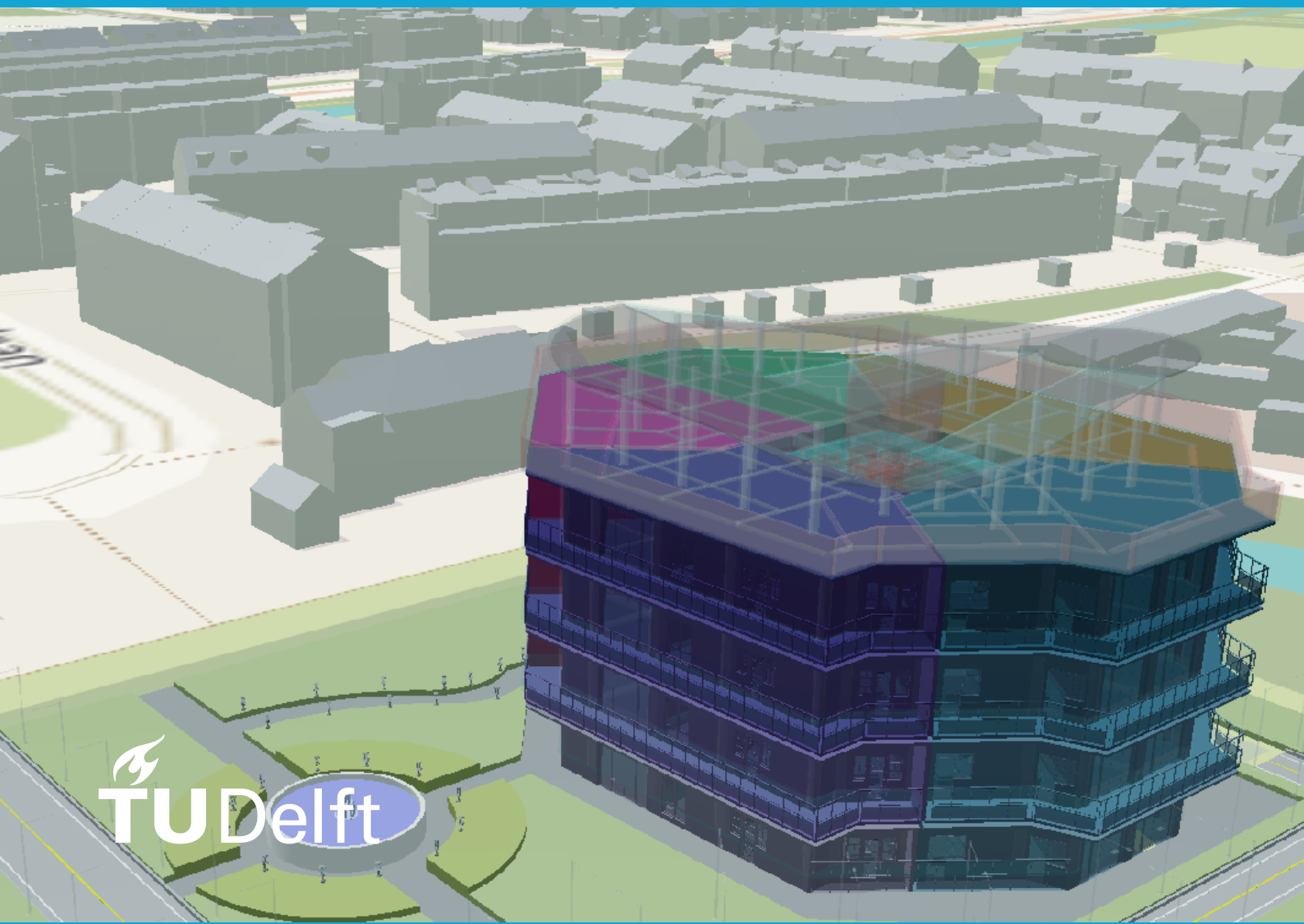
MSc thesis in Geomatics

A digital twin based on Land Administration

Ping Mao

2024



MSc thesis in Geomatics

A digital twin based on Land Administration

Ping Mao

July 2024

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

Geo-Database Management Centre
Delft University of Technology

Supervisors:	Prof. dr.ir. P.J.M. van Oosterom
	Dr. A. Rafiee
Co-reader:	Prof. dr.ir. C.Lemmen

Abstract

As urban architectural environments become increasingly complex and densely populated, the demand for precise registration of legal statuses, encompassing both private and public interests, has become more urgent. Traditional 2D cadastral registration systems are increasingly inadequate for addressing the multifaceted and vertical nature of modern urban landscapes. These systems are limited in scope and unable to fully capture the intricacies of multi-level property rights, overlapping parcels, and underground constructions.

This study uses a BIM/IFC model for the building's physical representation. The party and Rights, Restrictions, and Responsibilities (RRRs) data are stored in a DBMS, with all data and the building location being fictitious. Visualization is achieved in 3D over the web using Cesium JS, an extensible globe viewer.

Unlike earlier 3D cadastral systems, this research has developed a new 3D Land Administration prototype based on ISO 19152-2 (Land Administration Domain Model (LADM)). The objective is to explore improved methods for analyzing and visualizing RRRs in complex buildings. Novel techniques include presenting UML instance-level LADM diagrams for selected parties and/or apartments, showing RRRs and BAUnits linking them. The study introduces a new method for displaying surrounding buildings at varying Level of Detail (LoD), with closer buildings rendered in higher detail and more distant buildings shown in less detail. This selective detailing enhances both performance and clarity in visualizations.

A key feature of this digital twin system is its real-time update capability. This study supports the updating of party and rights information in the backend database, accurately reflecting these updates in the public front-end version. This ensures the maintenance and visualization of the most current property rights data. The system also integrates sunlight simulation, which is crucial for urban planning, architectural design, and aiding buyer decision-making.

For more details, visit the project's website 3D web-visualization platform and GitHub.

Acknowledgements

I would like to express my heartfelt gratitude to my advisors, Pater and Azara. From identifying the initial research direction to every meeting we had, your continuous and patient guidance ensured that I stayed on the right academic path. Your suggestions have constantly refined and improved my research. Your unwavering support and encouragement have given me confidence throughout this journey.

I am deeply thankful to my family; words cannot fully express my gratitude. Thank you for all the emotional and financial support over these two years. Your love, understanding, and unconditional support have given me the courage to keep moving forward and have helped me navigate all the ups and downs of this journey.

I also want to extend my sincere thanks to my friends. Thank you for exploring many cities and landscapes with me over these two years, for being there to help me fend off negative emotions during my low points, for all the practical advice during my thesis process, and for all the assistance during my academic journey. Your companionship and support have been invaluable.

Contents

Abstract v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Potential use cases . . . 2
- 1.3 Research questions . . . 2
- 1.4 Thesis Outline . . . 3

2 Related work 5
- 2.1 Land administration systems . . . 5
- 2.2 Previous 3D Land Administration System . . . 6

3 Methodology 13
- 3.1 Land Administration Domain Model . . . 13
  - 3.1.1 Party Package . . . 13
  - 3.1.2 Administrative Package . . . 14
  - 3.1.3 Spatial Unit Package . . . 14
- 3.2 Digital Twin . . . 14
- 3.3 Requirements of 3D Land administration systems (LAS) . . . 15
- 3.4 Virtual Globes . . . 16
- 3.5 System Integration and Implementation . . . 17

4 System design 19
- 4.1 Data Collection . . . 19
- 4.2 Model Origin Adjustment . . . 21
- 4.3 Format Conversion . . . 21
- 4.4 Scene Initialization . . . 22
  - 4.4.1 Initializing the Cesium Viewer . . . 22
  - 4.4.2 Loading Models in glTF Format . . . 22
- 4.5 Adaptive LoD . . . 24
- 4.6 Floor Visibility Control . . . 25
- 4.7 Sunlight simulation . . . 27
- 4.8 Dynamic Slicing View . . . 29
- 4.9 Visualizing Underground Spaces . . . 30
- 4.10 LADM chart visualization . . . 31
  - 4.10.1 Code list value description . . . 32
  - 4.10.2 Database Preparation . . . 35
  - 4.10.3 Frontend-Backend Data Interaction . . . 37
  - 4.10.4 Data Transformation and Classification . . . 38
  - 4.10.5 LADM Chart Display . . . 39
  - 4.10.6 Click Query Process . . . 39
  - 4.10.7 Search Query Process . . . 40

4.10.8 Convert Data into Graph Format 42

5 Results **45**
5.1 Loading 3D BAG and BIM Models 45
5.2 Interactivity 46
5.3 Floor Visibility Control 47
5.4 Sunlight Simulation 47
5.5 Dynamic Slicing View 48
5.6 Visualizing underground space Functionality 49
5.7 Dynamic Data Display 50
5.7.1 Personal Property Inquiry 51
5.7.2 Search and Information Display 52
5.8 Usability testing 54
5.8.1 User Groups 55
5.8.2 Questionnaire Design 55
5.8.3 Test Results 56
5.9 Reflection on system 56

6 Conclusion and Future Work **59**
6.1 Research overview 59
6.2 Future work 62

List of Figures

2.1 Dutch cadastral map [Hagemans et al., 2022] . . . 6
2.2 Legal volumes of the Delft Station case in 3D PDF [Stoter et al., 2016] . . . 7
2.3 Legal volumes of the Amsterdam case in 3D PDF [Stoter et al., 2017] . . . 7
2.4 Visualization of the urban area using Cesium JS . . . 8
2.5 3D LAS system interface showing physical and legal models . . . 9
2.6 Comparison between 2D plan (a) and 3D BIM representation (b) of ownership boundaries for units 203 and 204 [Atazadeh et al., 2017] . . . 10
2.7 Platform User Interface [Andritsou et al., 2023] . . . 10
2.8 Visualization of legal information based on LADM in a 3D platform . . . 11
2.9 Mapping of private, common, and structural ownership spaces within Building Information Modeling (BIM)/ Industry Foundation Classes (IFC) to LADM [Alattas et al., 2021] . . . 11

3.1 The four core classes of LADM [Lemmen, 2012] . . . 13
3.2 Requirements of 3D Land Administration System [Shojaei et al., 2013] . . . 16
3.3 Cesium web globe . . . 17

4.1 Web-Based 3D Visualization Architecture . . . 19
4.2 3D Bag Data Download . . . 20
4.3 Apartment Model . . . 20
4.4 Apartment with Surrounding Information Model . . . 20
4.5 Adjusted Origin Point in Blender . . . 21
4.6 Conversion Process . . . 21
4.7 Legal Space Model . . . 24
4.8 Adaptive Level of Detail (LoD) Visualization in the 3D LAS System . . . 25
4.9 Comparison of lighting effects at night . . . 27
4.10 Sunlight simulation effects of seasonal changes . . . 28
4.11 Visualization of the clipping plane. . . . 29
4.12 Visualization of underground spaces . . . 30
4.13 Illustration of Monique's Ownership Rights in the LADM Model[Lemmen et al., 2022] . . . 31
4.14 LADM Information of Jack . . . 32
4.15 LADM Information of James and Bella . . . 32
4.16 LA_Party . . . 35
4.17 LA_RRR . . . 36
4.18 LA_BAUnit . . . 36
4.19 LA_SpatialUnit . . . 37

5.1 3D physical model . . . 45
5.2 3D legal model . . . 46
5.3 surrounding environment . . . 46
5.4 Floor Visibility Control . . . 47

5.5 Sunlight Simulation 48
5.6 Cross Section View 48
5.7 Overlay effect showing selected floor 49
5.8 Visualization of underground spaces 50
5.9 Property information 51
5.10 click the ”check” button 51
5.11 clicks the ”Fly to” button 52
5.12 Search by Association of owners 53
5.13 Search by Tenant Name 53
5.14 Search by Owner Name 54
5.15 Search by Apartment ID 54
5.16 Pie chart of three types of participants 56

List of Tables

4.1 Parameters for the `addGLTF` function. 23
4.2 Functions, methods, and events used for controlling floor visibility. 27
4.3 Code List for LA_Party 33
4.4 Code List for LA_RRR 34

Acronyms

LADM Land Administration Domain Model v
LAS Land administration systems ix
RRRs Rights, Restrictions, and Responsibilities v
DT Digital Twin 14
LoD Level of Detail v
BIM Building Information Modeling xi
IFC Industry Foundation Classes xi

1 Introduction

This chapter begins with a brief discussion of the motivation behind this graduation project and then explores the potential use cases of the proposed 3D LAS. The main research questions are defined, including how to enhance the visualization and understanding of 3D LAS. Finally, this chapter provides a general overview of the thesis, outlining its structure and the key contributions of each chapter.

1.1 Motivation

The rapid growth of urban populations and the corresponding expansion of city infrastructures have triggered a fundamental reevaluation of spatial utilization strategies in contemporary urban environments. This growth often pushes cities to expand not only horizontally but also vertically into the sky and downward into the earth. Such multidimensional expansion has become crucial in densely populated urban centers, where the availability of land for housing and infrastructure is severely limited. This limitation naturally drives the development of multi-story and vertical buildings to maximize the efficient use of space.

This study is motivated by the critical need for efficient and transparent management of urban land, particularly in vertical developments. Traditional land management systems, which are largely two-dimensional, are increasingly challenged to manage the complexity inherent in vertically integrated property rights. The representation of parcels in 2D maps of multi-layered overlapping properties is handled by dividing the map into multiple parcels. Each parcel illustrates various rights. This method may be straightforward for those involved in property registration, but it can be difficult for those unfamiliar with the actual scenario to understand.

Against this backdrop, this study is dedicated to exploring and implementing an innovative land management solution focused on the visualization of apartment rights. A prototype 3D LAS is developed that not only integrates the physical and legal spatial models of multi-story apartments but also dynamically displays LADM data on a web interface. This enables users to intuitively query apartment units and their associated legal information, such as ownership, usage restrictions, and responsibilities. Moreover, the system allows various stakeholders—including real estate developers, governmental agencies, buyers, and sellers—to access and update information, thereby ensuring the timeliness and accuracy of the data.

By integrating modern GIS technology with land management standards, this study's information system prototype not only enhances the transparency of land management but also deepens users' understanding of apartment rights and the LADMframework. Furthermore, it provides a valuable reference and foundation for the development of future technologies and policies related to this field.

1.2 Potential use cases

Multi-Level Property Management: This system can be particularly useful in large multi-story residential and commercial complexes where different entities own different parts of the building. Property managers and homeowner associations can use the system to clearly define and manage property boundaries, rights, and responsibilities.

Real Estate Transactions: The system can streamline real estate transactions by providing potential buyers and real estate professionals with clear, detailed visualizations of property rights and restrictions. This reduces ambiguity and risk associated with property investments, especially in complex environments like skyscrapers or mixed-use developments.

Legal Dispute Resolution: In cases of property disputes, particularly in vertical real estate developments, the system can provide a clear and authoritative reference that can help resolve conflicts over boundaries and rights more efficiently. By providing a detailed and agreed-upon representation of property limits and rights, the system can reduce the frequency and severity of legal disputes.

Government Record Keeping and Taxation: Government agencies responsible for land registry, taxation, and infrastructure development can use this system to maintain accurate and up-to-date records of land ownership and usage. This would enhance the efficiency of property taxation and public service delivery.

Disaster Management and Response: In the event of a disaster, the system can assist emergency responders by providing them with detailed layouts of buildings, including access points and ownership details. This can be crucial for efficiently navigating complex buildings in emergency situations.

By addressing these use cases, the 3D LAS can significantly contribute to more efficient, transparent, and responsible land management and urban development.

1.3 Research questions

How does the integration of a Digital Twin, encompassing both real-time legal data of 3D property rights and surrounding environmental factors, enhance the visualization and understanding of 3D LAS complexities?

Sub Questions:

1. Why is the inclusion of a building's surrounding environment critical when visualizing 3D LAS data?

2. What are the essential requirements for a 3D web viewer system that aims to integrate detailed land administration data?

3. How does the system integrate the LADM information from the database with the 3D models, enabling users to query corresponding model information?

1.4 Thesis Outline

Chapter 1 provides an overview of the research background, motivation, and objectives, potential use cases and research questions.

Chapter 2 reviews existing literature and systems in the field, setting the context for the contributions of the current study. It shows how this study builds on and differs from previous work.

Chapter 3 primarily discusses the legal framework and methodology for visualizing apartment rights within a 3D land administration system.

Chapter 4 provides more detailed information on the research of this 3D LAS prototype. It details the use of the Vue framework and Cesium for front-end development, with PostgreSQL database for backend operations.

Chapter 5 describes the result and reflection of the system.

The final Chapter 6 answers the research questions posed in the introduction and outlines potential directions for future research.

2 Related work

In this chapter, a detailed examination of the existing literature and research on land administration systems is presented, with a particular focus on 3D LAS. The fundamental concepts and components ofLAS and their significance in land management, legal registration, and sustainable development are discussed. Various studies that have advanced the field of 3D LAS. This includes the integration of BIM and IFC with the LADM to enhance the representation and management of legal boundaries within complex urban environments.

2.1 Land administration systems

A land administration is a modern, parcel-based land information system that maintains records of land interests, including RRRs. Typically, it comprises a geometric description of land parcels, linked to additional records that explain the nature, ownership, or control of these interests, as well as the value of the parcels and any improvements. It can be used for fiscal purposes, such as valuation and fair taxation, for legal purposes like property conveyancing, to aid in land and land-use management, including planning and administrative tasks, and to promote sustainable development and environmental protection [International Federation of Surveyors (FIG), 1998].

The fundamental entities in cadastral registration are ”real estate”, ”immovable property” or ”property”, and ”subjects” [Stoter, 2004]. Typically, land and the buildings on it are called real estate. The various rights associated with the land are termed real property [International Federation of Surveyors (FIG), 1998].

The LAS aims to provide a foundation for land management by integrating spatial, legal, and administrative data. Notably, the term ”3D Land Administration” is increasingly favored over ”3D Cadastre” because it encompasses the entire scope of land registration and cadastral work [Kalogianni et al., 2020].The adoption of 3D LAS is driven by the fact that a 2D representation of ownership in complex apartments may be insufficient to fully describe and register these rights [Atazadeh et al., 2017].

The Netherlands’ Cadastre, Land Registry, and Mapping Agency, known as ”Kadaster,” consists of land registration and cadastre, providing legal certainty for every parcel of land and water in the Netherlands. The cadastral map visualizes the relative locations and shapes of all cadastral parcels in the Netherlands. It covers the entire country seamlessly, with 100% land coverage. The geometry of cadastral boundaries is initially recorded in field survey sketches and then processed into a national map. However, the graphical quality of the current cadastral map has limitations due to variations in working methods over time. In urban areas, the standard deviation of boundary points is approximately 20 cm, while in rural areas, it is about 40 cm [Hagemans et al., 2022].An excerpt of the Dutch cadastral map is shown as follows Figure 2.1 below.The cadastral map displays the boundaries of parcels,

parcel numbers, and major buildings. It also includes house numbers, street names, and waterway names.

Figure 2.1: Dutch cadastral map [Hagemans et al., 2022]

The Dutch LAS can register different types of RRRs. The first type is ownership (eigendomsrecht), which theoretically includes not only the land itself but also the space above and below the parcel. In addition to full ownership, there are various forms of limited ownership [Guler, 2022].

2.2 Previous 3D Land Administration System

The study by [Stoter et al., 2016], titled "First 3D Cadastral Registration of Multi-level Ownership Rights in the Netherlands," focuses on the integrated structure of the Delft city hall and underground railway station. An interactive 3D PDF was created, included in the deed, and registered in the land registry. This case demonstrates the practical benefits and feasibility of 3D cadastral registration, highlighting the advantages of 3D visualization for representing complex property rights, as shown in Figure 2.2.

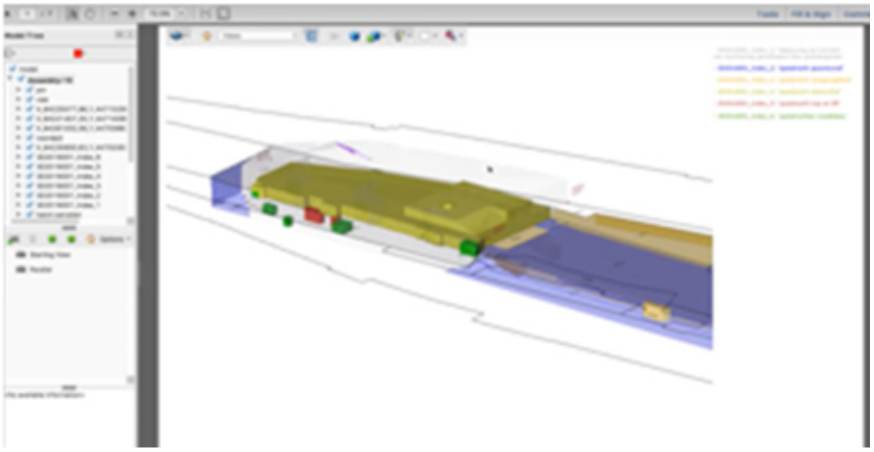


Figure 2.2: Legal volumes of the Delft Station case in 3D PDF [Stoter et al., 2016]

Another case by [Stoter et al., 2017] involves a building complex in Amsterdam. This project builds on the Delft case, improving the methodology and providing a general workflow from design data to legal documents. Architects created models representing legal spaces, which were used to generate the 3D PDF. Additional views of the building complex were generated, allowing users to switch easily between different perspectives within the 3D PDF. Exploded views clearly displayed different legal entities at various heights. Finally, data checks ensured that the legal volumes accurately represented stakeholders' legitimate interests, as shown in Figure 2.3.

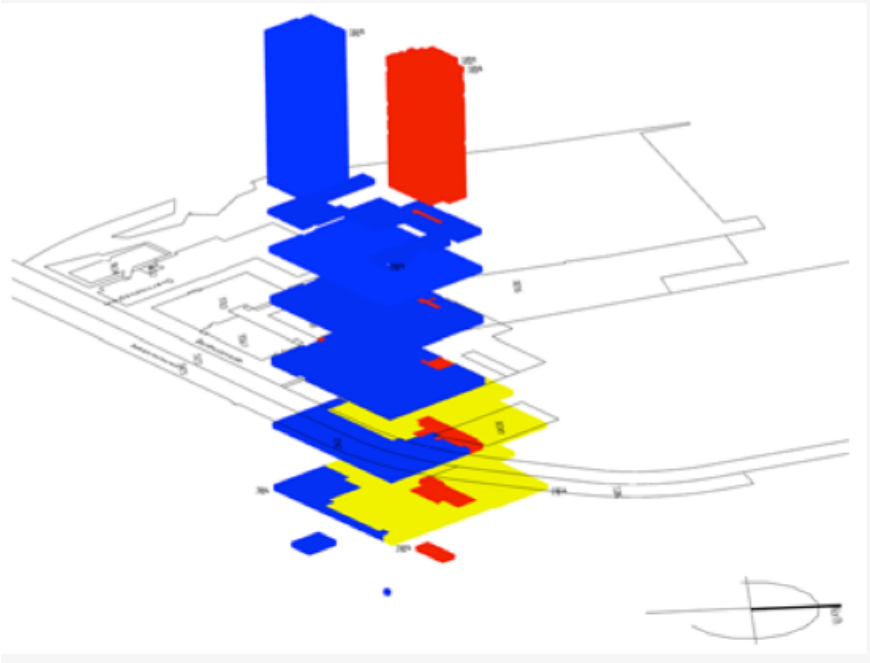


Figure 2.3: Legal volumes of the Amsterdam case in 3D PDF [Stoter et al., 2017]

The research [Cemellini et al., 2020] presents a 3D LAS prototype that significantly enhances the visualization and management of 3D cadastral parcels and utilizes Cesium JS to visualize 2D and 3D data in a 3D format, aiming to better represent legal boundaries and visualize underground spaces. However, users encountered positioning errors when selecting high-rise building units with the mouse. The 3D model, not being based on IFC data, lacked sufficient detail and did not include the surrounding environment of the studied buildings. The visualization of the urban area is shown in Figure 2.4.This study focuses on usability testing and continuous improvement based on user feedback, demonstrating the effectiveness of a user-centered design approach.

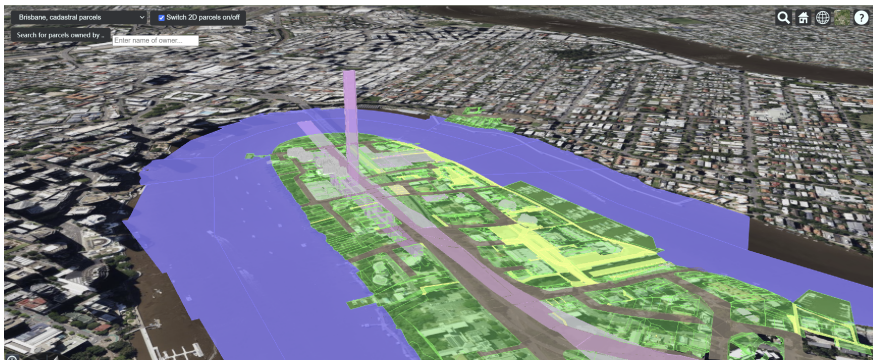


Figure 2.4: Visualization of the urban area using Cesium JS

The 3D LAS system described in OGDB (https://bpd2.ogdb.nl/bpd/project/9531/landgoed-hoevesteijn) provides both physical and legal models that can be displayed separately. This system supports interactive functionality, allowing users to click on each apartment unit to retrieve specific information, as shown in Figure 2.5. Additionally, it includes sunlight simulation for individual buildings. However, it does not account for the shadowing effects of surrounding buildings and does not incorporate LADM.

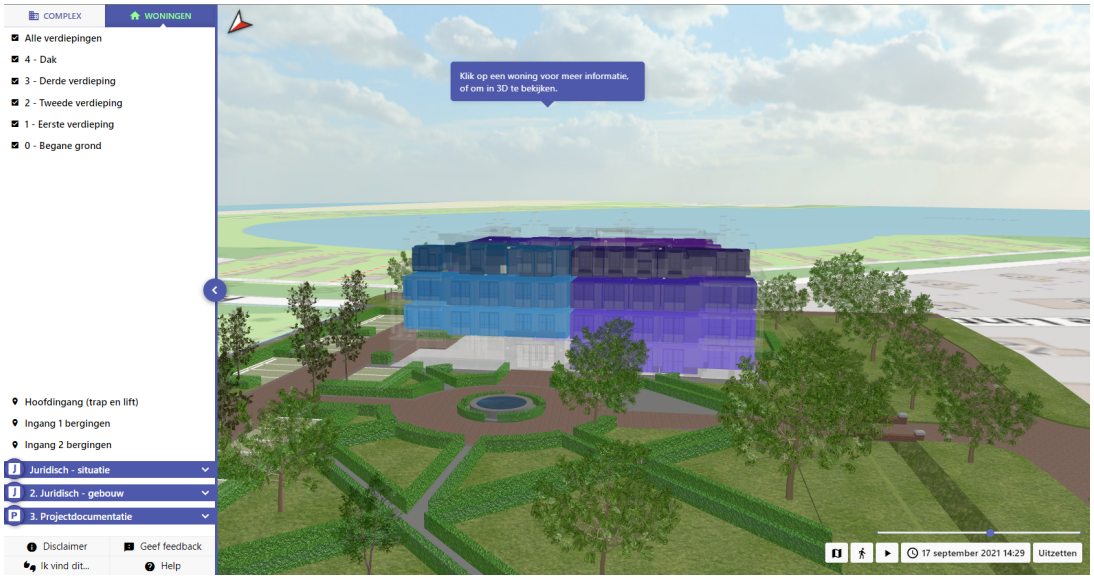


Figure 2.5: 3D LAS system interface showing physical and legal models

Another research [Atazadeh et al., 2017] demonstrates the capability of using BIM to represent internal ownership boundaries within buildings. The approach involves analyzing the existing data structures of IFC to identify and propose relevant entities suitable for modeling building ownership boundaries. These boundaries have been established within the Revit modeling environment. Figure 2.6 illustrates a comparison between traditional 2D plans and the 3D BIM representation of ownership boundaries for units 203 and 204.

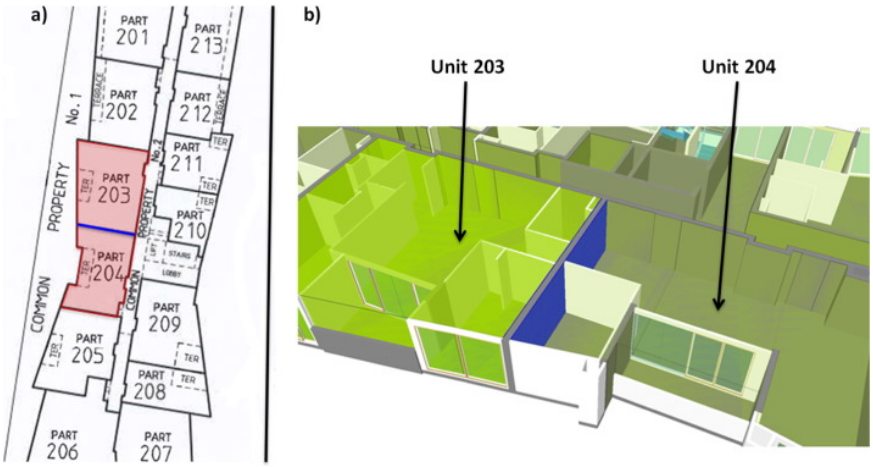


Figure 2.6: Comparison between 2D plan (a) and 3D BIM representation (b) of ownership boundaries for units 203 and 204 [Atazadeh et al., 2017]

The research [Andritsou et al., 2023] aims to create a detailed, low-cost, and interactive digital twin (DT) of a broader urban area by merging 3D land administration data with sustainable energy data. The platform integrates AI and energy consumption data to simulate time-based scenarios and provide actionable insights for energy management. This study also utilizes IFC models, focusing on monitoring energy consumption in buildings. The final user interface of the platform allows users to interact with various architectural, structural, and spatial components of each building and environmental information for each one of them, as shown in Figure 2.7.

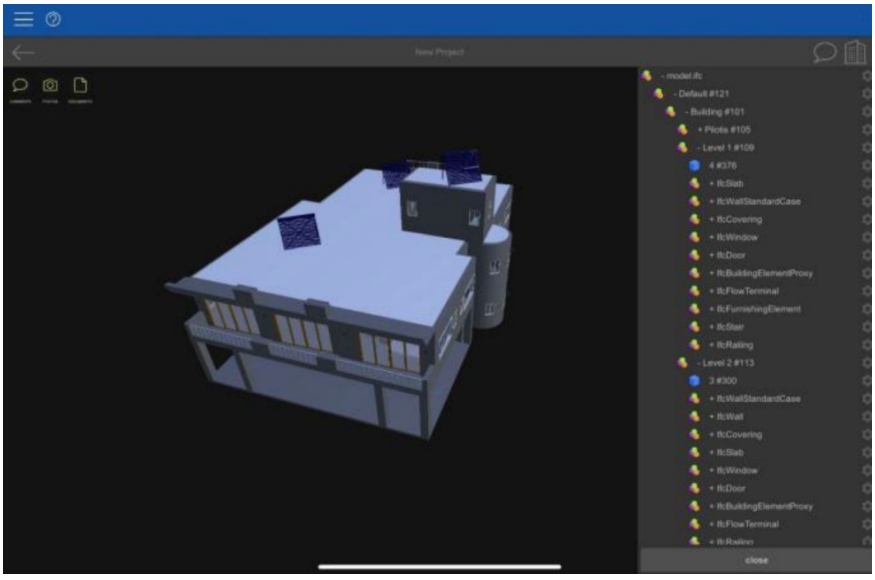


Figure 2.7: Platform User Interface [Andritsou et al., 2023]

This study [Broekhuizen et al., 2021] added legal information based on the LADM to IFC

models, stored it in a spatial database, and visualized it on a 3D platform. However, all the LADM information is provided to users in tabular form, as shown in Figure 2.8.

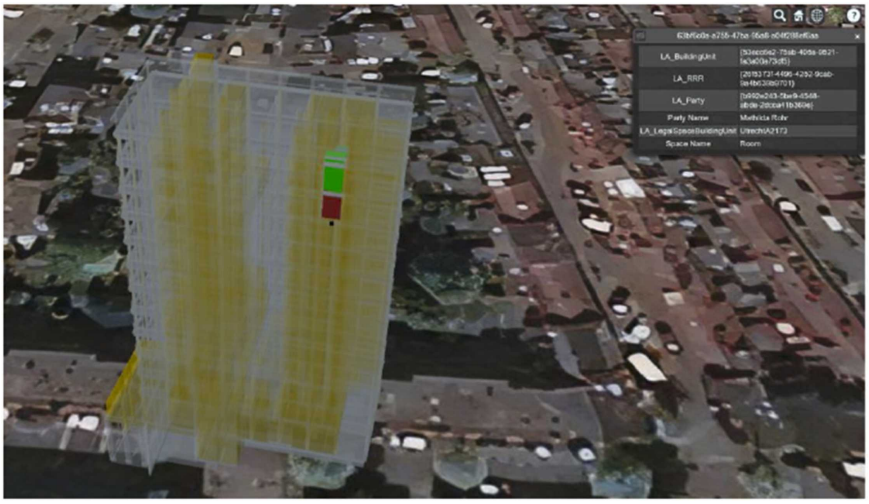


Figure 2.8: Visualization of legal information based on LADM in a 3D platform

Another research [Alattas et al., 2021] focuses on mapping BIM/ IFC data to LADM within the context of Saudi Arabia. This study not only identifies private and common spaces but also defines the ownership of structural elements within buildings. The methodology includes developing a 3D LADM-based country profile and mapping IFC models to this profile to accurately represent legal boundaries. However, the study highlights the need for higher quality models for more precise implementation. Figure 2.9 illustrates how private and common spaces, as well as structural elements, are defined within the BIM model, including the ownership of inner walls between different apartments.

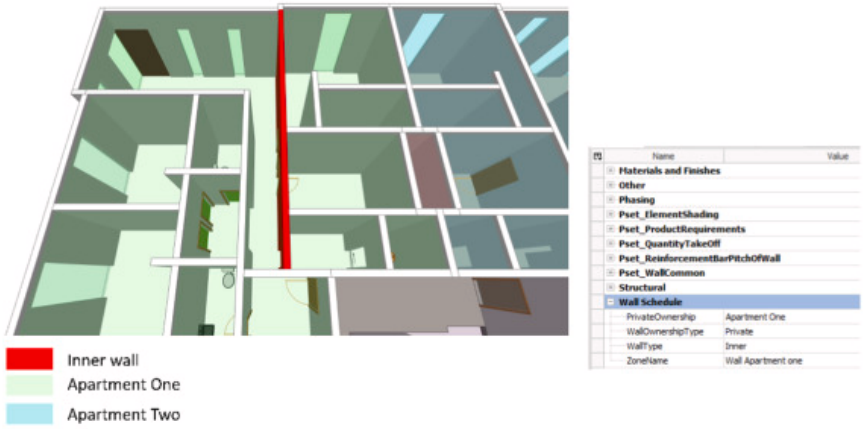


Figure 2.9: Mapping of private, common, and structural ownership spaces within BIM/ IFC to LADM [Alattas et al., 2021]

3 Methodology

This chapter explore the LADM, the concept of Digital Twins, the requirements for a 3D LAS, the use of virtual globes, and the integration and implementation process.

3.1 Land Administration Domain Model

The Land Administration Domain Model (LADM, ISO 19152) is a conceptual model focused on the legal and geographical aspects of land administration. It provides packages that describe the framework of land administration. The three main packages of LADM are the Party Package (green), the Legal/Administrative Package (yellow), and the Spatial Unit Package (blue) [Lemmen et al., 2015], as shown in Figure 3.1.

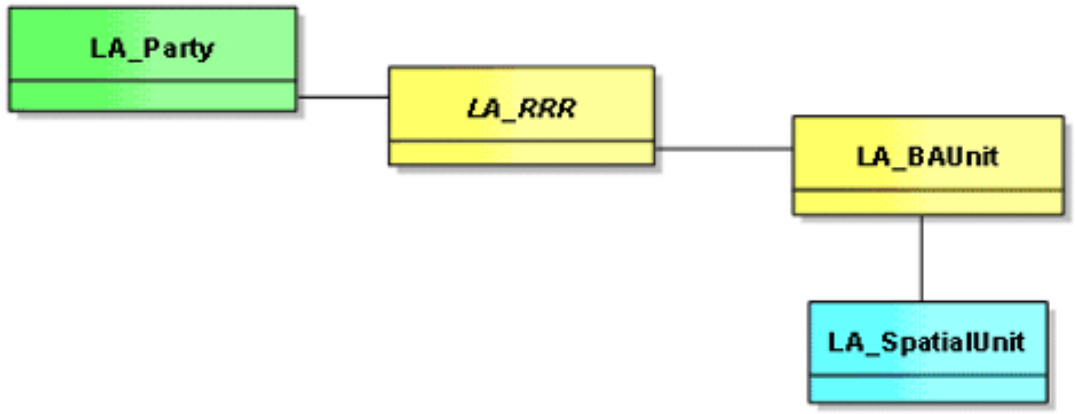


Figure 3.1: The four core classes of LADM [Lemmen, 2012]

3.1.1 Party Package

The Party Package defines the `LA_Party` class, representing the parties involved in land administration, such as individuals or organizations. The primary class in this package is `LA_Party`, which includes any person or organization participating in rights transactions. Organizations can range from companies and municipalities to state bodies and church communities.

3.1.2 Administrative Package

The Administrative Package includes RRRs and Basic Administrative Units (LA_BAUnit). LA_RRR consists of three concrete subclasses: LA_Right, LA_Restriction, and LA_Responsibility. A 'right' is a type of action, activity, or set of actions that participants in the system may perform concerning the associated resource, such as ownership, lease, or informal rights. A 'restriction' is a condition, whether state-based or non-state-based, that mandates refraining from certain actions. A 'responsibility' is a duty, either formal or informal, to perform certain actions.

The 'BA Unit' (Basic Administrative Unit) is an administrative entity comprising zero or more spatial units. It is associated with one or more unique and homogeneous RRRs within the land administration system. For instance, a 'BA Unit' might be a basic property unit that includes two spatial units, such as an apartment and a parking space.

3.1.3 Spatial Unit Package

The LA_SpatialUnit class is part of the Spatial Unit Package and manages the concept of physical entities, which can represent buildings, parcels, etc. The relationship between the LA_BAUnit class and the LA_SpatialUnit class allows for the association of the 2D/3D legal entities in LA_BAUnit with the 2D/3D physical objects in LA_SpatialUnit, thereby integrating legal concepts with physical realities.

3.2 Digital Twin

The term Digital Twin (DT) was first used by the National Aeronautics and Space Administration (NASA) in the technology domains: modeling, simulation, information technology, and integrated technology roadmap. NASA's Apollo program introduced the concept of "twins," two identical spacecraft built to mirror the condition of the spacecraft for the duration of the mission [Shafto et al., 2010; Boschert and Rosen, 2016].

In the recent decade, DT technology has significantly progressed and found applications in various sectors, including manufacturing, healthcare, smart cities, and construction. A digital twin is essentially a digital replica of a physical entity, system, or process, created to provide a dynamic and real-time representation. This model relies on data from sensors, IoT devices, and other data sources to accurately reflect the state and behavior of its physical counterpart [Fuller et al., 2020].

According to [Qi and Tao, 2018], digital twins facilitate the integration of Building Information Modeling (BIM), Geographic Information Systems (GIS), and the Internet of Things (IoT) to offer a comprehensive view of building projects. This integration aids in making informed decisions, improving efficiency, and reducing costs.

Digital twins represent real-world scenarios in a digital manner. The most advanced method for urban digitization appears to start with 3D city information models [Schrotter and Hürzeler, 2020]. These models primarily aim to describe the state of the urban physical environment.At the same time, there is a widespread consensus that digital twins are not merely equivalent to 3D city models. They also include other attributes, such as the lifecycle management of individual urban objects and assets [Lehtola et al., 2022].

This study presents a sophisticated 3D system that accurately replicates the physical structure of an apartment building and its individual rooms. The system allows users to interactively select any apartment to access real-time updates of basic apartment information and LADM information, ensuring that legal data is dynamically synchronized with the physical world.

A key feature of this system is its advanced sunlight simulation capability. It not only follows the system's current time but also enables users to simulate sunlight and shadow effects for any specified time. This functionality supports architects in optimizing their designs and aids potential buyers in assessing the lighting conditions of rooms. By providing dynamic visualization and real-time updates, this 3D system significantly enhances decision-making processes, showcasing the practical applications of Digital Twin technology, such as real-time data integration, predictive analysis, and effective decision support.

3.3 Requirements of 3D LAS

A 3D Land Administration Visualization system is structured into three distinct layers to optimize performance and user interaction as shown in the Figure 3.2. The Presentation Layer, serving as the system's front-end, interacts directly with users. It encompasses land administration features that visually represent crucial data like property boundaries and ownership details, alongside visualization features equipped with dynamic scaling, rotation, and other interactive elements to enhance user engagement. The Application Layer includes non-functional attributes that bolster technological diversity, system interoperability, and overall usability. These features, while not directly tied to visualization, indirectly uplift the visual quality by ensuring system robustness and seamless integration with diverse technologies. This layer functions as a pivotal bridge between the user interface and the backend data management. The Data Access Layer marks the foundational tier, ensuring data accessibility and supporting format conversions. It links to spatial databases, flat files, and web services, crucial for the versatile and scalable management of data in various formats. Collectively, these layers form a comprehensive system that not only facilitates effective data visualization but also enhances system reliability and user experience in land administrationt [Shojaei et al., 2013].

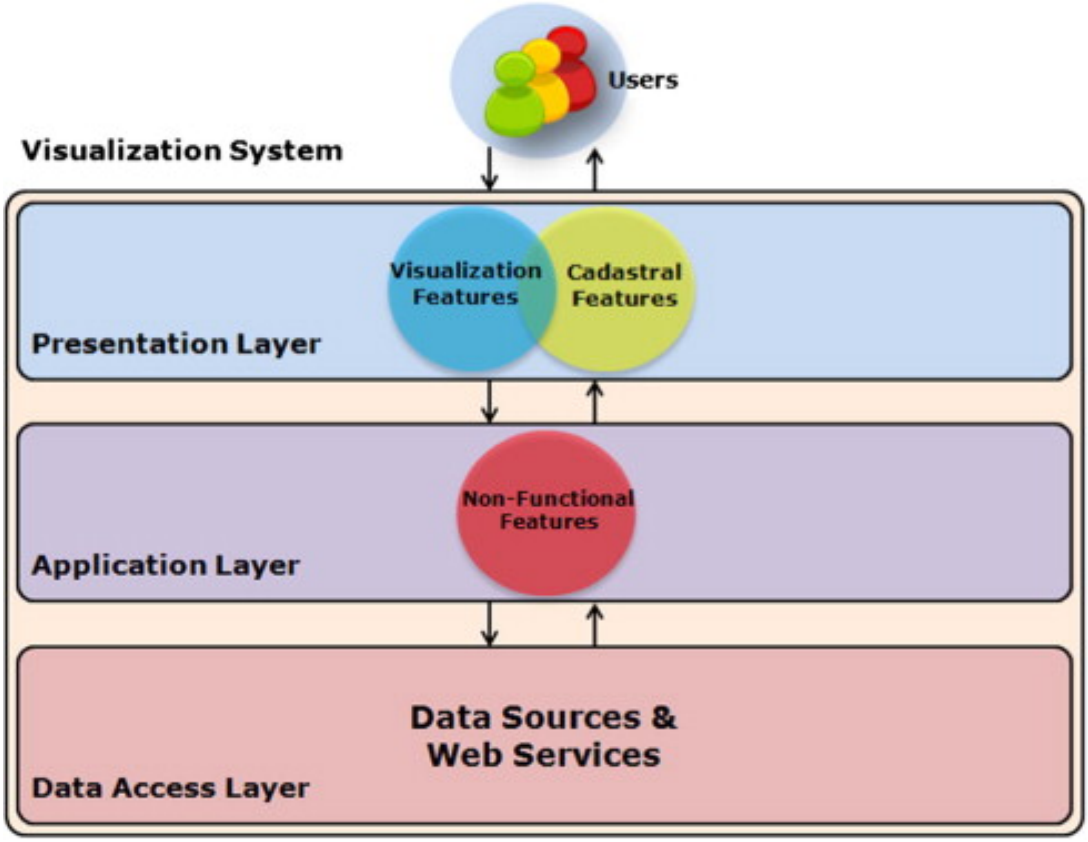


Figure 3.2: Requirements of 3D Land Administration System [Shojaei et al., 2013]

3.4 Virtual Globes

With the development of 3D web-based applications, virtual globes have emerged as a new medium for visualizing and interacting with geographic data. These tools allow users to freely navigate within a virtual environment, changing perspectives and positions as needed. Several WebGL-based virtual globes have been developed to facilitate cross-platform and cross-browser applications, including Cesium, OpenWebGlobe, and WebGLEarth [Keysers, 2015].

Among these, Cesium stands out as a notable virtual globe, as shown in the Figure 3.3. Cesium is an open-source JavaScript library that enables the creation of 3D virtual globes and 2D maps in web browsers. As open-source software under the Apache 2.0 license, Cesium can be used freely for both commercial and non-commercial purposes. Its key features include:

- **Unified API**: Provides a consistent API for 3D globes, 2D maps, and the Columbus view (2.5D).
- **Industry-standard Format Support**: Visualizes 3D models using formats such as KML, glTF (GL Transmission Format), GeoJSON, TopoJSON, and CZML (Cesium Language).

- **High-resolution Terrain Visualization**: Supports the visualization of high-resolution terrain data.
- **Layered Imagery**: Allows layering images from multiple sources, including WMS, TMS, WMTS, Bing Maps, Mapbox, Google Earth Enterprise, OpenStreetMap, ArcGIS MapServer, standard image files, and custom tiling schemes.
- **Extensive Geometry Library**: Includes a wide range of support for 2D and 3D geometries, enabling users to draw polylines, polygons, ellipsoids, spheres, labels, billboards, and sensors.
- **Interaction Handlers**: Provides handlers for mouse and keyboard events, camera movements, and zooming/panning of the virtual globe.

Figure 3.3: Cesium web globe

By leveraging these features, Cesium offers a robust and versatile platform for developing advanced geospatial applications.

3.5 System Integration and Implementation

The core challenge of this study lies in visualizing legal data on the frontend and accurately integrating legal models with physical models. The integration and implementation of the 3D LAS involved several key steps to achieve a functional and efficient system.

Data Standardization

To ensure seamless integration and interoperability, it was crucial to standardize both legal and spatial data. Legal data adhered to the ISO 19152:2012 standard, while spatial models

were formatted in IFC or OBJ. This standardization ensured that data from various sources could be effectively combined and visualized within the system.

User Interface Development

The user interface (UI) was developed using Vue.js and Cesium to provide an interactive and intuitive experience for users.

- **Component-Based Architecture**: Vue.js's component-based architecture was utilized to modularize the UI, making it easier to manage and extend.
- **3D Visualization with Cesium**: Cesium's capabilities were leveraged to render 3D models of buildings and their legal spaces.

Backend Development and Data Handling

The backend of the system was developed using Node.js, handling data requests and serving as the intermediary between the frontend and the PostgreSQL database. Key components of the backend included:

- **API Development**: RESTful APIs were created to enable the frontend to communicate with the database. These APIs handled queries for property information, legal data, and spatial data. For instance, a POST endpoint `/query/queryall` was defined to execute SQL queries received from the client and return the results.
- **Data Transformation**: Methods were implemented to transform and map database entries to the corresponding 3D models and UI components, ensuring accurate and intuitive data presentation.

Unit Testing

Thorough testing and validation were conducted to ensure the reliability and accuracy of the 3D LAS. This involved:

- Testing individual components and features to verify their correctness.
- Meetings to gather feedback and make necessary adjustments to improve usability and functionality.

4 System design

This research presents the development of a 3D mapping application that integrates Cesium within the Vue.js framework. By leveraging Vue's component-based architecture, various objects and methods from Cesium were effectively encapsulated. The development process involves several critical technical steps, as illustrated in Figure 4.1. The application enables the visualization of apartment building models, including their legal spaces, and allows users to access detailed ownership information for each apartment unit. The ownership information is structured based on the LADM, ensuring standardized property data representation.

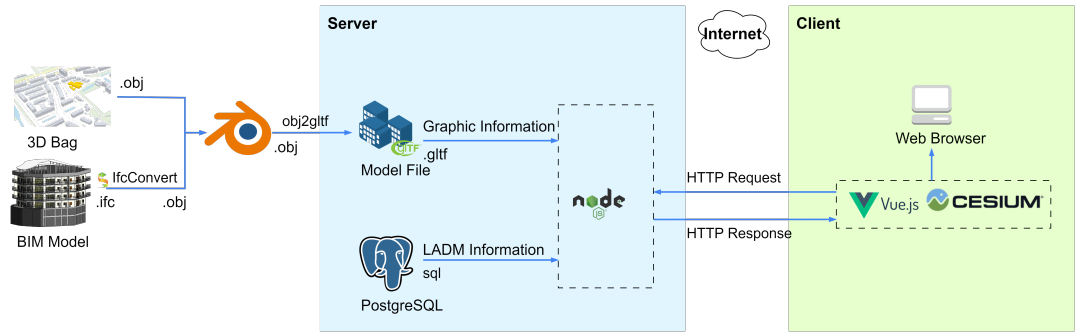


Figure 4.1: Web-Based 3D Visualization Architecture

4.1 Data Collection

The system utilizes a 3D BAG dataset provided in OBJ format, encompassing three levels of detail: LoD1.2, LoD1.3, and LoD2.2. This dataset is crucial for creating detailed representations of urban environments, with each level of detail offering different granularity suitable for various scales of visualization. The data acquisition process is illustrated in Figure 4.2.

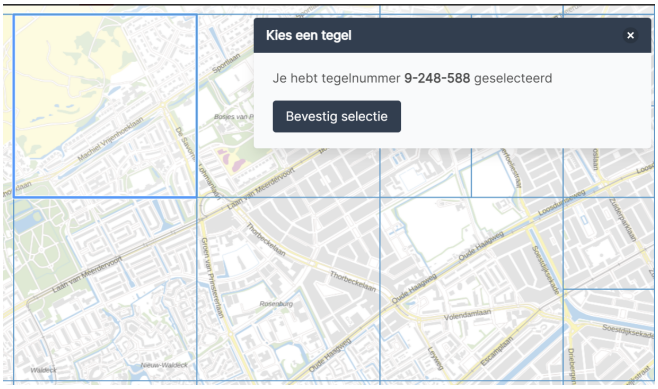


Figure 4.2: 3D Bag Data Download

Apartment models, typically used for sharing and exchanging construction and building data, are converted from the Industry Foundation Classes (IFC) format to the more universally compatible OBJ format. This study uses a model comprising 19 apartment units spread across four above-ground floors and two underground floors, as shown in Figure 4.3. Additionally, a parcel model representing the building's location has been prepared, as depicted in Figure 4.4.

Figure 4.3: Apartment Model

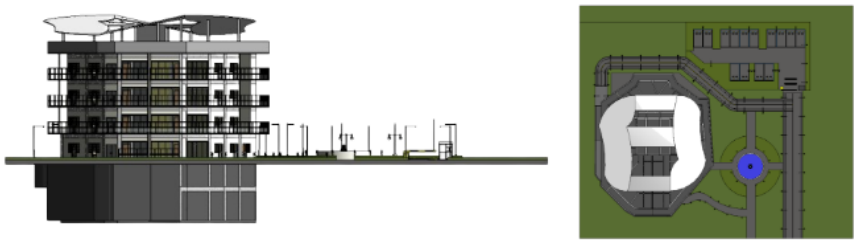


Figure 4.4: Apartment with Surrounding Information Model

4.2 Model Origin Adjustment

In Blender, the origin point of the models was adjusted to ensure accurate positioning within the 3D space using WGS84 coordinates. This adjustment is crucial for correctly placing the models within the virtual globe provided by Cesium, as shown in Figure 4.5.

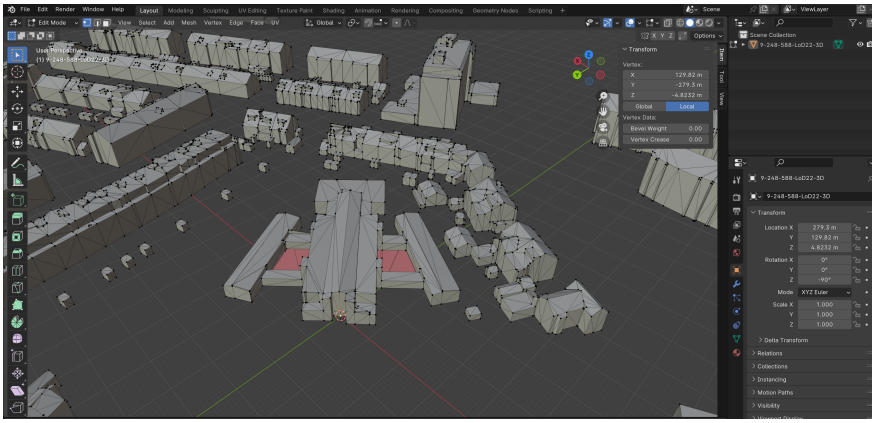


Figure 4.5: Adjusted Origin Point in Blender

4.3 Format Conversion

To integrate the models with Cesium, it was necessary to convert all model files into a format supported by Cesium. In this research, all models were converted to the glTF format. This step utilized the obj2gltf tool provided by Cesium and IfcConvert from IfcOpenShell. The operational workflow for this conversion process is illustrated in Figure 4.6.

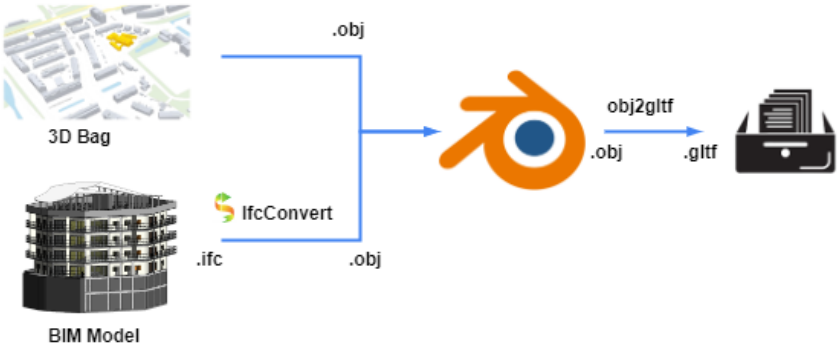


Figure 4.6: Conversion Process

4.4 Scene Initialization

After preparing the models and converting them to the glTF format, the next step is to initialize the scene on the front end. The initialization process includes configuring the Cesium viewer, loading the glTF models, and ensuring they are correctly positioned within the virtual environment.

4.4.1 Initializing the Cesium Viewer

To begin with, i set the Cesium Ion access token, which is essential for accessing Cesium's library of 3D tilesets, imagery, and terrain data. Several features are enabled, including fullscreen functionality (fullscreenButton) and timeline control (timeline). The scene mode is set to 3D only (scene3DOnly) to ensure that users operate within a fully three-dimensional space.

Additionally, I define the initial camera position, and configure translation, rotation, and zoom functionalities. This involves setting the camera's destination and orientation, which specify the coordinates and direction to which the camera should move.

Here is the code that sets up the initial camera view:

```
const HomeCamera = {
  destination: new Cesium.Cartesian3.fromDegrees(
    AppConfig.cameraP.lng,
    AppConfig.cameraP.lat,
    AppConfig.cameraP.h
  ),
  orientation: {
    heading: AppConfig.cameraP.heading,
    pitch: AppConfig.cameraP.pitch,
    roll: AppConfig.cameraP.roll,
  }
};
```

In the above code:

- `destination` sets the target position of the camera using geographic coordinates (longitude, latitude, and height) converted to Cartesian3 coordinates.
- `orientation` defines the camera's direction, including the heading (horizontal rotation), pitch (vertical rotation), and roll (tilt).

4.4.2 Loading Models in glTF Format

After initializing the Cesium Viewer, the next step involves adding models in glTF format to the scene. The process of loading and positioning models accurately includes setting the model's coordinates, orientation, and appearance. This is achieved through a custom function (loadGLTF.js) that takes parameters such as the model identifier(id), the URL of the

glTF file(url), position, rotation angle, pickability, color blend mode, and custom shaders, as shown in the Table 4.1.

Parameter Name	Default Value	Possible Values	Description
`viewer`	N/A	Cesium Viewer instance	The Cesium Viewer instance where the model will be added.
`id_`	N/A	String	A unique identifier for the model.
`url`	N/A	String (URL)	The URL of the glTF file.
`position`	N/A	Array [longitude, latitude, height]	The geographic coordinates where the model will be placed.
`show_`	`true`	`true`, `false`	Determines if the model should be initially visible.
`rotationz`	N/A	Number (degrees)	The rotation angle around the Z-axis in degrees.
`scale`	`1`	Number	The scale factor for the model.
`allowPick`	`true`	`true`, `false`	Determines if the model can be picked (interacted with).
`BlendMode`	`"MIX"`	`"MIX"`, `"HIGHLIGHT"`, `"REPLACE"`	The color blend mode for the model.
`color_`	N/A	Cesium.Color instance	The color to be applied to the model.
`CS`	`undefined`	Custom Shader	An optional custom shader for the model.

Table 4.1: Parameters for the `addGLTF` function.

The visualization of legal space models is achieved through custom shaders, where each apartment unit is assigned a unique color. These colors are defined using the BuildingMaterial class, which is then applied during the addGLTF function call. For instance, a custom shader can be created with

```
new BuildingMaterial(new Cesium.Cartesian3(232/255, 137/255, 185/255)).
```

The BuildingMaterial class extends Cesium's CustomShader class. Within its constructor, a uniform variable named color_ is defined to pass the color value. In the fragment shader, the material.diffuse property is set to a color obtained by blending color_, while the material.alpha property is set to 0.5. This setup effectively controls the color and transparency of the 3D models, providing a clear and distinct visualization of each legal space, allowing users to clearly and intuitively distinguish and locate each apartment unit, as illustrated in Figure 4.7.

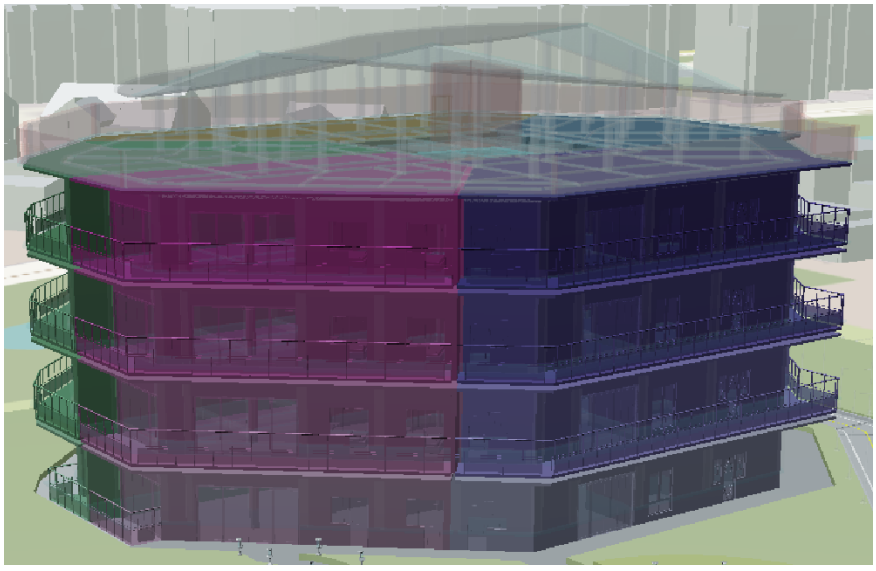


Figure 4.7: Legal Space Model

The main differences between adding physical models and legal space models lie in the additional parameters used for legal space models, specifically the use of custom shaders (BuildingMaterial) to define unique colors and transparency for each unit. This allows for a clear and distinct visualization of the legal spaces within the apartment building.

4.5 Adaptive LoD

The system includes a feature to visualize the surroundings of the central building using different LoD. This implementation allows for a more efficient and clear visualization experience by adjusting the complexity of the model based on the distance from the central building. In the Blender software, these modifications were applied according to the distances. Specifically, the system uses LoD2.2 for areas within 200 meters, LoD1.3 for areas between 200 to 300 meters, and LoD1.2 for areas beyond 300 meters. The system provides an "Adaptive LoD" button, allowing users to switch between different levels of detail as shown in Figure 4.8. This feature enhances the user's ability to explore the environment dynamically, focusing on high-detail visualization when close to the building and reducing detail when further away to improve performance and clarity.

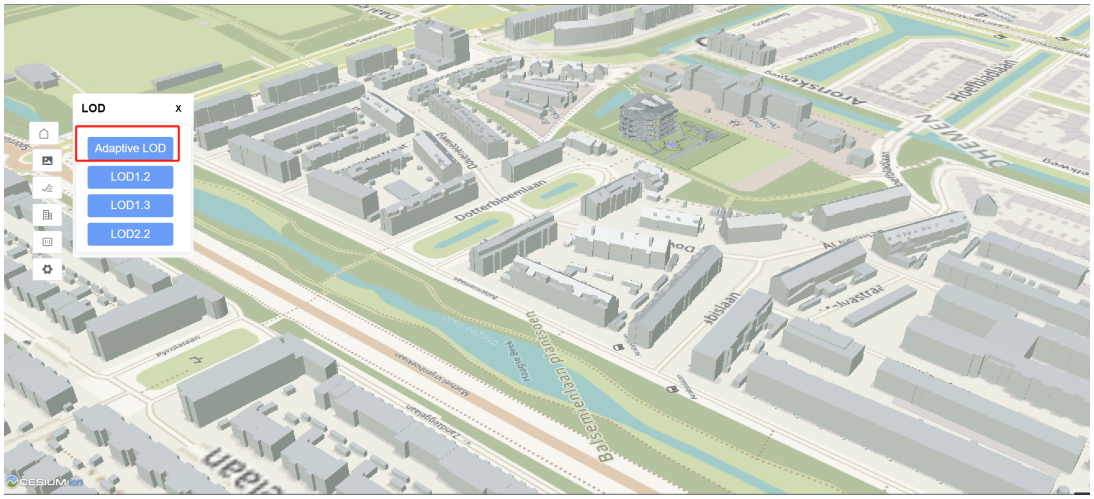


Figure 4.8: Adaptive Level of Detail (LoD) Visualization in the 3D LAS System

4.6 Floor Visibility Control

Floor visibility control is achieved through several steps and methods, involving the use of tree view components, enabling checkboxes, and corresponding events and methods to dynamically adjust the visibility of models.

1. **Create a Tree View using the Element UI `<el-tree>` Component**

   Bind the tree view to the `building_levels` data property, representing the hierarchical structure of the building.

```
building_levels: [{
    id: 1, key: 'ALL', label: 'ALL', children: [
        { id: 8, key: 'top', label: 'Roof' },
        { id: 7, key: '4', label: '3' },
        { id: 6, key: '3', label: '2' },
        { id: 5, key: '2', label: '1' },
        { id: 4, key: '1', label: '0' },
        { id: 3, key: 'B1', label: 'B1' },
        { id: 2, key: 'B2', label: 'B2' }
    ]
}]
```

2. **Enable Checkboxes**

   Enable checkboxes in the tree view by setting the `show-checkbox` attribute, allowing users to select multiple floors or rooms.

3. **Update Selected Nodes and Model Visibility**

   The `CheckFun` method is triggered when the selection state of a node changes, updating the `checkedNodes` property to reflect the current selection state.

```
CheckFun(data, checklist) {
    this.checkedNodes = checklist.checkedNodes; // Update selected nodes
}
```

The `CheckChange` method is triggered when the checkbox state changes, dynamically adjusting the visibility of models based on user selection.

- **BUILDING Mode:** Retrieve the corresponding floor model from `ModelListC` based on `data.key` and set its `show` property to `checked`.

- **LEGAL Mode:** Iterate through `ModelListF`, checking if the model's `key` matches the selected floor `key`, and set the `show` property based on the checkbox state.

```
CheckChange(data, checked) {
    if (this.show_controller) {
        if (data.id != 1) {
            if (this.activeName === this.show_mode_list[0]) { // BUILDING Mode
                ModelListC.get(data.key).show = checked;
            } else { // LEGAL Mode
                ModelListF.forEach((model, key) => {
                    if (key.split('-')[0] === data.key) {
                        model.show = checked;
                    }
                });
            }
        }
    }
}
```

4. **modeChangeFun Method**

   Controls the visibility of models based on the selected mode.

Floor visibility control is achieved by creating a tree view using the `el-tree` component, enabling checkboxes, updating the `checkedNodes` property, and using the `modeChangeFun` method to dynamically adjust the visibility of models based on the selected mode, as shown in Table 4.2. This structured approach ensures accurate display of models in different view modes, allowing users to view and manage different parts of the building as needed.

Name	Type	Description
`CheckFun`	Method	Updates `checkedNodes` to reflect the current selection state
`CheckChange`	Method	Adjusts model visibility based on the node's check status
`modeChangeFun`	Method	Controls model visibility based on the selected mode. Type -1: Show entire building; Type 0: Show floors; Type 1: Show rooms
`show-checkbox`	Property	Enables checkboxes for each node, allowing users to select multiple floors or rooms
`@check`	Event	Triggered when a checkbox is clicked, updates `checkedNodes`
`@check-change`	Event	Callback for when the node's selected state changes, adjusts model visibility
`building_levels`	Data Attribute	Array containing the tree view data structure defining the building hierarchy
`checkedNodes`	Data Attribute	Holds the currently selected nodes
`ModelListC`	Data Structure	Stores floor models
`ModelListF`	Data Structure	Stores room models
`ModelListAll`	Data Structure	Stores the entire building and other related models
`activeName`	Data Attribute	Holds the name of the currently active tab

Table 4.2: Functions, methods, and events used for controlling floor visibility.

4.7 Sunlight simulation

To achieve realistic sunlight simulation, I utilized Cesium's attribute viewer.scene.globe.enableLighting, which dynamically adjusts the scene's lighting based on the sun's position. This global lighting effect enhances the visual fidelity of the scene, making it responsive to real-world solar positions.

1. **Dynamic Lighting Control**

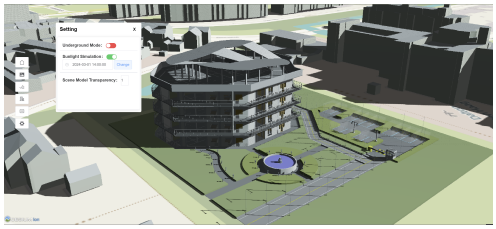
   I implemented a custom updateScene function to further control the lighting effects. This function calculates the dot product based on the positions of the sun and the camera to determine whether it is day or night. During nighttime, the function adjusts the lighting intensity accordingly. The comparison between scenes without and with the updateScene function is shown in Figure 4.9. The Left figure depicts the scene at 1 AM without the updateScene function, while right figure shows the same scene with the function applied, illustrating enhanced lighting realism.

Figure 4.9: Comparison of lighting effects at night

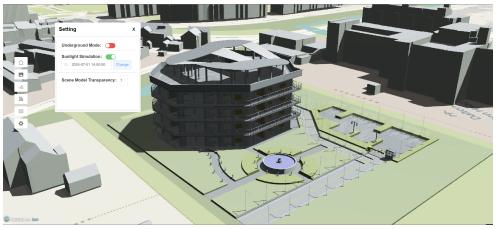
2. **Shadow Effects:** To enhance the scene's realism, I also enabled shadow effects. By setting viewer.shadows and viewer.shadowMap.enabled, activated shadow mapping.

3. **Real-Time Simulation and User Interaction:** Cesium automatically sets the scene's current time to the system time during initialization. Once the sunlight simulation feature is enabled, it automatically renders lighting effects and shadows, providing a dynamic and realistic environment. To further enhance functionality and user experience, I added a date and time selection feature. By using the el-date-picker component, users can select their desired date and time. The selected value is automatically bound to the timestr variable, converted to a Cesium.JulianDate object, and then set as the current time in the scene. I have set specific dates to illustrate different seasonal effects: March 1st for spring, July 1st for summer, October 1st for autumn, and January 1st for winter, each at 2 PM, as shown in the Figures4.10.

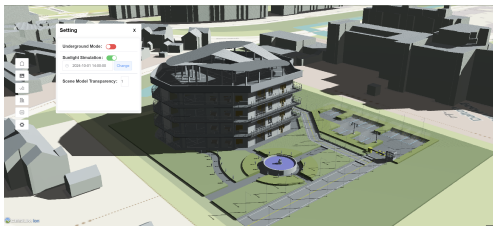
   This combination of real-time lighting adjustments, shadow effects, and user-interactive date and time selection creates a robust and immersive environment, beneficial for applications such as architectural design and real estate decision-making. Users can observe how different times of the day affect lighting and shadows in the scene, making informed decisions based on realistic simulations.



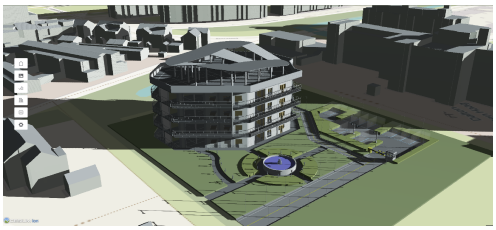
(a) Spring



(b) Summer



(c) Autumn



(d) Winter

Figure 4.10: Sunlight simulation effects of seasonal changes

4.8 Dynamic Slicing View

The development of this functionality was primarily inspired by the 3D Tiles Clipping Plane example from Cesium Sandcastle. By using `Cesium.ClippingPlane`, I define the clipping plane's normal vector (e.g., `Cesium.Cartesian3(1.0, 0.0, 0.0)`) and distance (e.g., `0.0`). For debugging purposes, I add a clipping plane entity to the scene through the `createClippingPlanesEntity` method, visualizing the material and color properties of the clipping plane, as shown in Figure 4.11.

Figure 4.11: Visualization of the clipping plane.

The `addGLTF` method is used to load GLTF models and apply the clipping plane collection (`clippingPlanes`) to the model. The following code demonstrates this process:

```
let model = viewer.scene.primitives.add(Cesium.Model.fromGltf({
  url: 'path/to/model.gltf',
  modelMatrix: modelMatrix,
  scale: 1.0,
  clippingPlanes: clippingPlanes
}));
```

Mouse events are bound using Cesium's `ScreenSpaceEventHandler` to listen for mouse wheel events that adjust the clipping plane's position. This allows users to interactively modify the clipping effect, ensuring that the clipping plane's position dynamically changes with the mouse wheel events.

During the rendering process, Cesium's internal rendering engine automatically calculates the distance from each vertex to the clipping plane, hiding the geometry behind the clipping plane to achieve the visual effect of clipping. This implementation allows users to view the internal structure of 3D models more intuitively and flexibly.

4.9 Visualizing Underground Spaces

The underground space visualization feature can be enabled by clicking the *Underground Mode* button. This feature is inspired by the globe translucency example in Cesium Sandcastle.

To allow the camera to move into the underground space, it is necessary to disable the camera's collision detection. This is achieved by setting the `enableCollisionDetection` property of the `screenSpaceCameraController` to `false`. Next, the transparency of the globe is configured to enable visualization of the underground space. This is done by setting the `frontFaceAlphaByDistance` property of the `globe.translucency` object. This property controls the transparency of the globe based on the distance from the camera. The code is as follows:

```
scene.screenSpaceCameraController.enableCollisionDetection = !checked;
globe.translucency.enabled = checked;
globe.translucency.frontFaceAlphaByDistance = new Cesium.NearFarScalar(
  400.0,  // Near distance
  0.0,    // Near transparency value
  800.0,  // Far distance
  1.0     // Far transparency value
);
```

To enhance the observation of underground structures, a *Transparency* button is provided. By calling the `Cesium.Color.fromAlpha` method, users can set the transparency of the models. The camera view can be adjusted using the `camera.flyTo` method. The effect is shown in the following figure 4.12.

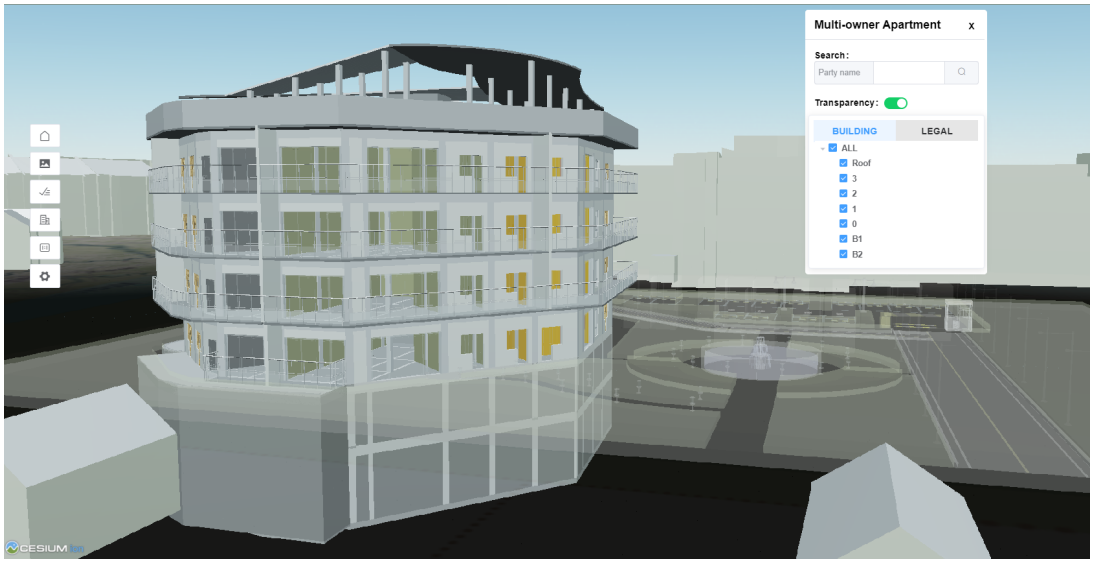


Figure 4.12: Visualization of underground spaces

4.10 LADM chart visualization

This study refers to the instance-level cases mentioned in (ISO 19152:2012 and the examples provided by [Lemmen et al., 2022]. Various legal scenarios are hypothesized and visualized based on these references, and these use cases by creating UML-instance level diagrams shown in 3D land administration system.

Figure4.13 provides a reader-friendly diagram that clearly illustrates the LADM model. Monique (LA_Party) holds ownership rights (LA_RRR) to three spatial units (LA_SpatialUnit): an apartment (WR18-4), a parking lot (WR18-5), and a laundry room (WR18-6). These units are grouped into Basic Administrative Unit 12 (LA_BAUnit).

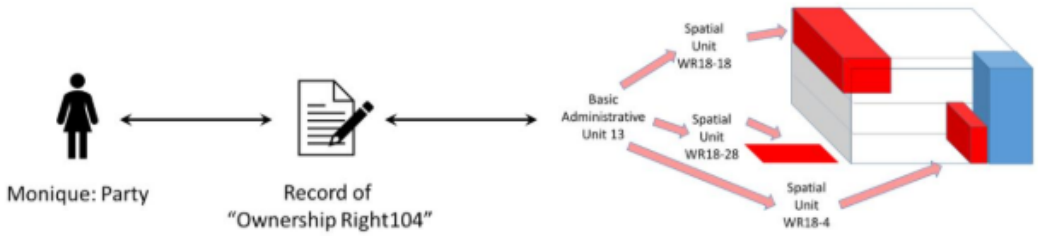


Figure 4.13: Illustration of Monique's Ownership Rights in the LADM Model[Lemmen et al., 2022]

Figure 4.14 is a UML instance-level diagram that illustrates a hypothetical case from this study. The homeowner, Jack, owns three apartments. He has rented these three apartments to Sophia, Mia, and Ella respectively. Figure 4.15 is another hypothetical case from this study. James and Bella have jointly purchased an apartment and a parking lot. I also assumes examples where the apartment owner took a loan from the bank during the purchase, and a transaction example involving both the previous and current apartment owners.

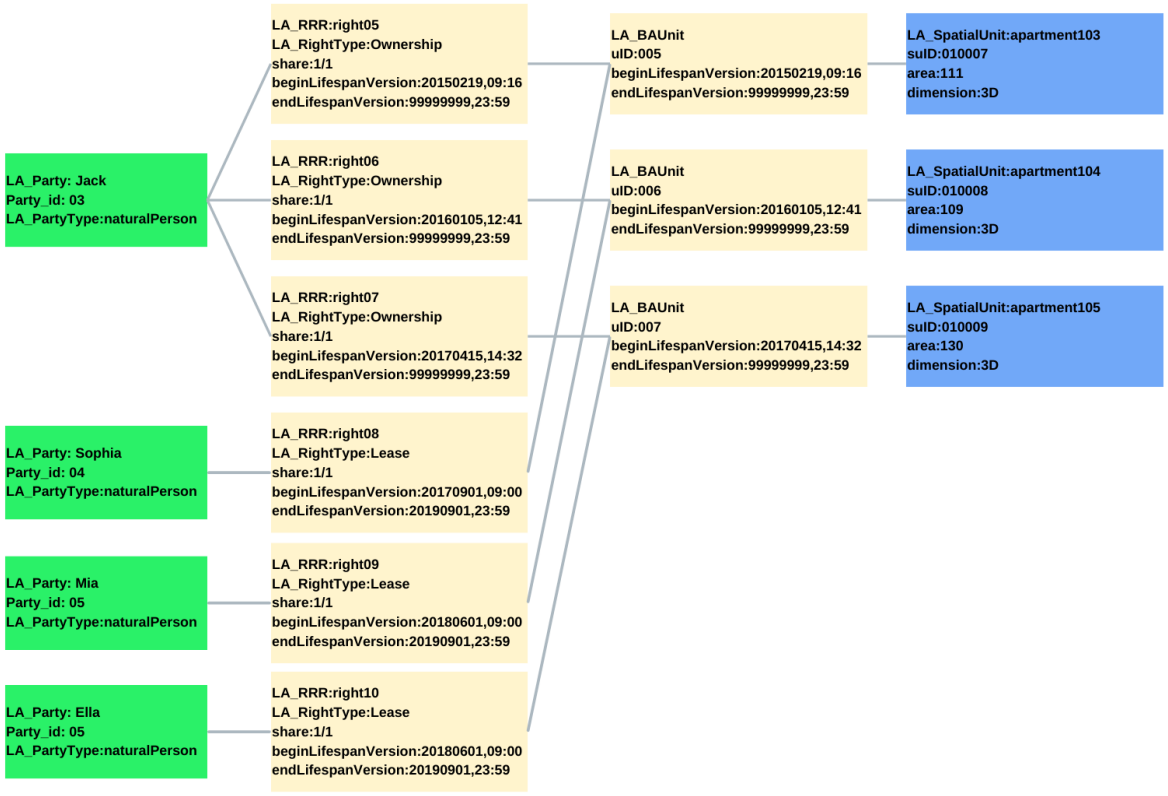


Figure 4.14: LADM Information of Jack

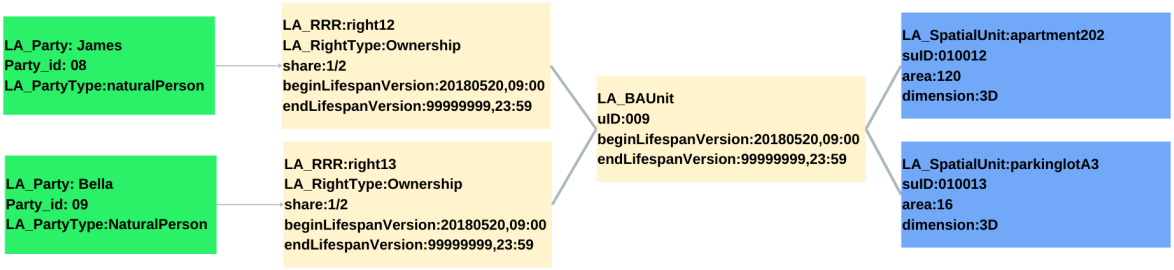


Figure 4.15: LADM Information of James and Bella

4.10.1 Code list value description

This study strictly adheres to the code lists specified in the LADM (ISO/DIS 19152-2, `https://www.iso.org/standard/81264.html`). The specific code list values and their descriptions are detailed in the following Table 4.3 and Table 4.4.

Party Package

Code List Name	Value	Description
LA_PartyType (from LA_Party)	naturalPerson	A natural person entitled to enjoy their property peacefully and without disturbance. No individual can be deprived of their property except under public interest and the conditions prescribed by law and by the general principles of international law.
	nonNaturalPerson	A non-natural person refers to any legal entity that is not a natural person. This includes corporations, government bodies, and other organizations. Unlike natural persons, these entities act collectively through their members to hold and exercise specific rights. The rights held by non-natural persons are recorded formally, ensuring that they can participate in legal transactions and possess property similarly to natural persons[Lemmen et al., 2015].

Table 4.3: Code List for LA_Party

Administrative Package

Code List Name	Value	Definition
LA_RightType (from LA_Right)	commonOwnership	Holding the ownership (of related baunits) together, e.g., within an organization, enterprise, community or State indivisibly rather than in the names of the individual members or groups of members as common property.
	ownership	(1) Ownership is fundamentally a legal process that revolves around the concept of 'title'. A title serves as the evidence that establishes who has the legal right to a property. This process ensures that ownership is recognized and protected by law. (2) Ownership encompasses a collection of rights that allow the owner to use, enjoy, and dispose of the property, including the right to transfer it to others. (3) Ownership also includes the rights to grant a lease, an easement, or a security interest in the property, as well as other subordinate rights that may be attached to it.
	lease	(1) Right to lease a piece of land. (2) A lease is a legal agreement that permits the lessee to use a building, piece of equipment, or land for a defined duration, usually in exchange for rent.
	usufruct	(1) The legal right of using and enjoying the fruits or profits of something belonging to another. (2) Usufruct provides a temporary right to use and benefit from another's property without altering its character(EULIS / ELRA term) .
LA_RestrictionType (from LA_ Restriction)	mortgage	Mortgage restriction

Table 4.4: Code List for LA_RRR

4.10.2 Database Preparation

The system utilizes PostgreSQL to store the LADM information, including the tables LA_Party, LA_RRR, LA_BAU, and LA_SpatialUnit. These tables are essential for managing details about parties, rights, building units, and spatial units, respectively, as show in Figure 4.16, 4.17 4.18and 4.19.

Query Query History

```
1 SELECT * FROM public."LA_Party"
2 ORDER BY "Party_id" ASC
```

Data Output Messages Notifications

	Party_id [PK] character varying	LA_PartyType character varying	Party_name character varying
1	01	nonNaturalPerson	Association of owners
2	02	naturalperson	Mary
3	03	naturalPerson	Jack
4	04	naturalPerson	Sophia
5	05	naturalPerson	Mia
6	06	naturalPerson	Ella
7	07	naturalPerson	Luna
8	08	naturalPerson	James
9	09	naturalPerson	Bella
10	10	naturalPerson	Henry
11	11	naturalPerson	Jackson
12	12	naturalPerson	Alice
13	13	naturalPerson	Alex
14	14	naturalPerson	John
15	15	naturalPerson	Luke
16	16	naturalPerson	William
17	17	naturalPerson	Lucas
18	18	naturalPerson	Jessica
19	19	naturalPerson	Tom
20	20	nonNaturalPerson	Bank
21	21	naturalPerson	Victoria

Total rows: 24 of 24 Query complete 00:00:00.098

Figure 4.16: LA_Party

Query Query History

```
SELECT * FROM public."LA_RRR"
ORDER BY "rID" ASC
```

Data Output Messages Notifications

	rID [PK] character varying	LA_RightType character varying	share character varying	beginLifespanVersion timestamp without time zone	endLifespanVersion timestamp without time zone	LA_RestrictionType character varying
1	restrict01	[null]	1/1	2020-06-23 10:00:00	2050-06-23 10:00:00	mortgage
2	right01	commonOwnership	[null]	2015-01-01 10:00:00	9999-12-31 23:59:59	[null]
3	right02	commonOwnership	[null]	2015-01-01 10:00:00	9999-12-31 23:59:59	[null]
4	right03	Ownership	1/1	2018-07-04 15:17:00	9999-12-31 23:59:59	[null]
5	right04	Ownership	1/1	2016-11-08 11:46:00	9999-12-31 23:59:59	[null]
6	right05	Ownership	1/1	2015-02-19 09:16:00	9999-12-31 23:59:59	[null]
7	right06	Ownership	1/1	2016-01-05 12:41:00	9999-12-31 23:59:59	[null]
8	right07	Ownership	1/1	2017-04-15 14:32:00	9999-12-31 23:59:59	[null]
9	right08	Lease	1/1	2017-09-01 09:00:00	2019-09-01 23:59:59	[null]
10	right09	Lease	1/1	2018-06-01 09:00:00	2019-06-01 23:59:59	[null]
11	right10	Lease	1/1	2018-03-01 09:00:00	2019-03-01 23:59:59	[null]
12	right11	Ownership	1/1	2016-12-23 09:15:00	9999-12-31 23:59:59	[null]
13	right12	Ownership	1/2	2018-05-20 09:09:00	9999-12-31 23:59:59	[null]
14	right13	Ownership	1/2	2018-05-20 09:09:00	9999-12-31 23:59:59	[null]
15	right14	Ownership	1/1	2015-04-01 11:06:00	9999-12-31 23:59:59	[null]
16	right15	Lease	1/2	2017-09-01 09:00:00	2019-09-01 23:59:59	[null]
17	right16	Lease	1/2	2017-09-01 09:00:00	2019-09-01 23:59:59	[null]
18	right17	Ownership	1/1	2018-06-10 13:23:00	9999-12-31 23:59:59	[null]
19	right18	Ownership	1/1	2018-06-10 13:23:00	9999-12-31 23:59:59	[null]
20	right19	usufruct	1/1	2015-02-28 16:34:00	9999-12-31 23:59:59	[null]
21	right20	Ownership	1/1	2015-02-28 16:34:00	9999-12-31 23:59:59	[null]

Figure 4.17: LA_RRR

Query Query History

```
SELECT * FROM public."LA_BAUnit"
ORDER BY id ASC
```

Data Output Messages Notifications

	id [PK] character varying	beginLifespanVersion timestamp without time zone	endLifespanVersion timestamp without time zone	uID character varying
1	1	2015-01-01 00:00:00	9999-12-31 23:59:59	001
2	10	2018-03-01 09:00:00	2019-03-01 23:59:59	007
3	11	2016-12-23 09:15:00	9999-12-31 23:59:59	008
4	12	2018-05-20 09:09:00	9999-12-31 23:59:59	009
5	14	2015-04-01 11:06:00	9999-12-31 23:59:59	010
6	15	2017-09-01 09:00:00	2019-09-01 23:59:59	010
7	17	2018-06-10 13:23:00	9999-12-31 23:59:59	011
8	18	2018-06-10 13:23:00	9999-12-31 23:59:59	012
9	19	2015-02-28 16:34:00	9999-12-31 23:59:59	013
10	2	2015-01-01 00:00:00	9999-12-31 23:59:59	002
11	20	2015-02-28 16:34:00	9999-12-31 23:59:59	013
12	21	2018-12-25 12:01:00	9999-12-31 23:59:59	014
13	22	2015-04-03 08:29:00	2021-08-01 14:04:00	015
14	23	2021-08-01 14:04:00	9999-12-31 23:59:59	016
15	24	2020-06-23 10:00:00	9999-12-31 23:59:59	017
16	25	2020-06-23 10:00:00	2050-06-23 10:00:00	017
17	26	2016-09-14 16:59:00	9999-12-31 23:59:59	018
18	27	2018-08-10 15:09:00	9999-12-31 23:59:59	019
19	28	2016-02-16 09:04:00	9999-12-31 23:59:59	020
20	29	2017-04-27 16:59:00	9999-12-31 23:59:59	021
21	3	2018-07-04 15:17:00	9999-12-31 23:59:59	003

Total rows: 27 of 27 Query complete 00:00:00.109

Figure 4.18: LA_BAUnit

Query Query History

```
1  SELECT * FROM public."LA_SpatialUnit"
2  ORDER BY "suID" ASC
```

Data Output Messages Notifications

	suID [PK] character varying	area numeric	dimension character varying
1	010001	3000	2D
2	010002	630	3D
3	010003	300	3D
4	010004	110	3D
5	010005	16	3D
6	010006	120	3D
7	010007	111	3D
8	010008	109	3D
9	010009	130	3D
10	010010	110	3D
11	010011	16	3D
12	010012	120	3D
13	010013	16	3D
14	010014	111	3D
15	010015	16	3D
16	010016	109	3D
17	010017	16	3D
18	010018	130	3D
19	010019	16	3D
20	010020	110	3D
21	010021	16	3D

Total rows: 37 of 37 Query complete 00:00:00.094

Figure 4.19: LA_SpatialUnit

4.10.3 Frontend-Backend Data Interaction

The backend server is set up using Node.js, as detailed in the `server.js` file. This file uses the Express framework to configure the server and interacts with the PostgreSQL database through the pg module.

In the `server.js` file, an API endpoint `/query/queryall` is defined to handle POST requests sent by the client. The parameters for connecting to the PostgreSQL database storing LADM

data need to be properly set.

On the frontend, Axios is used to send POST requests to the backend API endpoint, containing a `sql` field in the request body that specifies the SQL query to be executed.

The server receives the request and extracts the SQL query from the request body. Then, it executes the SQL query using the `pg` module and returns the query results. Specifically:

- `t_apartment`: Contains a static SQL query string used to select all records from the `public.t_apartment` table.
- `LA_Party`, `LA_RRR`, `LA_BAUnit`, `LA_SpatialUnit`: These are functions that accept an `id` parameter and return a dynamically generated SQL query string used to select records from the corresponding table based on the `id`.

4.10.4 Data Transformation and Classification

The data transformation and classification process begins with sending a POST request to the backend API endpoint `/query/queryall` using Axios. . This request includes an SQL statement `SELECT * FROM public.t_apartment` to query data from the `t_apartment` table.

Next, the transformData method processes the raw data retrieved from the `t_apartment` table. This method iterates through each data item and sends additional SQL queries for each field (such as `LA_RRR_id`) to obtain detailed information. For example, to query the `LA_Party_id`) field, the `LA_Party` table is queried with the SQL statement:

```
SELECT * FROM public."LA_Party" WHERE "Party_id" = '${id}'
```

Similar queries are performed for the `LA_RRR`, `LA_BAUnit`, and `LA_SpatialUnit` tables. After the transformData method completes, the returned data includes comprehensive LADM information from the `LA_Party`, `LA_RRR`, `LA_BAUnit`, and `LA_SpatialUnit` tables.

Subsequently, the `categorizeToNameWithMap` method is used to classify the data based on specific fields such as `Apartment_id` and `Party_name`.. The classified data is stored in two Map objects. This classification supports the query functionality of the 3D system.

In this implementation, Map objects are used to store categorized data, allowing for efficient retrieval based on a specific key. One Map object (relationship_Building) categorizes the data by Apartment_id, allowing for quick access to all records related to a specific apartment. Another Map object (relationship_Name) categorizes the data by Party_name, facilitating efficient retrieval of all records related to a specific party. This structured approach enhances the performance and organization of the data management system, facilitating subsequent query operations.

4.10.5 LADM Chart Display

To visualize LADM information, the AntV G6 graphics library is used.

To initialize the graph instance, use the `G6.Graph` constructor and pass the configuration parameters. These parameters include `container` (the container ID), `width` and `height` (the container's width and height), `modes` (user interaction modes), `minZoom` and `maxZoom` (zoom scaling), and `layout` (layout type). The `modes` property is configured to enable `drag-canvas` and `zoom-canvas` modes, allowing users to drag and zoom the graph. And an event listener for the `window.onresize` event is set up to dynamically adjust the graph size based on window size changes.

4.10.6 Click Query Process

The process includes setting up a click event handler, retrieving the clicked model, sending the model's ID, highlighting the clicked model, triggering an event bus, listening for the event, and finally, initializing and processing the data associated with the click event.

First, in the custom `pickEntity` method, set up a click event handler using `pickHandler. setInputAction` to listen for the user's left-click events (`Cesium.ScreenSpaceEventType.LEFT_CLICK`).

Next, retrieve the clicked model by using the `drillPick(event.position)` method, which returns an array of all models at the clicked position. Specify the first model clicked using `pickmodels[0]`.

Then, set the clicked model's ID and type to "click", and send them using the `sendID` method:

```
this.sendID({ type: 'click', content: pick.primitive.id });
```

Afterward, update the selected model's color to highlight it using `Cesium.Color.fromCssColorString`.

Following this, the `sendID` method triggers the `change-show-info-mitt` event via the event bus `$bus`:

```
sendID(obj) {
  this.\$bus.emit("change-show-info-mitt", obj);
}
```

In response, the `SelectInfo` component's `mounted` hook listens for the `change-show-info-mitt` event and calls the `changeDataInit` method for data initialization and processing:

```
mounted() {
  this.\$bus.on("change-show-info-mitt", (input) => {
    if (input !== "none") {
      this.changeDataInit(input);
    } else {
      this.show_controller = false;
    }
  });
}
```

Finally, the changeDataInit method processes input.type and input.content to retrieve or update the corresponding data and view.

Example Process

Suppose a user clicks on a model with the ID "1-02":

1. **Trigger** sendID: The sendID method is called and sends type: 'click', content: '1-02' .
2. **Event Bus Sends Event**: this.$bus.emit("change-show-info-mitt", type: 'click', content: '1-02').
3. **SelectInfo Component Listens to Event**: The SelectInfo component listens for the change-show-info-mitt event and calls the changeDataInit method.
4. **Process** changeDataInit:
   - input.type is click and input.content is "1-02".
   - Process the ID to obtain idkey as "102".
   - Since idkey has a value, proceed with updating the view.

4.10.7 Search Query Process

This section details the steps involved in executing a search query within the application. The process begins with the user initiating the search query and culminates in displaying the required data.

Initially, the user activates the search box by clicking a designated button, setting the search_info_show property to true, which makes the search interface visible. The user then enters their query information, selecting a query type (such as "Party name" or "Apartment id") and providing the corresponding query value. For instance, selecting label="Party name" with :value="select_list[0]" refers to the Party_name field in the AppConfig.js, which corresponds to the Party_name in the t_apartment table.

Upon entering the query, the user clicks the search button, triggering the sendID method. This method sends the selected query type and input value as parameters, emitting the change-show-info-mitt event through an event bus. The relevant code snippet for this process is:

```
<el-button slot="append" icon="el-icon-search"
@click="sendID({type: search_type, content: search_content})">
</el-button>
```

```
sendID(obj) {
  this.$bus.emit("change-show-info-mitt", obj);
}
```

The application listens for the change-show-info-mitt event within the SelectInfo component's mounted hook. Once the event is detected, the changeDataInit method is invoked to initialize and process the data. This method processes the input.type and input.content, retrieving or updating the corresponding data and view.

For instance, if the user selects "Party name," enters "Jack," and clicks the search button, the sendID method is invoked with the parameters {type: 'Party name', content: 'Jack'}. The event bus then emits the change-show-info-mitt event, which triggers the changeDataInit method to query the database for information related to "Jack."

The relationship_Name map is created based on the Party_name field in the t_apartment table using the categorizeToNameWithMap method, where idkey refers to "Jack." The detailed process involves making a POST request to the backend API endpoint to execute an SQL query, retrieving all data from the t_apartment table. The transformData method processes this data and sends additional SQL queries to fetch comprehensive LADM information from the LA_Party, LA_RRR, LA_BAUnit, and LA_SpatialUnit tables.

The transformed data is categorized and stored in Map objects based on the specified fields Apartment_id and Party_name using the categorizeToNameWithMap(data, key) method. The data is categorized by Party_name, resulting in a Map object where keys are Party_name values, and values are arrays of records related to that Party_name. For instance, "Jack" may have multiple Apartment_id records, each with associated information. Finally, the system updates the view with the retrieved data, providing the user with the requested information.

The entire process ensures that users can efficiently search for and retrieve detailed information about property ownership, tenancy, and related attributes based on the selected search criteria.

Example Process

Assume the user selects "Party name" in the search box, enters "Jack," and clicks the search button. The process is as follows:

1. Invoke sendID Method:

```
sendID({ type: 'Party name', content: 'Jack' }) {
  this.$bus.emit("change-show-info-mitt",
  { type: 'Party name', content: 'Jack' });
}
```

2. Trigger Event Bus: The sendID method triggers the change-show-info-mitt event through the event bus.
3. Initialize and Process Data: The changeDataInit method processes the input.type and input.content to query the corresponding database information:

```
changeDataInit({ type: 'Party name', content: 'Jack' })
```

4. Relationship Mapping: The relationship_Name map is created based on the Party_name field in the t_apartment table using the categorizeToNameWithMap method. Here, idkey refers to Party_name, which is "Jack."

5. Data Retrieval: A POST request is made to the backend API endpoint to execute an SQL query retrieving all data from the `t_apartment` table. The `transformData` method processes this data and sends additional SQL queries to retrieve comprehensive LADM information (e.g., `LA_Party`, `LA_RRR`, `LA_BAUnit`, `LA_SpatialUnit` tables). The transformed data is then categorized and stored in two Map objects based on the specified fields `Apartment_id` and `Party_name` using the `categorizeToNameWithMap(data, key)` method.
6. Example Map Object: Data is categorized by `Party_name`, resulting in a Map object where keys are `Party_name` values, and values are arrays of records related to that `Party_name`. In this example, "Jack" has three different `Apartment_id` records, each with associated information.
7. View Update: If data is available, the corresponding view is updated.

4.10.8 Convert Data into Graph Format

This subsection details the methods used to handle and process property and rental information within the system, ultimately transforming this data into a graph format suitable for visualization using AntV G6. Each method has a distinct purpose, ranging from gathering room and owner details to converting data into a visualizable format.

getRoomProp Method

The `getRoomProp` method collects information about the owners and tenants of a specified room. The process is as follows:

First, the method checks if the specified `room_id` exists in the `RoomList` using `this.RoomList.has(room_id)`. If the room exists, the method iterates through all related information using `this.RoomList.get(room_id).fo` For each record, it checks if `LA_RightType` is "Ownership" or "usufruct" and, if true, adds the owner's name to the `roomowner` array. All room information, including tenant details, is added to `roomdata`.

Next, the method iterates through each owner name in the `roomowner` array, as there may be multiple owners for a single property. For each owner, it retrieves all information from the `nameList` using `this.nameList.get(name).forEach()` and adds any unique data to `roomdata`. Finally, `roomdata` is passed to the `getAntVG6Data` method to convert it into the AntV G6 format.

getPeopleProp Method

The `getPeopleProp` method retrieves property and rental information for a specific individual. The method iterates over the individual's records in the `nameList` to determine if they are an owner or a tenant. If the individual is an owner, the property ID is added to `ownerRooms`; otherwise, it is added to `leaseRooms`. The method uses `roomdata` to store the combination of room ID and the person's name.

The method then iterates through `ownerRooms` to get information about all tenants in the individual's properties and through `leaseRooms` to get information about the owners of the

properties the individual is renting. For each rented property, it retrieves the owner's name from `nameList` and stores detailed information in `roomdata`. Finally, `roomdata` is converted to the AntV G6 graph format.

getBankProp Method

The `getBankProp` method retrieves all property information related to a specified bank, including rooms with bank loans and related loan information. The method first retrieves all properties related to the bank from the `nameList` and records the property IDs (`Apartment_id`) in `bankRooms`. Each piece of information is stored in `roomdata`.

The method then iterates through each room ID in `bankRooms` to check if the corresponding room information exists in `RoomList` and retrieves the loan holders' names for each room, recording them in the `people` array. It then checks if `nameList` contains information about each loan holder and retrieves all properties owned by each loan holder, storing this information in `roomdata`. Finally, `roomdata` is converted to the AntV G6 standard format.

getAntVG6Data Method

The `getAntVG6Data` method converts a given data list (`datalist`) into the AntV G6 graph format, handling nodes and edges. The method initializes variables such as `LA_Party`, `LA_RRR`, `LA_BAUnit`, and `LA_SpatialUnit` as Maps to store node data, an `id` counter for generating unique node IDs, and a `data` object to store the final node and edge data structure.

The method processes each type of data by checking if it already exists in the respective Map. If it exists, it uses the existing ID; otherwise, it creates a new node, updates the Map, and adds it to `data.nodes`. The method also updates `data.edges` to connect the `source` and `target` for each type of data.

5 Results

This chapter presents the functionality and significance of the 3D LAS developed in this study. It encompasses the integration of 3D BAG and BIM models, interactivity features, floor visibility control, sunlight simulation, dynamic slicing views, and underground space visualization. Additionally, it details the usability testing processes conducted to evaluate system usability and gather feedback for future improvements.

5.1 Loading 3D BAG and BIM Models

The concept of geospace in 3D LAS encompasses two primary dimensions: 3D physical space and 3D legal space. It is crucial to accurately display the location and texture information of apartment buildings on web platforms [Ying et al., 2012].

By correctly loading 3D BAG data and BIM models, the platform ensures that all components are accurately positioned and oriented, reflecting real-world scenarios, as shown in Figure 5.1 and 5.2. Additionally, In addition, the visualization requirement of transparency is used to distinguish physical objects from legal objects. The ability to load models with custom shaders and specific attributes allows for a high degree of customization. Users can tailor the visual appearance and behavior of the models to meet specific needs, such as visualizing legal spaces in this study.

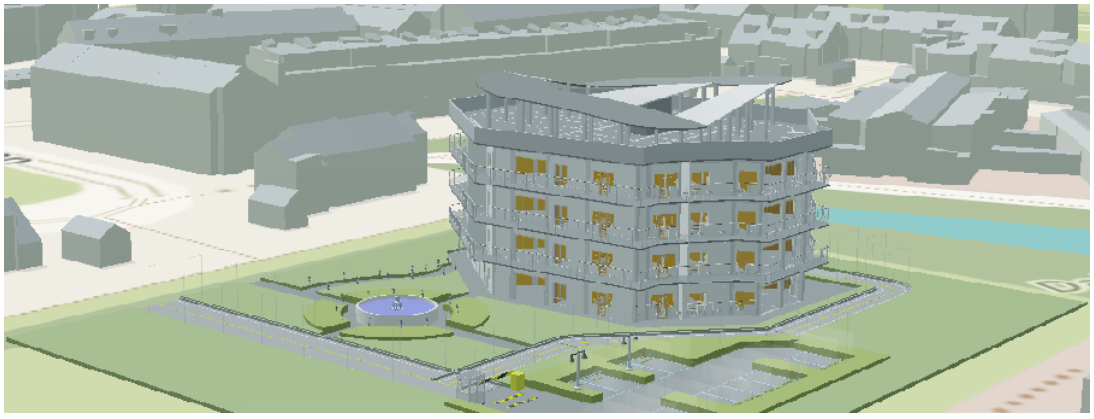


Figure 5.1: 3D physical model

This enables users to clearly and intuitively distinguish and locate each apartment unit, ensuring detailed and clear visualization of the apartment building on web platforms. Consequently, this enhances user interaction and understanding of complex land administration data.

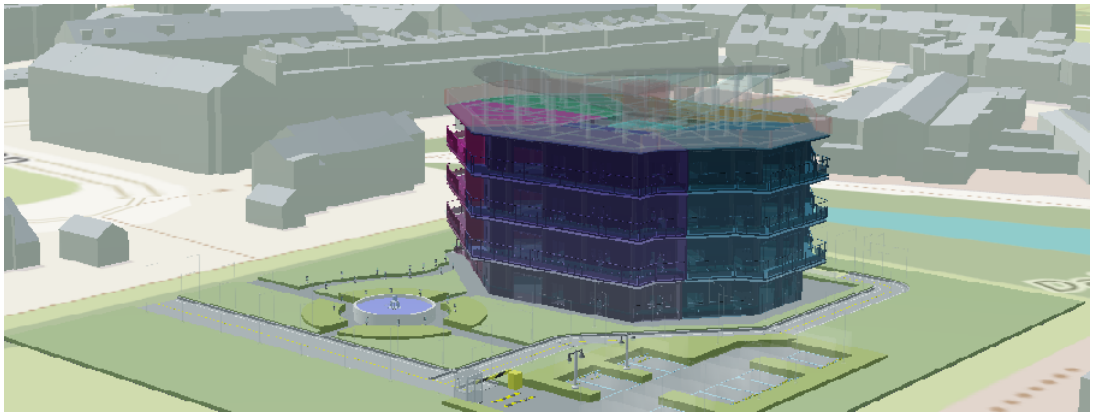


Figure 5.2: 3D legal model

5.2 Interactivity

In 3D land administration system, interactivity is a critical feature that not only enhances user engagement but also significantly improves the practicality of the system[Shojaei et al., 2013]. This study developed interactive functionalities that allow users to explore three-dimensional spaces through intuitive mouse operations. These operations include panning with the left mouse button, rotating with the right mouse button, and zooming with the scroll wheel. This interaction mode enhances the intuitiveness of user operations and enables users to observe the model from multiple perspectives and different zoom levels, providing a more comprehensive three-dimensional view. Additionally, users can explore the spatial relationship between apartment buildings and their surrounding environment by changing the basemap, as shown in Figure 5.3.

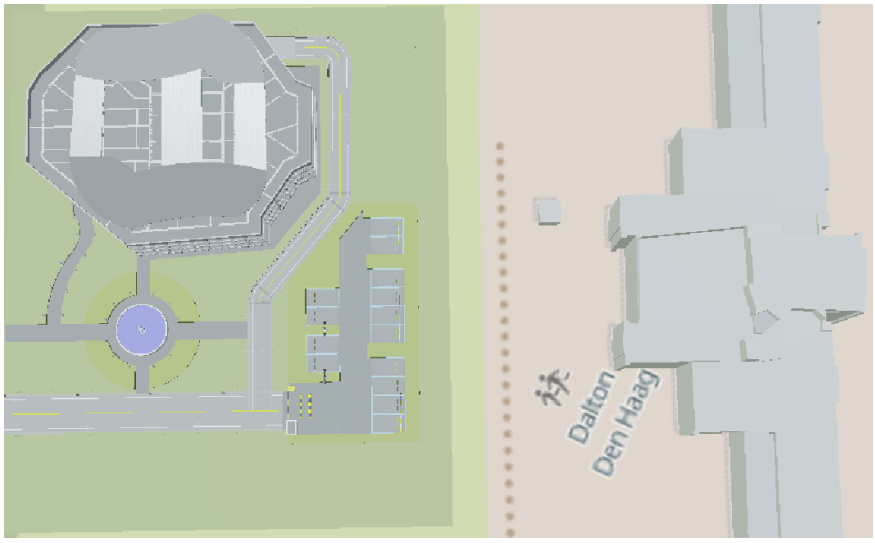


Figure 5.3: surrounding environment

5.3 Floor Visibility Control

This feature allows users to view different parts of the building as needed, enabling a clear and detailed examination of each floor's internal structure. The effect is illustrated in the accompanying figure 5.4.

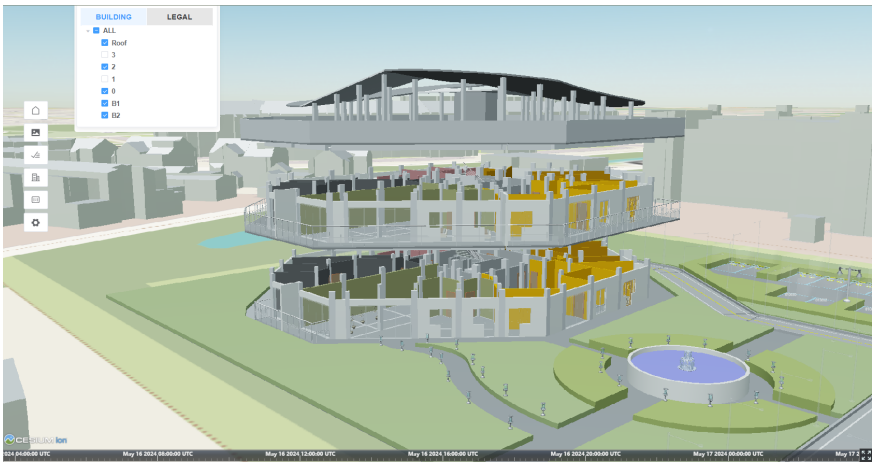


Figure 5.4: Floor Visibility Control

5.4 Sunlight Simulation

The sunlight simulation feature allows users to flexibly select and view scene effects at different times of the day, enhancing the overall user experience. This dynamic update capability enables real-time simulation of sunlight changes at any time, not just static time points. Users can intuitively see the lighting effects at various times, aiding in decision-making. The effects are illustrated as shown in the following Figure5.5. For instance, For example, this feature can be beneficial for architects who need to optimize building designs to ensure that the structure receives sufficient natural light throughout the day. The overall sunlight exposure of the building can be checked and whether it is blocked by surrounding buildings, accurately simulating the building's natural light exposure during the day. Sunlight conditions in different seasons and times of day. This is particularly important for optimizing building design, not only improving the comfort and liveability of the space, but also helping to reduce the building's environmental impact by minimizing the need for artificial lighting. Potential homebuyers can also benefit from assessing a property's lighting conditions at different times, helping them make an informed purchase. Potential property buyers can also benefit by assessing the lighting conditions of a property at different times, helping them make informed purchase decisions.

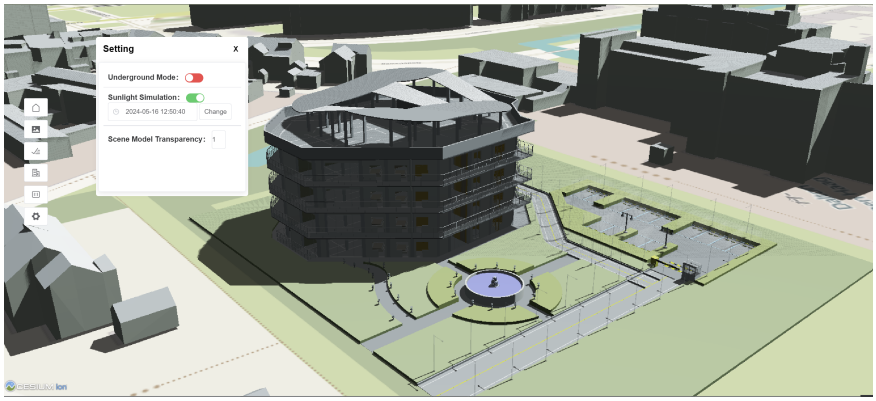


Figure 5.5: Sunlight Simulation

5.5 Dynamic Slicing View

When purchasing an apartment in the Netherlands, buyers receive a property deed and a division deed. The division deed includes division diagrams, which provide a 2D description of the apartment rights. The development of the dynamic cross-section view functionality aligns with these requirements by allowing users to dynamically and clearly view cross-section diagrams, thereby gaining a more intuitive understanding of the public and private spaces within a building.

Additionally, this functionality enables users to inspect the internal structure and layout of building models, examining design details without the need for physical edits or permanent modifications. This tool enhances the understanding of apartment right and interior design details, which is beneficial for both prospective buyers and architects. The effectiveness of this functionality is demonstrated in Figure 5.6. When the floor control is activated, an overlay effect is also available, allowing the selected floor to be fully displayed while dynamically clipping other unselected floors. The effect of this feature is shown in Figure 5.7.

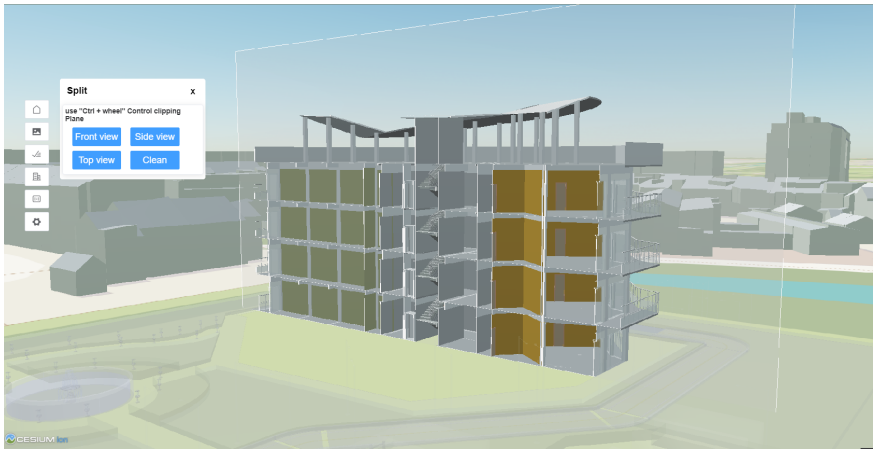


Figure 5.6: Cross Section View

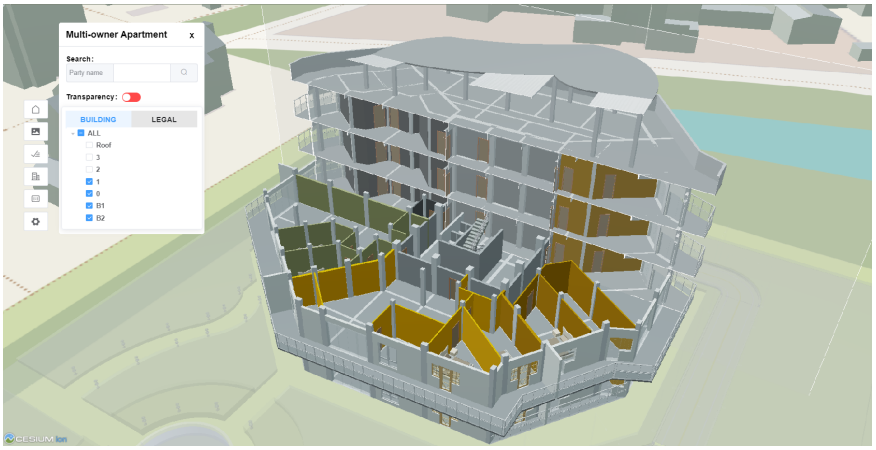


Figure 5.7: Overlay effect showing selected floor

5.6 Visualizing underground space Functionality

3D land administration system capable of representing objects above, below and through the ground surface[Vandysheva et al., 2011]. Visualizing underground spaces is crucial in a 3D land administration system. Many essential infrastructures, such as utilities, transportation networks, and communication systems, are located underground. By visualizing these elements, we can significantly improve planning, maintenance, and risk identification capabilities.

This study focuses specifically on the visualization of apartment rights. According to Article 5:106 of the Dutch Civil Code, the "...exclusive right to use certain parts of the building, which parts are to be used, as a separate private unit..." implies that apartment units may include private underground spaces, such as basements or garages.

For users, such as buyers and investors, these underground spaces are often seen as valuable additions to the property. Visualizing these spaces enhances the property's appeal and market value. The following Figure 5.8 illustrates the functionality of this feature:

Figure 5.8: Visualization of underground spaces

5.7 Dynamic Data Display

Upon receiving responses from the backend, the frontend dynamically updates to display information related to LADM and basic apartment details. This ensures that users have access to real-time data updates.

Interactive Graphical Representation:
The front end includes an interactive graphical representation(`LADM_GraphBox`) for visualizing relationships within the LADM data. It is powered by antv/g6, handles data conversion and display, converts input data into nodes and edges of relationships in LADM for visualization, enabling users to visualize complex property relationships and rights structures. It configures the graph with custom node styles, interaction modes (e.g., drag-canvas, zoom-canvas), and layout settings.

Data Handling and Visualization:
The graph's data model is prepared to handle updates dynamically. When the database data is updated, the graph automatically refreshes to reflect the changes, maintaining an up-to-date visual representation of the LADM relationships.

Rich Data Processing:
The class handles various types of queries, such as ownership details, lease information, and more complex associations like banking details related to properties. It uses custom logic to determine the type of each entity (e.g., owner vs. leaseholder) and appropriately formats the data for graphical representation.

Basic information and LADM information query results, as shown in the figure 5.9

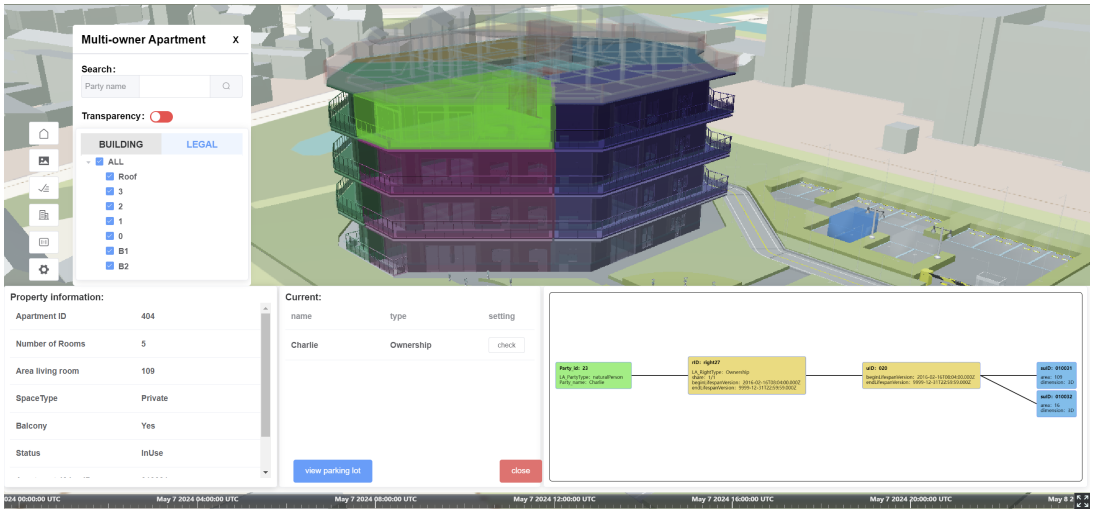


Figure 5.9: Property information

5.7.1 Personal Property Inquiry

Function Description: When the user clicks the ”check” button, the information view in the lower-left corner of the system refreshes to display a list of all properties owned by the selected person, as shown in Figure 5.10. By selecting one of the listed apartments and clicking the ”Fly to” button, the system will then display the basic property information, resident information, and LADM information for that apartment, as illustrated in Figure 2.

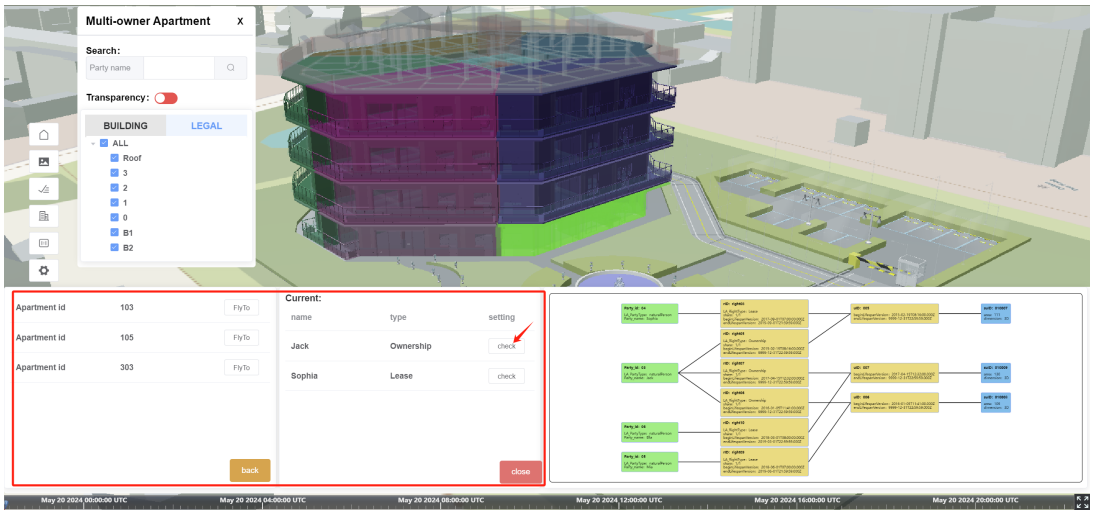


Figure 5.10: click the ”check” button

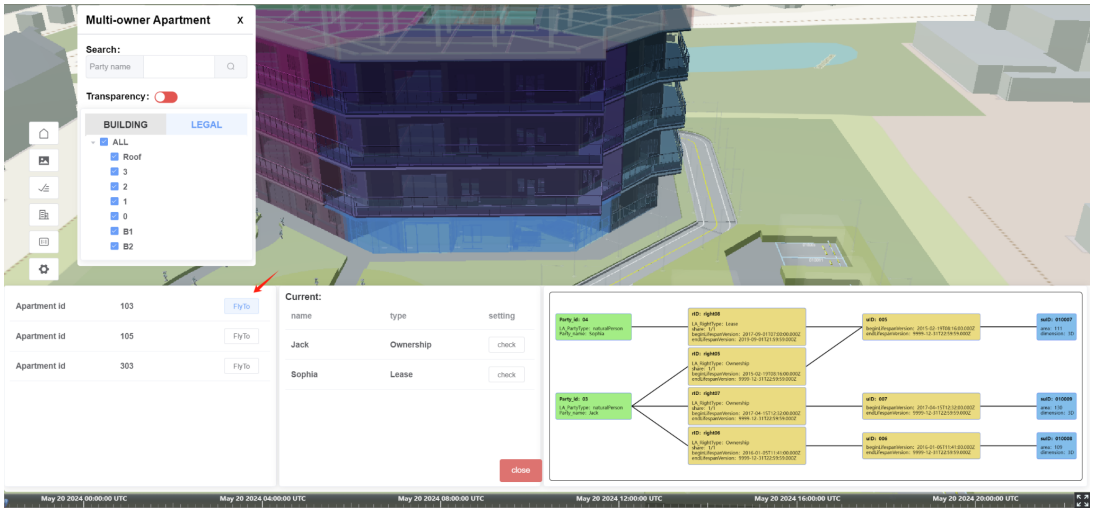


Figure 5.11: clicks the "Fly to" button

5.7.2 Search and Information Display

This feature supports two search modes: by Party Name and by Apartment ID. As illustrated in the figures:

Property Owner and Tenant Information Query

If the search is conducted using the name of an apartment owner, the system displays all information related to the owner. This includes all properties owned by them and information about tenants renting their properties. If the search is conducted using the name of a tenant, the system displays all information related to the tenant and the property owner. It will not show information about other tenants renting from the same owner.

Property Information Query

When the search is based on the Apartment ID, the system displays all information related to the property owner and the tenants of this apartment. It will not show information about other tenants renting from the same owner.

The figures 5.12, 5.13, 5.14, and 5.15 show how the system responds to these search queries, providing detailed information about owners, tenants, and properties based on the selected search criteria.

Figure 5.12: Search by Association of owners

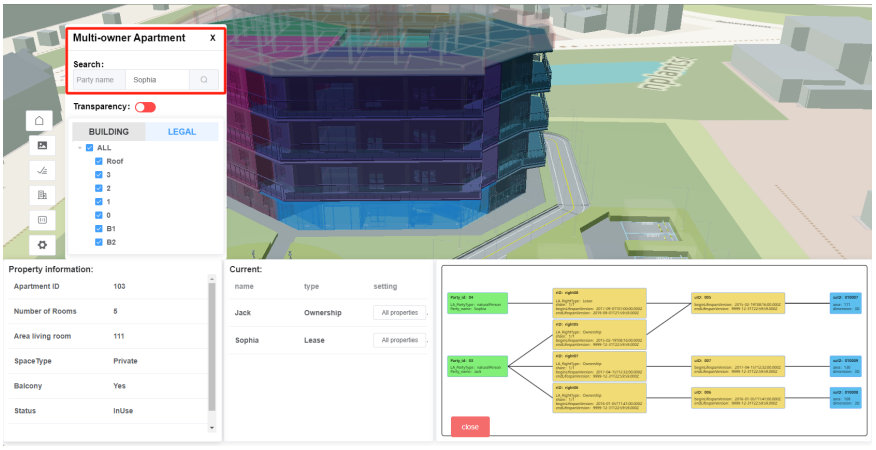


Figure 5.13: Search by Tenant Name

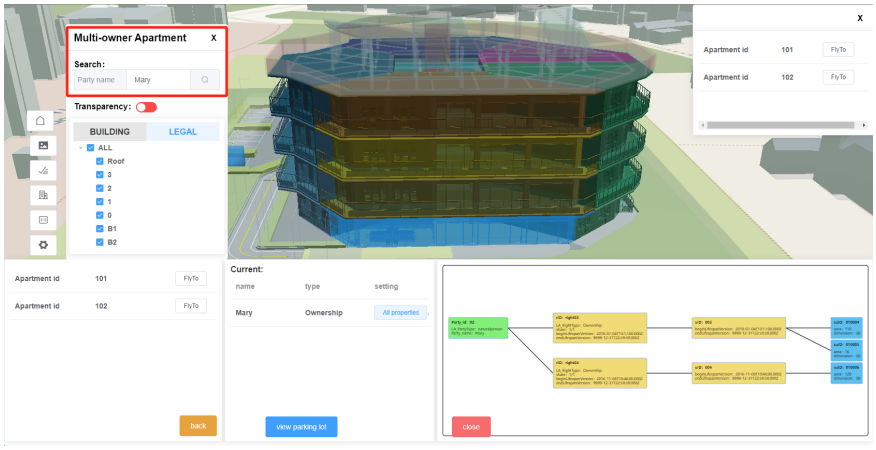


Figure 5.14: Search by Owner Name

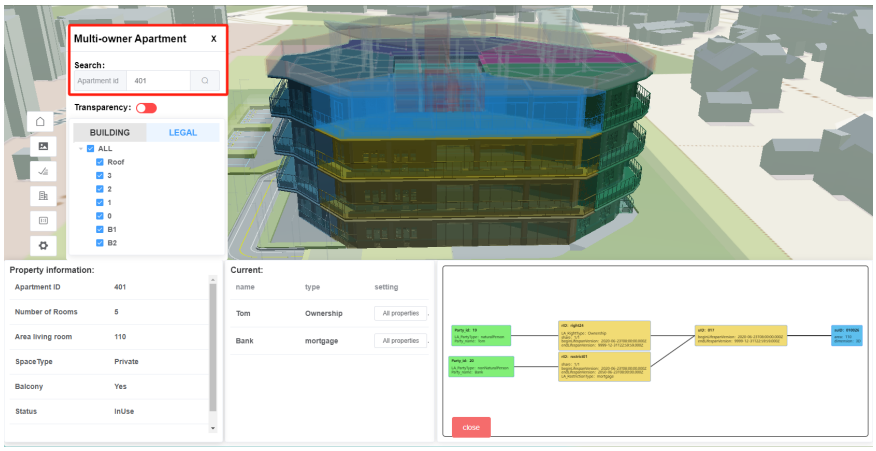


Figure 5.15: Search by Apartment ID

5.8 Usability testing

Usability testing allows developers to directly assess the usability and user-friendliness of their applications, gaining insights into potential issues encountered by testers. Through usability testing, the goal is to gather feedback from participants to understand the strengths and weaknesses of the current system.

According to ISO 9241-11, usability is defined as ”the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.” Usability encompasses effectiveness, which determines whether users can complete tasks and achieve their goals, and user satisfaction, which relates to their opinions and perceptions of the product. This study measures usability based on whether users can correctly answer questions and score the system while collecting their feedback. It is important to note that the participants in this usability test do not fully

represent the actual user group. Despite these limitations, usability testing provides direct user suggestions, reveals potential issues, and identifies areas for future optimization.

5.8.1 User Groups

- Individuals familiar with LADM
- Individuals with limited knowledge of LADM
- Individuals completely unfamiliar with LADM

5.8.2 Questionnaire Design

The questionnaire was created using Google Forms, providing context for test users and explaining LADM. To gather feedback, specific tasks were defined for users to perform and answer questions. The tasks for user testing included:

1. Switching base maps, panning, zooming, and rotating the view to locate target areas, testing 3D scene operations.
2. Controlling the visibility of floors.
3. Visualizing underground spaces.
4. Viewing ownership information.
5. Simulating lighting conditions.

All tasks were designed clearly and non-misleadingly, with simple reminders to help users find the relevant controls.

5.8.3 Test Results

Figure 5.16 shows the familiarity of the users with LADM. The majority of the participants were either generally familiar or not familiar with LADM, with only a small portion being experts. This distribution helps in understanding how different user groups interact with the system and provides insights into usability across varying levels of familiarity.

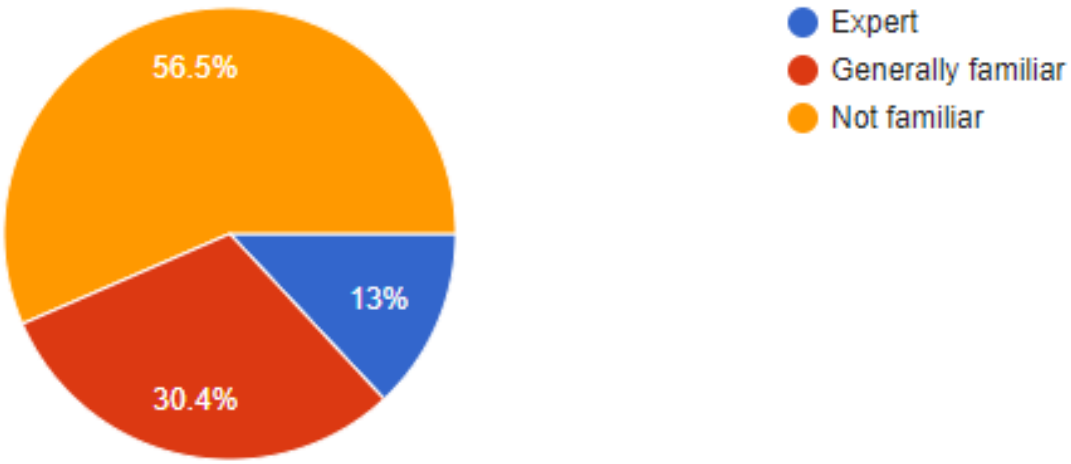


Figure 5.16: Pie chart of three types of participants

- Navigation (panning, zooming, rotating): 8.3
- Floor visibility and underground space: 9.5
- Viewing ownership information: 9.2
- Lighting simulation: 7.0

From the test results, it is evident that regardless of the users' familiarity with LADM, the UML instance-level visualization allowed them to directly access information without confusion about terminology. Users also provided valuable suggestions, such as preferring a time slider over a time selector for timeline navigation, enabling the Enter key for search functionality in addition to the search icon, and adding prompts about case sensitivity in searches. Other recommendations included adding a first-person view to navigate inside buildings, a general search bar, and incorporating more types of 3D RRRs (such as tunnels, airspace, and mining rights).

5.9 Reflection on system

This study leverages data from 3D BAG, and BIM model, along with legal data based on LADM assumptions. Accurately displaying data that reflects real-world scenarios and providing users with interactive, real-time information are key focuses of this research.

The system integrates various functionalities to enhance the visualization, interaction, and understanding of land administration data. The ability to switch between 3D physical models and 3D legal space models provides users with clear and intuitive representations of both physical and legal spaces. Interactive features, such as panning, rotating, and zooming, support user exploration of the 3D environment. Additionally, rich component functionalities—such as floor visibility control, underground space visualization, sunlight simulation, and dynamic slicing views—enhance the system's versatility and usability.

LADM information is dynamically displayed and updated in graphical form on the frontend, visualizing complex legal relationships. This not only provides practical tools for professionals but also offers an easier-to-understand format for users unfamiliar with these concepts. Features like personal property inquiry (which supports searches by party name to display all properties under an owner's name) provide comprehensive information.

Challenges Ensuring a solid understanding of LADM and apartment rights was crucial. The combination of technical development and domain expertise is indispensable for system development.

6 Conclusion and Future Work

In the final chapter of this thesis, the research questions are reviewed to evaluate the extent to which they were resolved. Building on this assessment, the study's limitations are documented and recommendations for future work to overcome these constraints are offered.

6.1 Research overview

1. QUESTION:

How does the integration of a Digital Twin, encompassing both legal data of 3D property rights and surrounding environmental factors, enhance the visualization and understanding of 3D LAS complexities?

ANSWER:
This study significantly enhances the visualization and understanding of the complexities of 3D LAS through several key functionalities:

- **User Test Feedback**: Usability testing demonstrated that users, regardless of their familiarity with the Land Administration Domain Model (LADM), were able to retrieve accurate information through the system. This indicates that the integration of Digital Twin technology enhances the ability to comprehend and interact with complex land administration data.
- **UML Instance-Level Diagrams**: UML instance-level diagrams within the system provide a more intuitive and visual representation of LADM information. These diagrams make it easier to understand the relationships between parties and spatial units.
- **Legal Data Integration**: The LADM provides a standardized framework for property rights management. Integrating LADM into the Digital Twin platform allows users to view all relevant LADM information associated with different apartment units in real-time, including property rights details. This transparency is crucial for buyers, sellers, and legal advisors, as it directly impacts decision-making processes. The real-time nature of the Digital Twin helps in updating legal information (e.g., changes in ownership) as they occur, ensuring all stakeholders have access to the most current information, which is vital for informed decisions regarding real estate development, investment, and management.
- **Surrounding Environmental Information Integration**: The visual representation of green spaces, parking areas, and surrounding buildings not only enhances the visual experience but also significantly improves the ability to assess potential living conditions, thereby increasing user satisfaction. Before deciding to buy or rent a property, users can intuitively and in real-time explore various aspects of their future living environment, such as nearby green spaces, parking facilities, and adjacent amenities. This

intuitive display helps better understand the advantages and limitations of potential residences, leading to more informed decisions that align with their expectations. Additionally, users can utilize the sunlight simulation feature of the digital twin platform to check the apartment's sunlight exposure and determine whether it will be shaded by neighboring buildings.

- **Improved Communication and Decision-Making**: The clarity and interactivity of the platform facilitate better communication among stakeholders, including homeowners, developers, urban planners, and legal professionals. By providing a more intuitive and easily interpretable way to view and understand complex Land Administration information, it supports more effective decision-making and collaboration.

2. QUESTION:

Why is the inclusion of a building's surrounding environment critical when visualizing 3D LAS data?

ANSWER:
Including a building's surrounding environment in 3D LAS visualization is critical for several reasons:

- **Potential Value in Real Estate and Market Analysis**: A comprehensive view of real estate, including the surrounding environment, significantly impacts market analysis, valuation, and marketing strategies. It helps buyers and investors understand the property environment, accessibility, community facilities, and overall appeal.
- **Potential Benefits for Notaries**: Comprehensive visualization, including surroundings, helps notaries conduct more thorough investigations, prevent disputes, and ensure transactions are based on accurate information. 3D visualizations can aid in creating more accurate descriptions and documentation.
- **Benefits for Property Transactions**: 3D visualizations provide detailed and accurate information about the property, its location, and its surrounding environment. This is crucial for both buyers and sellers during a transaction. For example, buyers can see the exact layout of the property, including its dimensions, topography, and proximity to neighboring buildings and amenities. This transparency reduces misunderstandings and increases buyer confidence.
- **Public Engagement and Transparency**: Detailed 3D visualizations that include the surrounding environment enhance public engagement by making it easier for residents to understand proposed changes in their community. This transparency helps build trust and facilitates smoother approval processes for new developments.

3. QUESTION:

What are the essential requirements for a 3D web viewer system that aims to integrate detailed land administration data?

ANSWER:

This study explores the essential requirements for a 3D web viewer system designed to integrate detailed land administration data effectively.

- **Data Format Conversion**: This study utilizes Cesium, an open-source JavaScript library designed for displaying three-dimensional Earth and map data. Cesium supports formats such as GLTF, GLB, and 3D Tiles. Thus, the initial step for effectively integrating detailed model datasets into the 3D Web viewer system involves converting the data into formats supported by Cesium.
- **Data Integration and Interoperability**: The system must be capable of integrating various data sources, including GIS data, land administration data data, and building models. It should ensure compliance with relevant standards such as OGC (Open Geospatial Consortium) standards for geospatial data and ISO standards for land administration.
- **User Interaction and Interface**:
  - **Design Principles**: The user interface is designed to be intuitive and accessible for users of all skill levels, featuring straightforward navigational buttons and components. This design philosophy ensures that the system can be easily used by both novice and professional users.
  - **Interactive Features**: The inclusion of interactive elements such as the ability to click on apartment models or search by apartment id or resident name for querying LADM information enriches user engagement. Advanced functionalities like drag-and-drop, rotation, and translation of model parts enhance the interactive experience, allowing for detailed examination and manipulation of 3D models.
- **Real-Time Land Administration Data Updates**: The digital twin system should support real-time data updates. This system updates legal property rights information in real time through the background database and synchronizes it to the front end to present it to users.

4. QUESTION:

How does the system integrate the LADM information from the database with the 3D models, enabling users to query corresponding model information?

ANSWER:

- **Load Models and Set IDs:** In the `addGLTF` method, assign a unique id to each model. These id are predefined during model loading and are used for subsequent queries and processing.
- **Set Click Event Handler:** Use the `pickHandler.setInputAction` method to set up a click event handler that listens for left-click events. `pick.primitive.id` is used here to get the id of the clicked model.
- **Call sendID Method:** Invoke the `sendID` method with the model's id and the type "click". This method sends the id to the `SelectInfo` component.
- **SelectInfo Component Processing:** The `SelectInfo` component calls the changeDataInit method, which updates the component using the information from the clicked model.

- **Load Apartmnet Basic Information:** Apartment basic information is stored in the legal_info_list.json file. A map named LegalInfoList is constructed using this file, where id are used as keys, and other related attributes and values are stored as values. Thus, the basic information of the apartment displayed in the bottom-left corner of the system can be updated based on the id.
- **Load Resident Information:** In the `changeDataInit` method, the id obtained from the click event is processed into `idkey`. This `idkey` is used to get the corresponding apartment's resident information, updating the resident information displayed in the bottom center of the system.
  - Using `idkey`, the corresponding apartment's information list is retrieved via relationship_Building.get(idkey). This list is traversed, and the information is updated into the `current_owner` and `old_owner` arrays, which are then displayed in the bottom center of the system.
- **Load LADM Information:** Clicking on a model retrieves the corresponding LADM information because the id is obtained from `IDmap.get(idkey)`, updating the LADM chart in the bottom-right corner of the system.
  - The `legal_info_list.json` file constructs a mapping named `IDmap`, where "Apartment ID" or "Name" is used as the key and the id as the value.
  - Using `IDmap.get(idkey)`, the id is retrieved. If `LA_RRR_id` contains `LA_RestrictionType`, the `G6Data.getBankProp(idkey)` method is called to update the relevant LADM chart. If `Party_name` is "Association of owners," the `G6Data.getPublic()` method is called to update the LADM chart for public areas. Otherwise, the `G6Data.getPeopleProp(idkey` method is called to update the LADM chart for individual.

6.2 Future work

- **Dealing with Real-World Legal Data Complexity** The legal data used in this study is hypothetical and simplified for demonstration purposes. In the real world, legal data will be much more complex. Future research should address the challenges of real-world legal data.
- **Management of Geometry Data in the Database** While this study integrates legal data with 3D models, future work could explore the inclusion of graphic data (e.g., BIM models) directly within the database.
- **Querying Individual Structural Elements** In this study, the functionality to query private and public spaces is provided, but it does not include the capability to query individual structural elements, such as walls, columns, or floors. This design choice complies with Dutch legal requirements, which do not necessitate the registration of individual structural components within the Land Administration system. Future work could investigate incorporating methods for accurately representing and querying individual structural components.
- **Different Types of 3D Legal Spaces** Explore different types of 3D legal spaces, such as air space rights, underground utility networks, and mining concessions. This can be realized by defining new data models and visualization techniques that accurately represent these complex legal spaces, and integrating them into the existing system.

- **Integration of Valuation and Spatial Plans** Extend the system to include valuation and spatial plans in 3D, in accordance with the new LADM parts 4 and 5. This can be done by exploring valuation models, developing tools for creating and managing spatial plans, and ensuring that these new features comply with the updated LADM standards.

Bibliography

Alattas, A., Kalogianni, E., Alzahrani, T., Zlatanova, S., and van Oosterom, P. (2021). Mapping private, common, and exclusive common spaces in buildings from BIM/IFC to LADM. a case study from saudi arabia. *Land Use Policy*, 104:105355.

Andritsou, D., Alexiou, C., and Potsiou, C. (2023). BIM, 3D cadastral data and AI for urban management - a case study for energy consumption monitoring. *Journal of Urban Management*, 10:1–15.

Atazadeh, B., Kalantari, M., Rajabifard, A., and Ho, S. (2017). Modelling building ownership boundaries within BIM environment: A case study in victoria, australia. *Computers, environment and urban systems*, 61:24–38.

Boschert, S. and Rosen, R. (2016). Digital twin—the simulation aspect. *Mechatronic futures: Challenges and solutions for mechatronic systems and their designers*, pages 59–74.

Broekhuizen, M., Kalogianni, E., and van Oosterom, P. (2021). BIM models as input for 3D land administration systems for apartment registration.

Cemellini, B., van Oosterom, P., Thompson, R., and de Vries, M. (2020). Design, development and usability testing of an LADM compliant 3D cadastral prototype system. *Land use policy*, 98:104418.

Fuller, A., Fan, Z., Day, C., and Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE access*, 8:108952–108971.

Guler, D. (2022). 3D registration of apartment rights using BIM/IFC: Comparing the cases of the netherlands, saudi arabia, and turkey. In *Proceedings of the XXVII FIG Congress 2022*, pages 1–21. FIG.

Hagemans, E., Unger, E.-M., Soffers, P., Wortel, T., and Lemmen, C. (2022). The new, LADM inspired, data model of the dutch cadastral map. *Land Use Policy*, 117:106074.

International Federation of Surveyors (FIG) (1998). *The FIG Statement on the Cadastre*. FIG Bureau, Canberra, Australia, 11 edition. Published in English.

Kalogianni, E., van Oosterom, P., Dimopoulou, E., and Lemmen, C. (2020). 3D land administration: A review and a future vision in the context of the spatial development lifecycle. *ISPRS international journal of geo-information*, 9(2):107.

Keysers, J. (2015). *Review of digital globes 2015*. Australia and New Zealand Cooperative Research Centre for Spatial Information.

Lehtola, V. V., Koeva, M., Elberink, S. O., Raposo, P., Virtanen, J.-P., Vahdatikhaki, F., and Borsci, S. (2022). Digital twin of a city: Review of technology serving city needs. *International Journal of Applied Earth Observation and Geoinformation*, 114:102915.

Lemmen, C. (2012). A domain model for land administration.

Lemmen, C., da Silva Mano, A., and Chipofya, M. (2022). LADM in the classroom - making the land administration domain model accessible. In *FIG Congress 2022: Volunteering for the future = Geospatial excellence for a better living*.

Lemmen, C., Van Oosterom, P., and Bennett, R. (2015). The land administration domain model. *Land use policy*, 49:535–545.

Qi, Q. and Tao, F. (2018). Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *Ieee Access*, 6:3585–3593.

Schrotter, G. and Hürzeler, C. (2020). The digital twin of the city of zurich for urban planning. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 88(1):99–112.

Shafto, M., Conroy, M., Doyle, R., Glaessgen, E., Kemp, C., LeMoigne, J., and Wang, L. (2010). Technology area 11: Modeling, simulation, information technology and processing roadmap. *NASA Office of Chief Technologist*.

Shojaei, D., Kalantari, M., Bishop, I. D., Rajabifard, A., and Aien, A. (2013). Visualization requirements for 3D cadastral systems. *Computers, Environment and Urban Systems*, 41:39–54.

Stoter, J., Ploeger, H., Roes, R., van der Riet, E., Biljecki, F., and Ledoux, H. (2016). First 3D cadastral registration of multi-level ownerships rights in the netherlands. In *Proceedings of the 5th International FIG 3D Cadastre Workshop, Athens, Greece*, pages 18–20.

Stoter, J., Ploeger, H., Roes, R., van der Riet, E., Biljecki, F., Ledoux, H., Kok, D., and Kim, S. (2017). Registration of multi-level property rights in 3D in the netherlands: Two cases and next steps in further implementation. *ISPRS International Journal of Geo-Information*, 6(6):158.

Stoter, J. E. (2004). 3D cadastre. *PhD Thesis, TU Delft*.

Vandysheva, N., Ivanov, A., Pakhomov, S., Spiering, B., Stoter, J., Zlatanova, S., and Van Oosterom, P. (2011). Design of the 3D cadastre model and development of the prototype in the russian federation. In *Proceedings of the 2nd International Workshop on 3D Cadastres, Delft, The Netherlands*, pages 16–18.

Ying, S., Guo, R., Li, L., and He, B. (2012). Application of 3D gis to 3D cadastre in urban environment.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.