies explores the required environment to achieve this goal.

1.1.1 RDF and triples

At the core of the Semantic Web is the Resource Description Framework (RDF), a data model for describing resources on the Web. RDF is a graph data model that consists of

[2]W3C, 2015b.

triples, which are statements about resources. A triple consists of a subject, a predicate, and an object. The subject is the resource that is being described, the predicate is the property of the subject, and the object is the value of the property. Both the predicate and the object can, in turn, become the subjects of other triples. Listing 1 shows an example of an RDF database described in the Turtle format.

```
@prefix fog: <https://w3id.org/fog#> .
@prefix omg: <https://w3id.org/omg#> .
@prefix bot: <https://w3id.org/bot#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix flupke:  <http://flupke.archi#> .

flupke:room1 rdf:type bot:Zone ;
    bot:containsElement flupke:coneOBJ ;
    bot:containsElement flupke:cubeGLTF .

flupke:coneOBJ omg:hasGeometry flupke:coneOBJ_geometry-1 ;
    rdf:type bot:Element .

flupke:cubeGLTF omg:hasGeometry flupke:cubeGLTF_geometry-1 ;
    rdf:type bot:Element .

flupke:coneOBJ_geometry-1 rdf:type omg:Geometry ;
    fog:asObj_v3.0-obj
    ↪ "https://raw.githubusercontent.com/flol3622/AR-Linked-BIM-viewer/⌋
    ↪ main/public/assets/database_1/coneOBJ.obj"^^xsd:anyURI
    ↪ .

flupke:cubeGLTF_geometry-1 rdf:type omg:Geometry ;
    fog:asGltf_v1.0-gltf
    ↪ "https://raw.githubusercontent.com/flol3622/AR-Linked-BIM-viewer/⌋
    ↪ main/public/assets/database_1/cubeGLTF.gltf"^^xsd:anyURI
    ↪ .
```

Listing 1: Example of an RDF database in turtle format

The basic, yet versatile, structure of a triple is illustrated in Figure 1.3. Both the subject and object are considered as nodes in the data graph, and they are linked by the predicate, which is referred to as an edge. Multiple triples can thus create and link multiple nodes or enrich a connection between two nodes by creating new edges between them. Each element contains a single resource that can be one of the three types: a URI, a literal, or a blank node. A Uniform Resource Identifier (URI) points to a resource on the web and, as its name states, is unique and unambiguous, thus enabling queries and reasoning of the same nature. A literal is a value, and a blank node is an anonymous

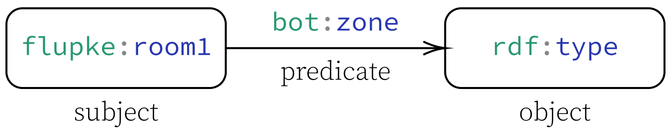

Figure 1.3: Triple structure

resource, sometimes used as a placeholder when the exact resource is not known or not necessary to specify. Due to their nature, a subject must be either a URI or a blank node, a predicate exclusively a URI, and the object may be any of the three types. As URI descriptions can be very long, a prefix can be used to shorten them. This is illustrated in Listing 1 with the `@prefix bot: <https://w3id.org/bot#>`, which declares that `bot:Zone` refers, in its full length, to the address `<https://w3id.org/bot#Zone>`.

This basic concept can be extrapolated to describe and store any kind of data. The advantage for the AEC industry would be to allow any stakeholders to describe and enrich the knowledge base of a building.

1.1.2 Ontologies and reasoning

When looking at Listing 1, a distinction can be made between two types of statements: some refer to classes or properties, such as `flupke:room1`, while others refer to facts associated with them, like `rdf:type`. The former is referred to as TBox for "terminology", and the latter is referred to as ABox for "assertions". The TBox is the part of the ontology that describes the classes and properties of the domain, while the ABox is the part of the ontology that describes the facts about the domain.

By developing an ontology, the domain of interest and the relationships between the classes and properties can be described. This is achieved by defining the classes and properties of the domain and their relationships. The ontology is then used to reason about the domain, inferring new facts based on the ontology and the existing facts within the domain. This is done by a reasoner, which is software capable of performing the reasoning itself on the ontology and associated data. As mentioned, the reasoner can be used to infer new facts, check if created facts are consistent with the ontology, and check if the ontology itself is consistent.[3] It is often integrated with RDF databases, also known as triplestores or graph databases.

Classes, properties, and their relationships can be defined using Resource Description Framework Schema (RDFS), which is a vocabulary for describing RDF schemas using a basic set of constructs. As an extension of RDFS, Web Ontology Language (OWL) is a vocabulary for describing ontologies using a more expressive set of constructs tailored to the needs of ontologies. Both RDFS and OWL are considered to be formal ontologies themselves, as they describe the classes and properties of the domain of RDF.

[3] W3C, 2015a.

1.1.3 Triplestores and SPARQL

As briefly discussed in 1.1.2, triplestores are RDF databases that store data in the form of a graph. They are used to store and query Linked Data and are often integrated with a reasoner. The data itself is retrieved and modified using the SPARQL Protocol and RDF Query Language (SPARQL).[4] In contrast to Structured Query Language (SQL), SPARQL queries are able to work across multiple triplestores, called SPARQL endpoints. These are known as federated queries, and their results are combined into a single result set. This is useful when the data is distributed across multiple triplestores in a decentralized manner.[5] For example, multiple stakeholders participating in a project, each with their own database.

1.1.4 Complexity of the data graph

The complexity of the data graph is a major concern when working with LDBIM. This section discusses the origins of the different sources of geometric data that enrich it.

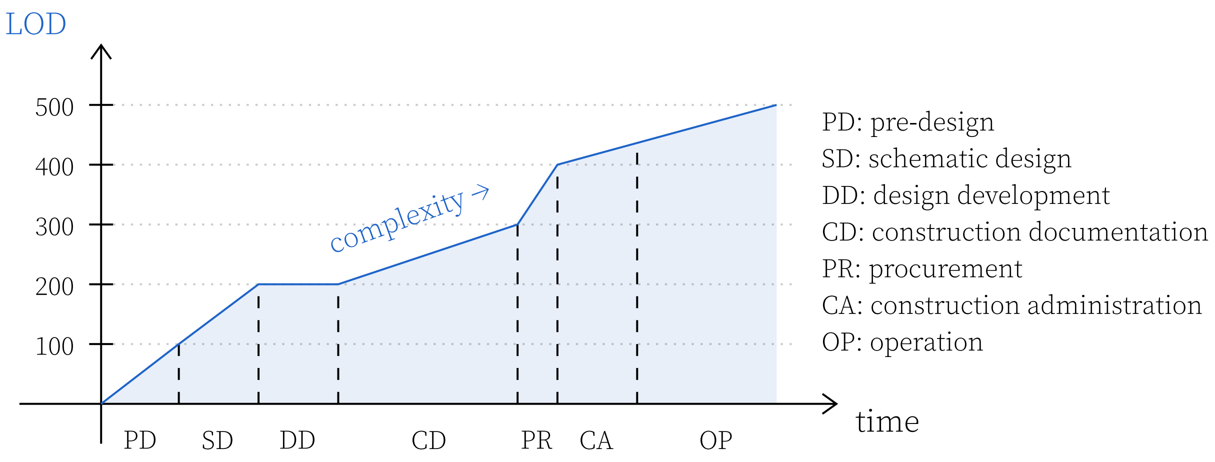

Figure 1.4: Evolution of LOD during the life-cycle of a building.
Based upon the Macleamy Curve (Ilozor & Kelly, 2012)

1.1.4.1 BIM geometry

The 3D model of a building consists of a multitude of sub-models, describing objects for all the different stakeholders participating in the project. Some describe very large objects, and some very small parts. Both can be defined in their most simple and abstract form or have an intricate and complex geometry. For instance, a door can simply be defined as a box, or up to the level of the screw-thread for the hinge system. The level of abstraction is here described as the Level of Detail (LOD), which is most of

[4] W3C, 2015c.
[5] Ontotext, 2022.

the time pre-selected for the needs of a BIM model, and is applied throughout a single model.

As shown in Figure 1.4, a standard BIM workflow goes through multiple phases, each with their associated model and LOD. This makes it an important concept in the AEC industry, as it allows for a very efficient workflow. The modeling step is approached from a top-down perspective, starting with rougher geometries describing the broader ideas of a concept model and evolving to a more refined model for the construction documentation phase. As the last and longest-standing model, a higher LOD can be used to describe subtle changes in the evolution of a building during the operation phase.

1.1.4.2 LDBIM geometry

The interconnectivity of semantics can also be applied to geometry descriptions. This could allow the co-existence of multiple LODs in a single model database. Besides storing the evolution of a single element's geometry, it enables the linking of the different LODs, described in 1.1.4.1, to each other. Not only that, but extending onto the size of the models described in Table 1.1, already existing Mechanical, Electrical and Plumbing (MEP), structural, alongside many other stakeholders' geometry can be added.

1.2 Research questions

Johansson et al., 2015 presented in their paper a new BIM viewer equipped with the powerful Coherent Hierarchical Culling algorithm (CHC)++. This is a third-generation occlusion culling algorithm developed by Mattausch et al., 2008a, the first being the CHC (Bittner et al., 2004), followed by the Near Optimal Hierarchical Culling (NOHC) (Mattausch et al., 2008b). Their conclusion stated that although occlusion culling is very efficient, it is still bound to the scene size, which is limited by hardware capabilities. More precisely, the Graphics Processing Unit (GPU), Video Random Access Memory (VRAM), and Random Access Memory (RAM) capacities.

This thesis proposes the introduction of culling algorithm technology within the context of LDBIM to address the previously mentioned issue of the scene's size, by culling the scene at its source prior to sending it to the viewer. As culling algorithms have been extensively researched and continue to evolve, as described in the previous paragraph, the research questions in this thesis concentrate on assessing the feasibility of introducing such algorithms in LDBIM. It aims to propose a set of possible solutions tailored to this specific problem, while highlighting possibilities for future research and specific use cases.

1.2.1 To what extent can LDBIM geometry be culled to be streamed to lightweight viewers?

This thesis focuses on computing with data snippets or triples inside a LDBIM model, not within. Meaning that the smallest unit of data that can be culled is the one described in one triple, which is in the most likely scenario, a single LOD of a single element. It implies that geometry is defined and separated at the object-level. It also implies that culling techniques such as back-face culling will not be handled in this thesis, and will be left to the viewer itself, not the database.

Which snippets of data are needed by the viewer? Is part of the question. The basic needs of the viewer consist firstly of the geometry itself, selecting the right geometry format for the application as well as the additional visual information such as color, texture, etc. Secondly, the identifier of each element is of crucial importance to maintain the link to other semantic resources in the graph. This enables the viewer to retrieve those resources for a multitude of use cases, transforming it into a user-friendly visual query tool.

1.2.2 Can existing semantics and ontologies be used to feed possible culling algorithms?

In contrast to the computer graphics industry, this interconnected context already contains both explicit and implicit, through inferencing, relationships inside the graph. Similarly to Johansson and Roupé, 2009 and their paper where they used the semantics of a BIM model in Industry Foundation Classes (IFC) format to develop culling techniques. However, this thesis will focus on the use of Semantic Web resources. Therefore, it will analyze both AEC-specific and AEC-related ontologies, such as Geographic Information System (GIS)-related, to see if they can be used to feed culling algorithms.

Chapter 2

State of the art

As mentioned in Johansson et al., 2015, existing research on the performance of currently used BIM viewers is quite limited. This state-of-the-art research will, therefore, focus on the overall features of some promising newer viewers and the ontologies that will be used in this thesis.

2.1 Existing BIM viewers and ontologies

2.1.1 Qoniq and LOD Streaming for BIM

Qonic focuses on developing an open platform BIM viewer. With the use of Unity to enable cross-platform compatibility, they focused on two main aspects: performance and aesthetics. The latter refers to the visual quality of the viewer, offering both a seamless experience for the viewer as well as a pleasant one, with, for example, the implementation of ambient lighting and shadow castings. The first and most researched aspect of their viewer, the performance, is mainly focusing on a LOD culling algorithm.
(T. Strobbe, personal communication, November 25, 2022)

2.1.1.1 Qoniq's approach to LOD streaming

Their core research is developing a dynamic LOD streaming model. Starting from the geometry and semantics of an IFC file, they compute an LOD hierarchy tree of the model. Through multiple mesh decimation algorithms, they reduce the number of triangles of each object's mesh, regardless of the semantics associated with that object. On top of that, a filtering algorithm is implemented in the streaming model to filter out objects, regarding their semantics, that are not relevant to the current camera position. In doing so, they both reduce the size of models far from the viewpoint and evaluate the need to show certain objects based on their nature, extracted from semantics in the IFC file, and their distance to the camera. The resulting dynamic LOD streaming

model is reevaluated at each camera move in Unity.
(T. Strobbe, personal communication, November 25, 2022)

Unity was chosen as it allows for writing once and deploying everywhere. This means that the viewer can be used on any platform, including mobile devices and browsers. The performance results are thus related to the hardware capabilities of each device, with the exception of the browser, where the performance of Unity's WebGL build is limited to a scene size of 2Gb.[1]

2.1.1.2 Advantages and trade-offs

Being able to run on many platforms, offering a smooth viewer experience and a pleasing aesthetic makes it an ideal candidate for lightweight viewers on the job site. However, the LOD library has to be computed on every model update. The decimation algorithms are furthermore computational results that are not humanly reviewed. This means that the quality of the resulting meshes is not guaranteed for the lower LODs, which are, as illustrated in Figure 1.4, already modeled in previous design phases. LDBIM could, by interconnection, recall previous LODs in the viewer's scene. Without the need for computational remodeling. Nevertheless, Qonic serves as this thesis's goal, outside the LDBIM context.

2.1.2 ld-bim.web.app

"The purpose of the app is to showcase our LBD toolset and to demonstrate the capabilities of Linked Building Data to newcomers."[2]

https://ld-bim.web.app/ demonstrates a viewer built around an RDF database. It separates the data from an IFC file into semantics, stored in the previously mentioned graph, and a glTF model, together with a JavaScript Object Notation (JSON) file containing a reference table. Extra local or remote graphs can be added to the User Interface (UI). As it contains a SPARQL engine to query and visualize, in the form of highlighting, the results of the query in a 3D viewer. The viewer is based on the ifc.js project, which is itself based on the three.js 3D JavaScript library.

2.1.3 AEC related ontologies

As mentioned in the second research question 1.2.2, this section will discuss AEC-related technologies, some of which are not yet approved by the W3C but are actively researched by the LBD-CG[3]

[1]Unity, 2023.
[2]Rasmussen and Schlachter, n.d.
[3]LBD-CG, 2022.

2.1.3.1 BOT

The BOT proposes a set of classes and properties, "which provides a high-level description of the topology of buildings including storeys and spaces, the building elements they may contain, and the 3D mesh geometry of these spaces and elements." (Rasmussen et al., 2020), as illustrated in Figure 2.1. This high-level description could be fed to portal-culling algorithms in a situation where the visibility is contained within one `bot:Space` or `bot:Storey`, or it could extend the scope to `bot:adjacentZone`[4]. Additionally, it could play a part in the construction of the Bounding Volume Hierarchy (BVH) needed for other occlusion culling algorithms, such as the CHC++ (Johansson et al., 2015).

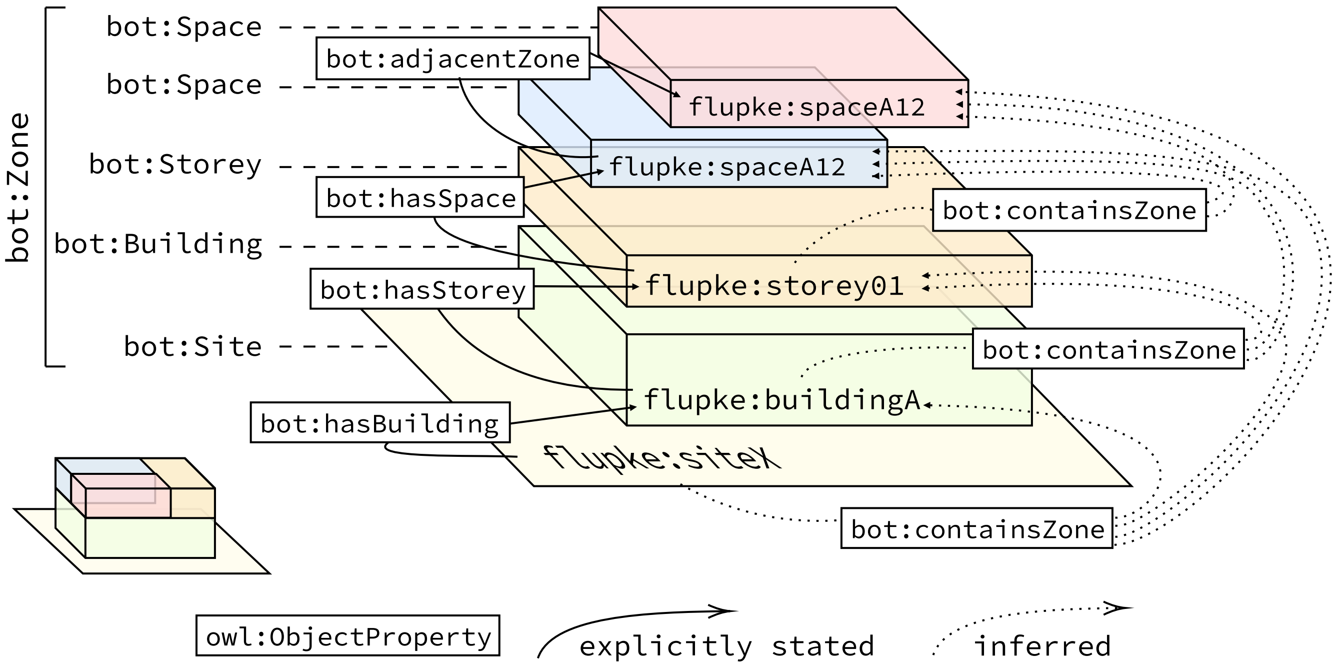

Figure 2.1: Illustration of the BOT ontology based on Rasmussen et al., 2020.

2.1.3.2 FOG and OMG

With the help of File Ontology for Geometry formats (FOG) and Object Management Group (OMG), geometry descriptions can be linked in the data graph. The innovation lies in the choice to store it either inside or outside the graph, by means of one triple referring to a literal or an URI. Listing 1 showcases multiple examples of objects assigned with a geometry description using an URI(Bonduel et al., 2019).

```
flupke:coneOBJ_geometry-1 fog:asObj_v3.0-obj "https://..."^^xsd:anyURI .
```

Listing 2: Example of FOG usage

[4]Linietsky et al., 2023.

Listing 2 describes a subject of datatype `xsd:anyURI` from the Extensible Markup Language (XML) Schema Definition (XSD)[5]. The versatile approach of Bonduel et al., 2019 also proposes the following datatypes: `xsd:string` for American Standard Code for Information Interchange (ASCII)-based geometry descriptions or `xsd:base64Binary` for binary geometry descriptions.

The format of the geometry is assigned directly by the predicate in Listing 2, which is `fog:asObj_v3.0-obj`. This further infers the statements in Listing 3.[6]

```
flupke:coneOBJ_geometry-1 fog:asObj "https://..."^^xsd:anyURI ;
    ex:LOD "100"^^xsd:integer .
```

Listing 3: FOG inference examples

Bonduel et al., 2019 refers to the proposal of the LBD-CG stated in Wagner et al., 2019 “to allow the modelling of properties on three levels”. The first and second levels are illustrated in Figure 2.2.

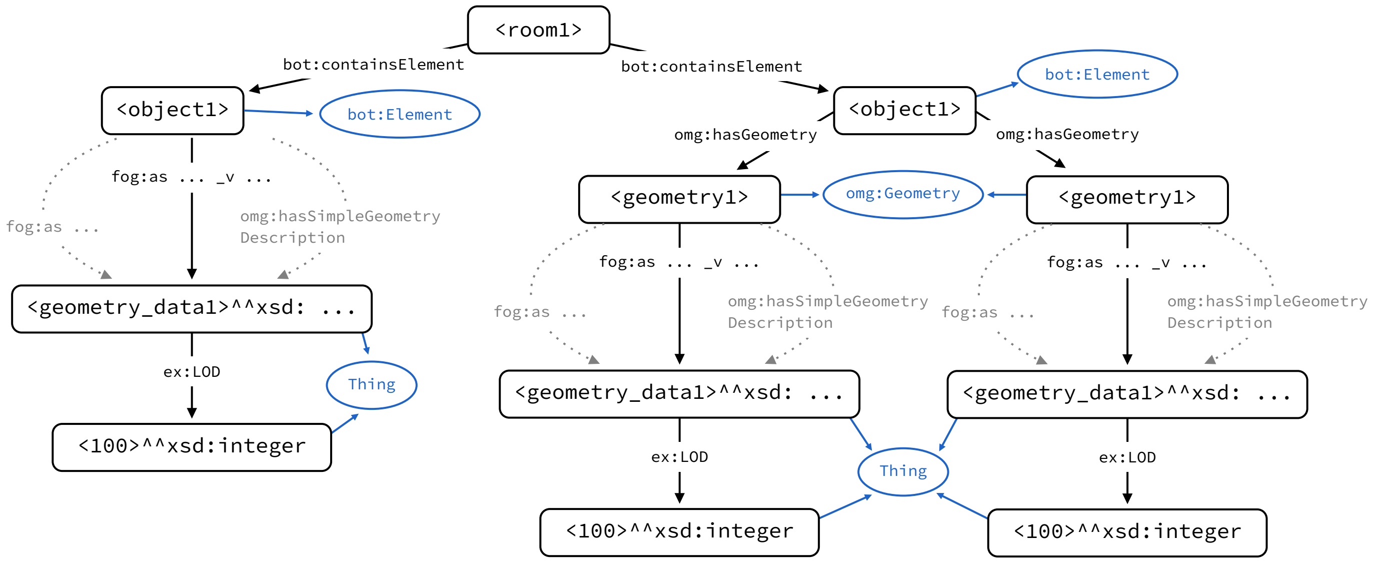

Figure 2.2: Illustration of the FOG and OMG ontology based on. LOD can’t be assigned to literal, needs to be changed

2.1.4 GIS related ontologies

-> Highlighting maturity and usefull data

2.1.4.1 geoSPARQL

-> 2D Limitations

[5]Carrol and Pan, 2006.
[6]Bonduel et al., 2020.

2.2 On the market viewers comparison

- DDS CAD Viewer
- Tekla BIMsight
- Autodesk Navisworks
- Solibri Model Viewer

2.2.1 General Features

-> Johansson et al., 2015
–> table 3 about Acceleration techniques

2.2.2 Interoperability

2.2.3 Scalability

2.2.4 Collaboration and Data Sharing

2.2.5 Customization and Extensibility

Chapter 3

Culling approaches

3.1 AEC related ontologies

3.1.1 BOT

3.1.2 FOG and OMG

3.2 GIS related ontologies

3.2.1 geoSPARQL

Chapter 4

Setup

4.1 Participants

This is a diagram:

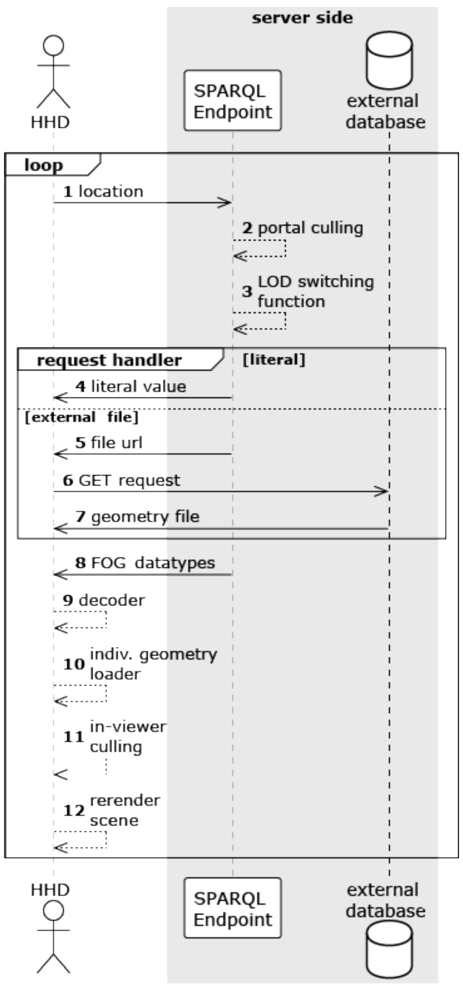

Figure 4.1: Sequence diagram

4.2 Framework

4.2.1 Nextjs

4.3 Querying

4.3.1 Front-end

4.3.2 Back-end

4.4 Rendering

4.4.1 Xeokit SDK

List of Acronyms

AEC	Architecture, Engineering and Construction	6
ASCII	American Standard Code for Information Interchange	17
BIM	Building Information Modelling	6
BOT	Building Topology Ontology	3
BVH	Bounding Volume Hierarchy	16
CHC	Coherent Hierarchical Culling algorithm	12
FOG	File Ontology for Geometry formats	16
GIS	Geographic Information System	13
GPU	Graphics Processing Unit	12
HHD	Hand Held Device	8
IFC	Industry Foundation Classes	13
JSON	JavaScript Object Notation	15
LBD-CG	Linked Building Data Community Group	6
LDBIM	Linked Data BIM	6
LOD	Level of Detail	11
MEP	Mechanical, Electrical and Plumbing	12
NOHC	Near Optimal Hierarchical Culling	12
OMG	Object Management Group	16
OWL	Web Ontology Language	10
RAM	Random Access Memory	12
RDF	Resource Description Framework	8
RDFS	Resource Description Framework Schema	10
SDK	Software Development Kit	
SPARQL	SPARQL Protocol and RDF Query Language	11
SQL	Structured Query Language	11
UI	User Interface	15
URI	Uniform Resource Identifier	9
VRAM	Video Random Access Memory	12
W3C	World Wide Web Consortium	8

XSD	XML Schema Definition	17
XML	Extensible Markup Language	17

References

Bittner, J., Wimmer, M., Piringer, H., & Purgathofer, W. (2004). Coherent hierarchical culling: Hardware occlusion queries made useful. *Computer Graphics Forum*, *23*, 615–624. https://doi.org/10.1111/J.1467-86

Bonduel, M., Wagner, A., Pauwels, P., Vergauwen, M., & Klein, R. (2019). *Including widespread geometry formats in semantic graphs using rdf literals*. http://lib.ugent.be/catalog/pug01:8633665

Cohen-Or, D., Chrysanthou, Y., Silva, C., & Durand, F. (2003). A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, *9*, 412–431. https://doi.org/10.1109/TVCG.2003.1207447

Ilozor, B. D., & Kelly, D. J. (2012). Building information modeling and integrated project delivery in the commercial construction industry: A conceptual study. *Journal of Engineering, Project, and Production Management*, *2*, 23–36. https://doi.org/10.32738/JEPPM.201201.0004

Johansson, M., & Roupé, M. (2009). *Efficient real-time rendering of building information models*. https://www.researchgate.net/publication/220758081

Johansson, M., Roupé, M., & Bosch-Sijtsema, P. (2015). Real-time visualization of building information models (bim). *Automation in Construction*, *54*, 69–82. https://doi.org/10.1016/j.autcon.2015.03.018

Mattausch, O., Bittner, J., & Wimmer, M. (2008a). Chc++: Coherent hierarchical culling revisited. *Computer Graphics Forum*, *27*, 221–230. https://www.academia.edu/14388994

Mattausch, O., Bittner, J., & Wimmer, M. (2008b). Chc++: Coherent hierarchical culling revisited. *Computer Graphics Forum*, *27*, 221–230. https://www.academia.edu/14388994

Rasmussen, M. H., Lefrançois, M., Schneider, G., & Pauwels, P. (2020). Bot: The building topology ontology of the w3c linked building data group. *Semantic Web*, *11*, 1–20. https://www.researchgate.net/publication/342802332

Wagner, A., Bonduel, M., Pauwels, P., & Uwe, R. (2019). Relating geometry descriptions to its derivatives on the web. *Proceedings of the 2019 European Conference for Computing in Construction*, 304–313. http://dx.doi.org/10.35490/ec3.2019.146

Werbrouck, J. (2018). *Linking data : Semantic enrichment of the existing building geometry*. http://lib.ugent.be/catalog/rug01:002494740

Referenced webistes

LBD-CG. (2022). *A list of ontologies related to linked building data*. https://github.com/w3c-lbd-cg/ontologies

Bonduel, M., Wagner, A., & Pauwels, P. (2020). *Fog: File ontology for geometry formats*. https://mathib.github.io/fog-ontology/

Carrol, J. J., & Pan, J. Z. (2006). *Xml schema datatypes in rdf and owl*. https://www.w3.org/TR/swbp-xsch-datatypes/#sec-xmls-dt

Linietsky, J., Manzur, A., & the Godot community. (2023). *Portal — documentation de godot engine (stable) en français*. https://docs.godotengine.org/fr/stable/classes/class_portal.html

Ontotext. (2022). *What is sparql?* https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/

Rasmussen, M. H., & Schlachter, A. (n.d.). *Ld-bim - bim meets linked data*. https://ld-bim.web.app/

Unity. (2023). *Manual: Memory in unity webgl*. https://docs.unity3d.com/2023.2/Documentation/Manual/webgl-memory.html

W3C. (2015a). *Semantic web: Inference*. https://www.w3.org/standards/semanticweb/inference

W3C. (2015b). *Semantic web: Linked data*. https://www.w3.org/standards/semanticweb/data

W3C. (2015c). *Semantic web: Query*. https://www.w3.org/standards/semanticweb/query

W3C. (2023). *Linked building data community group*. https://www.w3.org/community/lbd/