An Interoperable Web-based Application for 3D City Modelling and Analysis

Ziya Usta (ziyausta@artvin.edu.tr)
Artvin Coruh University

Çetin Cömert
Karadeniz Technical University

Alper Tunga Akın
Karadeniz Technical University

Research Article

Keywords: 3D Visualization, 3D Modelling, Web GIS, Geospatial, City Model, Open Standards

Posted Date: September 4th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3302763/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Earth Science Informatics on November 23rd, 2023. See the published version at https://doi.org/10.1007/s12145-023-01167-5.

An Interoperable Web-based Application for 3D City Modelling and Analysis

Ziya Usta[1*], Çetin Cömert[2†] and Alper Tunga Akın[2†]

[1*]Department of Geomatics Engineering, Artvin Çoruh University, Faculty of Engineering, Artvin, 08100, Türkiye.
[2]Department of Geomatics Engineering, Karadeniz Technical University, Faculty of Engineering, Trabzon, 61080, Türkiye.

*Corresponding author(s). E-mail(s): ziyausta@artvin.edu.tr;
Contributing authors: ccomert@ktu.edu.tr; alpertunga@ktu.edu.tr;
†These authors contributed equally to this work.

Abstract

This study presents a web-based application developed for the efficient management of large 3D city models (3DCMs). The application offers functionalities such as 3D tiling, web-based visualization, 3DCM generation from 2D data, and 3D spatial analysis. It complies with international standards and utilizes open-source software components. The application includes a 3D tiling tool that implements the OGC 3D Tiles Standard, allowing users to partition their 3DCMs into optimized tiles for storage, transmission, and rendering. A web-based procedural modeling tool enables users to generate 3DCMs from 2D building footprint data without the need for complex modeling languages. A 3D spatial analysis component empowers users to perform comprehensive spatial analyses within the web-based environment. Leveraging modern web technologies facilitates precise and efficient spatial computations. To provide standardized visualization, the application features a viewer and client application built using HTML5 and WebGL technologies. This viewer adheres closely to the 3D Tiles Standard, enabling immersive navigation and interaction with 3DCMs. By combining these components, this web-based application intuitiveize the management of large 3DCMs, offering enhanced visualization, analysis, and model generation capabilities. It represents a milestone in web-based 3D city model management, prioritizing international standards and open-source principles.

Keywords: 3D Visualization, 3D Modelling, Web GIS, Geospatial, City Model, Open Standards

1 Introduction

The 3-Dimensional City Model (3DCM) is a digital representation of buildings and other urban objects, with their attribute information and 3D geometries. The 3DCM is widely used in many fields such as urban infrastructure management, navigation, simulation, planning, emergency management, urban planning, tourism, and property valuation, through various spatial analysis tools such as silhouette analysis, determination of the solar energy potential of building roofs and facades, and modeling of noise distribution in the city. For a more detailed literature review on the usage areas of 3DCM, one can refer to [1].

On the other hand, with the Service-Oriented Architecture and Web Services paradigm that started in the 2000s, browser-based applications have become increasingly widespread. Thanks to advancements in browser architectures and the increasing number of software libraries such as OpenLayers [2], Leaflet [3], Turf [4], and developments in the "Web Framework" area, web-based Geographic Information System (GIS) applications have started to replace desktop applications. The display of 3DCMs in browsers and enabling user interaction has also become an interesting topic in this context [5],[6].

With the use of HTML5 and WebGL[7] together, unlike applications before these technologies, many web-based applications have been developed that are platform and browser independent and do not require any plugin installation on the client side. In a similar vein, visualizing 3DCMs over the web has become an essential topic in the field of web-based GIS. Additionally, many virtual globe applications have been developed [8]. Some of these include Cesium [9], WebGL Earth [10], and Open Web Globe [11].

Web-based 3D GIS is a relatively new field and has only been studied in a limited number of studies in the literature, requiring further research on various issues. For example, there are very few studies in the literature on 3D spatial analysis (3DSA) in the web-based 3D GIS context. The capabilities of related software for 3DSA are also quite limited. One reason for this is that research and related software in the literature have generally focused on 3D visualization. On the other hand, in the vast majority of 3DSA implementations, the analysis and visualization functions have only been executed on the visualization side as web-based.

When dealing with large-scale geospatial 3D data such as 3DCMs in web environments, a significant challenge is also the need to work with smaller data chunks due to memory and traffic limitations imposed by browsers and networking protocols. As a result, algorithms must be able to identify and manage affected chunks during visualization or analysis. Working with smaller chunks, or "tiles," presents several management difficulties, such as maintaining object integrity across tile borders and updating tile contents when data changes occur. If 3DSA and queries are in question, a key consideration is whether they are carried out in an "online" or "offline" mode. In the offline mode, the analyses are performed on desktop 3D software and the results are transmitted to the client for visualization. This is the typical approach used by so-called "3D web GIS" implementations. However, this approach can be cumbersome when users want to modify analysis parameters, as they need to re-perform the analysis on their desktop and repeat subsequent steps. Furthermore, users must work with two different software applications. This is not in line with the "true" web-based

operational mode, where all user interactions take place within a single web-based software. By contrast, performing analyses using the same web-based software in an online mode allows for a more seamless user experience. To achieve this, mechanisms for organizing and tiling data must be implemented, as demonstrated in this study.

The major issue involves choosing between a multiscale or multi-representational approach while managing varying levels of detail. In the multiscale mode, data is typically organized into different tilesets for each level of detail. In contrast, the multi-representation approach can handle different levels of detail within a single tileset. Each approach has its own advantages and disadvantages, which will be explored in greater detail later in this study. However, it is worth noting that a comprehensive discussion of this topic is currently lacking in the literature. When it comes to creating 3D city models, another key challenge is the shortage of open-source components. As far as we are aware, there is only one open-source tool available from Delft University. Other commonly used tools, such as the CGA (Computer Generated Architecture) component of ESRI CityEngine, are currently not web-based and are commercial products. In this study, a web-based component has been developed in the form of a REST web service to address this issue.

The interoperability is also a bottleneck and its problem refers to the neglecting of this issue in most of the related work. The way of developing interoperable applications is to employ standards. One problem here is the immaturity of the 3D geospatial data standards. The immaturity is due to either their definition or the limited number of implementations. Interoperability is an area that spatial data management communities have been working on for a long time. While it has been largely successful in 2D spatial data management, there are issues on the 3D side of the subject. Proprietary solutions developed over time have made interoperability difficult. Therefore, instead of producing proprietary solutions in this study, international standards were used, and for this purpose, OGC's 3D Tiles standard [12] was used. The implementation of the standard in this study is one of the limited number of implementations worldwide. The experience gained and the findings obtained in the implementation of this standard, which is quite new and whose standardization process was completed in 2018, are of great importance in terms of the maturity process of the standard. The relevant findings of this study, which will be mentioned in Section 4, are valuable in this regard.

In this study, the necessary algorithms for a series of 3DSA operations were developed. The web-based implementation of 3DSA operations is not only required in various other applications but also specifically needed in the background of 3D collaborative, interoperable software. The use of such software in various fields, such as Building Information Modeling (BIM), has the potential to be utilized including building inspections.

Consequently, this study aims to develop a web-based application to manage 3DCMs. The management of 3DCMs refers to the creation, analysis and visualization of the 3DCMs. In this context, one of the objectives is to develop a tiling component for displaying 3DCMs on browsers. Another objective is to develop a web-based 3D procedural modeling component. Hence, users can produce 3DCMs from 2D data without any software installation. Another important objective is to develop a 3DSA component by designing and implementing a 3D intersect algorithm to be able to carry

out spatial analyses online. Entire components were developed considering relevant standards to ensure interoperability.

The rest of the paper is organized as follows. In Section 1.1, the related studies are discussed and differences of ours from them are put forward. In Section 2, the development pipeline of the application is described. In Section 3, the user interface of the application and runtime metrics of the 3DSA component are represented. In Section 4, encountered issues or limitations during the development of the application are discussed. Finally, Section 5 concludes the study and states the planned future work.

1.1 Related Work

In this section, we have briefly reviewed the current state of the 3DCM context in three parts which are the three main contributions of this study. The first part discusses the state of the tiling and visualization studies. The second part discusses procedural modelling studies and the last one gives information about web-based 3DSA studies.

Throughout this brief review, we discussed the development of 3DCMs and the challenges associated with creating a comprehensive and flexible platform for visualizing and analyzing these models. We inspected various tools and technologies used in the field, such as 3D Tiles and Cesium.js, and the limitations and opportunities they present. The goal of this review was to provide insights into the current state of 3D urban modelling and to highlight areas where further research and innovation are needed to fully realize the potential of this field.

1.1.1 Tiling and visualization

The first studies in the field of web-based 3D applications were conducted before the development of HTML5 and WebGL technologies, so naturally, these technologies were not utilized. The software components used in these studies, such as Flash, Silverlight, and Java3D, require platform dependency and additional software component installation on the client. Examples of these studies include GeoPortail [13], 3D Macau [14], and Open3DGIS (O3DG) [15]. While GeoPortail and 3D Macau require the installation of Terra Explorer software as a 3D viewer, O3DG requires the installation of a browser plug-in.

With the development of Internet technologies such as HTML5 and WebGL, browser-independent 3D WebGIS applications have begun to be developed. Initially, many virtual globe applications were developed [8], including Cesium [9], WebGL Earth [10], and Open Web Globe [11]. In addition to virtual globes, many studies have been conducted for the visualization of 3DCMs over the web. 3DCMs have been displayed in browsers without any plug-ins using X3DOM [16] and WebGL [5]. In another study, the Java-based NASA World Wind virtual globe has been extended to display 3DCMs [17], which requires the installation of NASA World Wind on the user's computer.

In the study conducted by Mao and Ban [18], 3D urban models, including buildings and streets, were transformed into Java classes and visualized using X3DOM. Prieto et al. [19] expanded on this work by using 3D web services, such as W3DS, instead

of Java classes to visualize the 3D urban models in a browser. However, these studies used small-sized data and did not provide a tiling solution.

Gesquiere and Manin [20] developed a server-client architecture to display CityGML data in a browser. In their work, the 3D urban models in the CityGML format were divided into regular tiles (Fig.1). The tiles were requested from the server and displayed depending on the camera parameters. The biggest disadvantage of this application is the regular tiling. Objects in 3D urban models are distributed heterogeneously, and as a result of regular tiling, some tiles have many more objects than others. Therefore, the sizes of the tiles are different from each other, and the application cannot control the data size of the tiles. In addition, objects can intersect with multiple tiles, which makes object management difficult and compromises object integrity. The study did not provide any solution to this issue.

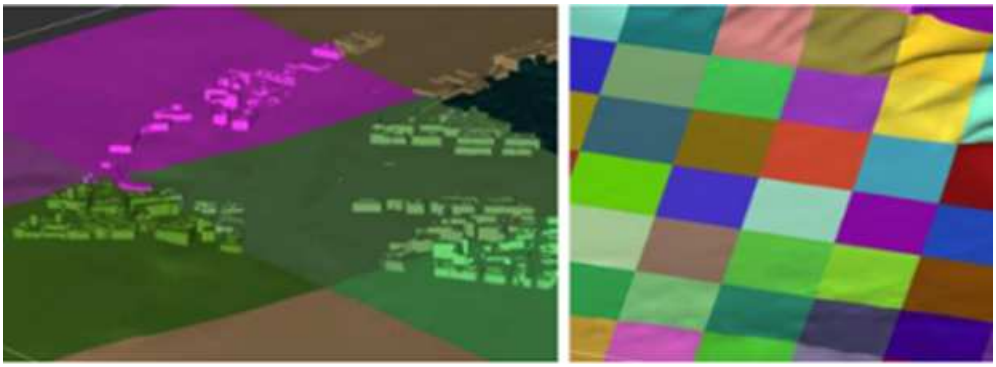

Fig. 1 Regular tiling, tile borders and buildings [20].

Prandi et al. [17] developed several smart city services requiring tiling. In this study, regular tiling was also performed and had the same deficiencies as [20]. Chaturvedi et al. [21] developed a web client for the 3DCityDB software component, allowing CityGML data to be viewed in a browser. In this study, regular tiling was also performed. Koukofikis et al. [22] achieved tiling using hierarchical data structures. They used 3D Tiles and 3D Portrayal Service standards in their work. In this study, tiling was performed using commercial software such as Cesium ion and FME. In such a workflow, these software programs that do not work integrated require many user interventions and do not work as web-based. The user works in multiple environments.

Gaillard et al. [23] developed a method that supports multiscale resolution for visualizing 3D city models using the 3D Tiles standard. In this study, the partitioning process was performed hierarchically using road network data, where buildings adjacent to main roads were placed in upper tiles, while buildings adjacent to side roads were placed in lower tiles. With this type of partitioning, the partitions were not based on data density, resulting in partitions with heterogeneous data sizes. Additionally, the maximum data density of the partitions cannot be controlled. Lu et al. [24] developed a web-based real-time visualization of large-scale meteorological data

in their study. For this purpose, the point cloud data was partitioned according to the 3D Tiles standard using the Octree data structure. This study only involves the partitioning of point cloud data and does not cover 3D city models.

Xu et al. [25] developed a method for partitioning BIM models using the 3D Tiles standard. This study partitioned BIM models and did not include 3D city models. Additionally, an automated workflow could not be provided in the study, and different software components, such as IFC Shell, were used for different processing steps, and the solution was not web-based. Jaillot et al. [26] expanded the 3D Tiles standard to enable the visualization of temporal changes, such as buildings that have been demolished or newly constructed, in a 3D city model displayed in a web browser. They used the py3dtiles open-source Python library for the tiling process in this study [29]. One limitation of this library for their developed component is that it partitions based on the number of objects, without taking into account the varying data dimensions of the objects. Additionally, py3dtiles is not a web-based component, as the focus of this study was on representing temporal changes rather than data management. This study is much narrower in scope than what is intended to be accomplished.

When examining relevant software components outside of academic studies, it can also be seen that there are a limited number of implementations available. One of these is "obj23dtiles" [27], an open-source Node.js module. Its biggest shortcoming is that it does not perform partitioning, only converting an entire city model in obj. format to a single b3dm file, which is the data format of 3D Tiles. This is only one of the many components implemented in this study, and even at the component level, it has its own drawbacks.

The open-source Node.js module "citygml-to-3d tiles" [28] is another tool that converts CityGML format data to a single b3dm file without any partitioning. This is again a very small component among the tools implemented in this study.

In this regard, the most advanced open-source component is "py3dtiles". Py3dtiles divides CityGML data hierarchically according to the 3D Tiles standard. Its biggest drawback is that it is not a web-based component. In addition, an important shortcoming is that it determines the highest data volume limit in terms of the number of objects in the tiling. As explained in the relevant section below, in the tiling software developed in this study, the data volumes of the geometries that make up the objects are taken as the basis, thus avoiding the mistake of treating a very high-volume building and a building with the opposite case as equivalent objects. On the other hand, like the other software components mentioned above, this component is only one of the components developed in this study.

A study conducted by TU Delft used the building and address dataset (BAG) [30] of the Netherlands to produce a nationwide 3DCM, which is presented as open data on the 3D BAG platform (URL-12) using the 3D Tiles standard. The platform serves as a 3D viewer and allows users to view the downloaded data, but it does not perform any tiling, and users cannot tile their own data; they can only view the 3D BAG dataset. As such, this study's contribution is a very small subset of the functions developed in the project's work packages. In another study conducted by TU Delft, a browser-based viewer for the CityJSON 3D building model format has been developed [31]. This application does not perform any tiling but only allows the imported dataset to be viewed in a web-based interface.

1.1.2 Procedural Modeling

The procedural modelling (PM) approach is used to generate 3DCMs in this component [32], [33]. PM is an umbrella term which includes many modelling techniques such as L-Systems, fractals and shape grammars. All of these techniques aim to create multiple instance models by utilizing a set of predefined rules and algorithms. The batch modelling approach of PM which minimizes user interaction and labour, fits very well into GIS applications which deal with the management of big data. The technique of the extrusion from 2D footprints, which is a widely used method [34],[35], is adopted to model 3DCM procedurally in our PM component.

In the literature, Esri CityEngine and RANDOM3DCITY [36] are notable examples of procedural modelling software. Esri CityEngine is a widely recognized software tool used for the generation of 3D urban environments. It allows users to create detailed 3D models of cities and urban landscapes using procedural modeling techniques. CityEngine performs rule-based modelling using the CGA (Computer Generated Architecture) language. On the other hand, RANDOM3DCITY is an open-source procedural modeling software component developed to fulfil the need for 3D city models at different levels of detail. As the name implies, it generates random, or in other words, synthetic data. The developed software component in this study, unlike CityEngine and RANDOM3DCITY, can work in a web-based environment and generate real-world data sets.

This component has other advantages as well. One of them is that the user doesn't have to learn a new language like CGA and can perform modeling entirely through an interactive interface. On the other hand, the modelling capabilities are not limited to the CGA language and the commercial software component provided by Esri CityEngine. In contrast, the component in this study is open source, allowing for capability extensions to be made without being dependent on a commercial company. Below is an example of a simple extrusion operation in the CGA language. Based on this, the user needs to understand the syntax of this scripting language, express 2D objects using the "Lot" keyword, and know and correctly express the direction and magnitude of the extrusion process.

"Lot →extrude(world.z, 10)"

1.1.3 Web-based 3D Spatial Analyses

Many existing works entitled 3D WebGIS were about the visualization of the 3D analysis on a client application, rather than performing 3D analysis itself online. The common point of the aforementioned works in this paragraph is that although they are named 3D WebGIS they do not perform 3D spatial analysis online. Van Ackere et al. [37] visualized the results of a flood analysis in 3D. In another work entitled 3D WebGIS Feng et al. [38] visualized terrains in 3D. Similarly, Chen et al. [39] developed a framework for 2D and 3D visualization of geospatial data to manage landslide hazards. Xiaoqing et al [40] visualized the results of the shortest path analysis in 3D by integrating ArcGIS and Skyline software. Li et al. [41] visualized earthquakes on a globe in 3D. Von Schwerin et al. [42] developed a VR application in which you can interact with 3D models. With this tool, users can search and query, in real-time

via a virtual reality (VR) environment, segmented 3D models of multiple resolutions that are linked to attribute data stored in a spatial database but cannot perform 3D spatial analysis. Pispidikis et al. [43] developed a web-based tool to query and visualize CityGML data. Although they mentioned 3D spatial analysis in their article, which spatial analysis can be performed is not explained and also, results of the work do not show 3D spatial analysis. They mostly focused on querying the CityGML data and visualizing queried results.

The most similar works to our proposed methodology are [8] and [44]. [8] developed a web-based tool to perform 3D buffer and 3D intersect analyses. The drawbacks of this work are that analyses are performed on the client using not on the server. Hence, when the analyzed 3D geospatial data exceeds the memory of the browser, the application does not work. Application is highly dependent on the size of the input data. Another limitation is that, in 3D intersection analysis, the application cannot calculate intersection points for partially intersected objects. As can be seen from the images of the results, partially intersected buildings are considered to be completely inside the intersected object, in that case, the sphere. In their work, [44] developed a 3D WebGIS application that performs 3D line-of-sight analysis on the browser. The limitation is that analysis is performed on the client as in Chaturvedi, hence application is highly dependent on the size of the input data.

2 Methodology

We have divided the methodology explanation into three parts which explain the three main components of the application. The first component is the tiling component that manages the decomposition of the 3DCM for streaming and visualization. The procedural modeling component generates 3DCMs from 2D building footprints provided by the user. The last component executes 3D spatial analyses on client requests and calculates the result volumes of these analyses to visualize.

2.1 Tiling Component and Web-Based 3D Display

Since a 3DCM is huge in size for streaming and visualization directly in browsers, the data is decomposed into multiple tiles. As needed, relevant tiles are sent over the network and processed by clients. Streaming and displaying only the most relevant data decreases the amount of the data transmitted and improves performance notably.

2.1.1 Decomposition of 3DCM

One of the requirements for tiling is choosing the decomposition method. The most basic method is to decompose regular fixed-sized rectangular tiles. This method does not represent data density in a heterogeneous distributed 3DCM case which creates heterogenous-sized tiles. Therefore, hierarchical tiling methods, such as R-Tree [45] or Quadtree [46], could be more appropriate to decompose heterogenous 3DCM data. Because the decomposition is done according to a data density threshold, a tile size, which generates tiles in homogenous sizes.

Tile size determines the amount of the data to be transmitted from server to client for a tile and directly affects the render quality. A client should be able to render and

display the contents of a tile without any latency and without degrading the rendering performance. Rendering performance is measured as frame per second (fps) which is the number of rendered images per second. In the game industry and computer graphics, 60 fps is accepted as a "good" rendering performance and 60 fps value is selected as the minimum acceptable rendering performance while determining the tile size in this study. Since rendering is done on the client, rendering performance highly depends on the client's hardware. In order to establish a balance between rendering performance and the tile size, data threshold tests are performed using a computer that has a low-end GPU (Graphical Processing Unit). The reason for choosing a computer with a low-end GPU is to guarantee 60fps even with such computers. Table 1 shows the specs of the test machine. 180 KBs is determined as the maximum tile size which is compliant with 60 fps according to empirical tests. The R-Tree data structure is used to decompose 3DCM into tiles regarding the pre-determined tile size threshold.

Table 1 System Specifications of the Test Machine

CPU	Intel i5 1.8Mhz Turbo Boost CPU
RAM	4GB DDR3 RAM
GPU	Intel HD Graphics 4000 GPU

2.1.2 Implementation of the 3D Tiles Specification

3D Tiles is an OGC standard developed for efficiently streaming and displaying extensive 3D geospatial data. It introduces a hierarchical data structure and a range of tile formats that enable the delivery of visualizable content. Notably, 3D Tiles does not impose strict guidelines for how the data should be visualized as long as the tileset is a hierarchical tree, allowing clients to interpret and display the 3D Tiles data in their preferred manner [12].

In 3D Tiles, hierarchical information and metadata about tiles are encoded to a JSON file called "tileset.json". Thus, the tileset.json file is consumed by the client at runtime and the hierarchy is derived for the visualization pipeline of the application. Fig.2 shows the UML class diagram for a tileset in 3D Tiles.

For each tile, the data of the tile has been written into a single b3dm file which is a 3D format of the 3D Tiles standard based on glTF. By batching multiple 3D models into a single file multiple 3D models can be transmitted with a single request and in the visualization pipeline, they can be rendered with the least number of WebGL draw calls.

WebGL uses points, lines and triangles as geometric primitives to render objects. Thus, 3D polygonal surfaces of the objects in 3D geospatial data must be triangulated to display them via WebGL. For this purpose, a 3D polygon triangulation algorithm Ear-Clipping is implemented using the earcut4j open-source Java library (Fig.3).

Most of the 3D geospatial dataset is not aligned to a digital terrain model (DTM) or has pre-calculated heights according to a different DTM. In such situations height differences may occur between 3D city models and DTMs hence, the heights of the city objects must be aligned according to the DTM. To clamp buildings to the terrain,

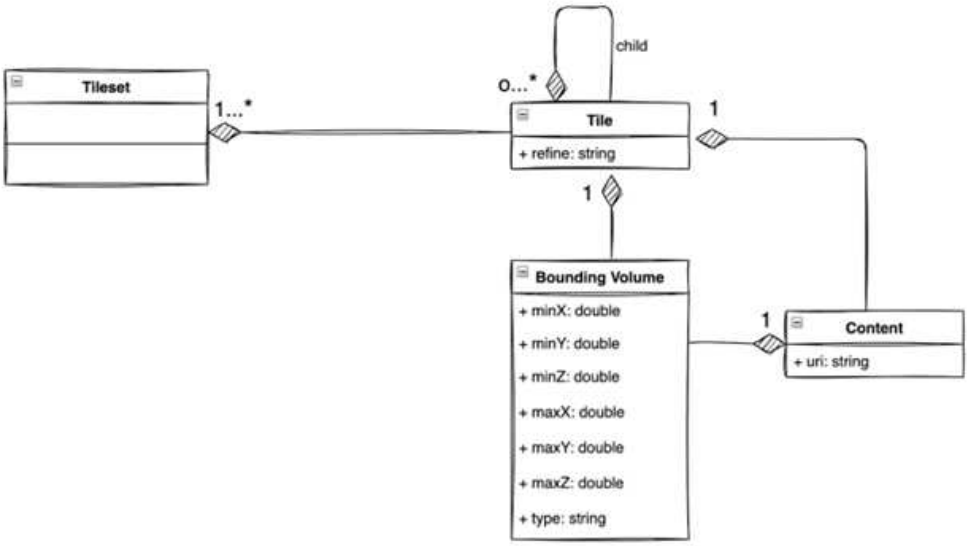

Fig. 2 UML Class Diagram for A Tileset in 3D Tiles

Fig. 3 3D polygon surfaces (left), triangulated polygons (right).

first, building footprints intersected with DTM then, an offset along the z-axis is applied until the min z value of the building footprint matches the min z value of the intersected pixels. Thus, buildings have been clamped to the terrain during the tiling (Fig.4).

2.1.3 Developing RESTful Web Services for Tiling Component

The tiling component has been implemented through RESTful web services. Client and server communicate with each other through web services developed by using the Java Jersey rest framework. The overview of the component can be seen in Fig.5.

Fig. 4 Buildings over the terrain (top), buildings clamped to the terrain (bottom)

Deployment of the component is done using Amazon Lambda Serverless. Serverless stands for not needing your own servers to execute the application on a server and Amazon Lambda is a serverless platform provided by Amazon Web Services (AWS). End users can upload 3D geospatial data in the CityGML format to the server. In the server, the uploaded file is stored in an Amazon S3 bucket. S3 stands for simple storage service and the bucket is a data container for uploaded objects in Amazon S3. Amazon lambda functions have been used to process the uploaded file in the S3 bucket. Using the citygml4j open-source java library CityGML file is parsed and using the earcut4j open-source java library 3D polygons of the city objects are triangulated. Then, using our 3D tilers based on R-Tree, the 3D tileset is generated in accordance with the OGC 3D Tiles standard. Then, the end user can download the 3D tileset to its local machine or display the generated tileset using the client of the application. For developing the client, Cesium.js, HTML5, WebGL and Prime Faces technologies have been used. Cesium.js has been used to render 3D tilesets and Prime Faces which is an open-source Java server pages library has been used to develop the user interface of the client.

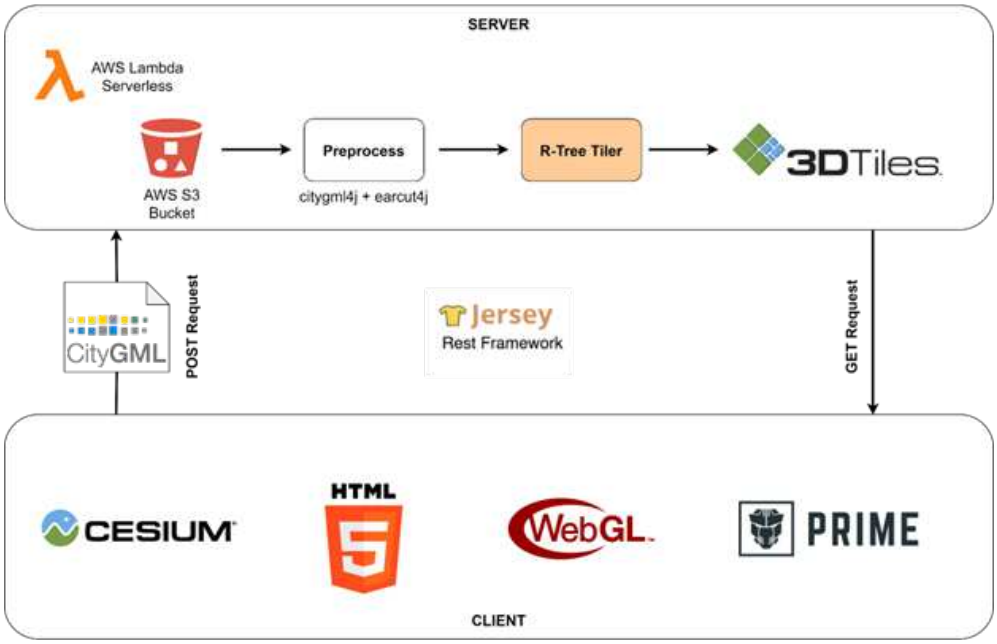

Fig. 5 The overview of the tiling component.

2.2 Procedural Modelling Component

As mentioned in Section 1.1.2., the 3DCMs are generated procedurally by extrusion of the 2D building footprint through the surface normals using the PM component. For extrusion, surface normals of the building footprints need to be calculated because these normals give the extrusion direction. The normal vector is a unit vector perpendicular to the surface (Fig.6). The surface normal vector (N) is calculated as the cross-product of the vectors of

$$|P2 - P1|$$

and

$$|P3 - P2|$$

.

After the calculation of surface normals for each building footprint, using the "height" attribute of the building footprints in metric units, building footprints are translated along surface normals and top surfaces are constructed. By indexing the top and footprint points, vertical surfaces are constructed and the LOD1 block model is derived (Fig.7).

If condominium unit plans (CUPs), which are demonstrating 2D borders of the condominiums, are provided, our modeller can also model condominium units as well at the city scale. For this purpose, each condominium is translated along the surface normal according to the height of the condominium in CUP. Then translated polygons are extruded and condominiums are modelled procedurally (Fig.8).

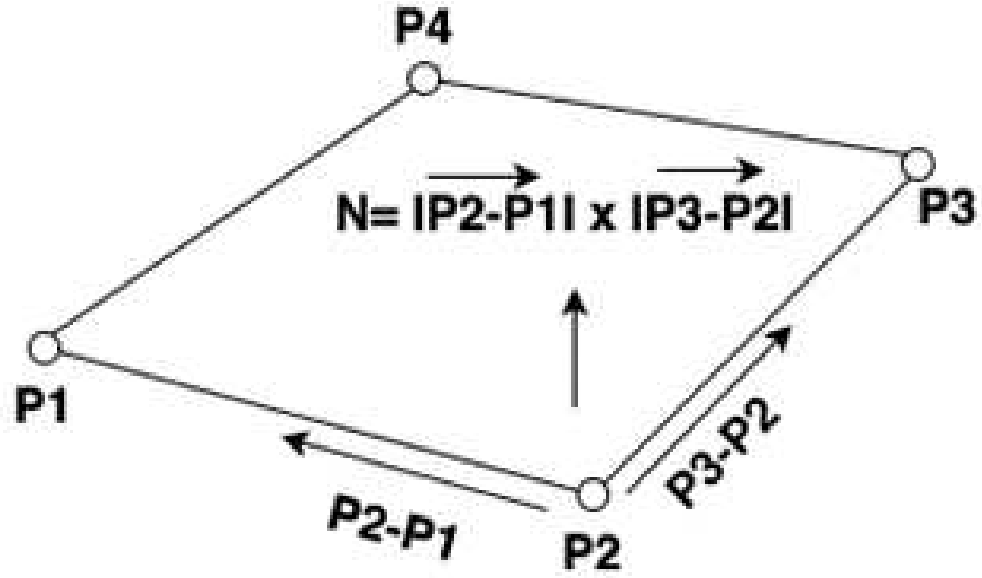

Fig. 6 Calculation of the surface normal[49]

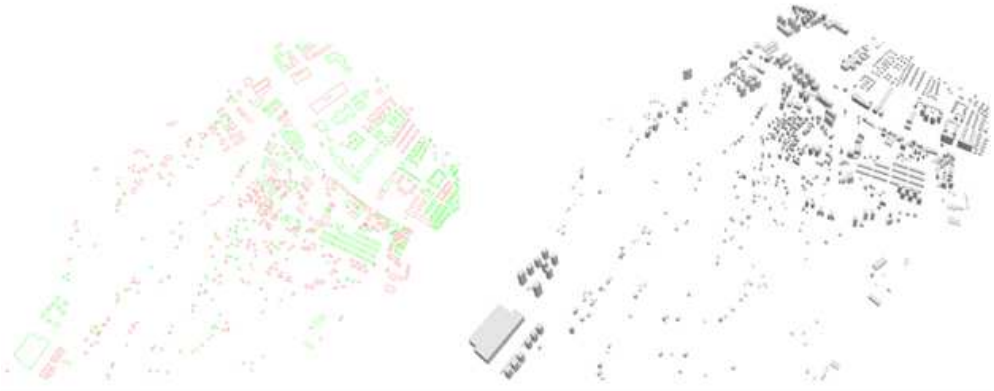

Fig. 7 2D building footprints (left) 3D block model (right).[49]

Geometries of the condominium units are stored as CityJSON files in our component. Since condominium units are not defined in the CityGML data model, an ADE is developed for storing condominium units (Fig.9).

The overview of the PM component is demonstrated in Fig.10. If the client sends 2D building footprint data with a modelling request to the server, the PM component produces a 3DCM and sends it to the tiling component for visualization.

Fig. 8 Polygons (left) multiplied and translated (middle) condominium unit in 3D.[49]

2.3 3D Spatial Analysis Component

3DSA component consists of three different types of analyses which are 3D Intersect, 3D Clip, and 3D Difference. These 3D GIS analyses are referred as '3D Boolean Operations' in the computer graphics domain. All three types of analyses are based on a 3D intersection algorithm implemented in this study.

The 3D intersection algorithm consists of two parts. The first part is to detect whether there is an intersection between two 3D objects and the second part is to construct the intersected section as a 3D polygonal mesh. To determine the collision between objects, a plane is swept through objects along three main axes. Vertices of the objects are stored in an array and as they intersect the surface while the plane is being swept, they are removed from the array. If the surface intersects with the vertices of the second object before the vertices of the first object are completely removed, there is a possible intersection, and this part of the algorithm returns a "true" Boolean value (Fig.11). This operation needs to be returned as "true" for all three main axes to detect an exact intersection. If the operation returns "false" for one of the axes, it means there is no complete intersection between objects. This part of the 3D intersection algorithm is a necessity for the acceleration of the algorithm without traversing the entire dataset.

The Cork open-source library was selected for generating the volumes of the detected intersections. Cork offers a robust 3D intersection functionality that effectively handles numerical errors resulting from floating point arithmetic, all within a user-friendly interface. This reliability is achieved through the use of a fixed-precision approach [48]. Cork utilizes the Nef polyhedra representation [47]. The fundamental concept behind using Nef polyhedra for Boolean operations is to represent each object as a Nef polyhedron, which stores neighbourhood information around each vertex. The Nef polyhedron data structure consists of faces and half-edges that connect these faces. A key feature of the Nef polyhedron is its ability to represent both the interior and exterior of the object it represents. When performing a Boolean operation on two Nef polyhedra, their faces and half-edges are combined to create a new Nef polyhedron. This process involves identifying the intersection points of the half-edges and faces of the two polyhedra and generating new half-edges and faces. The resulting Nef polyhedron accurately represents the Boolean operation performed on the original objects (Fig.12).

Fig. 9 3D condominiums stored in the CityJSON file.

The other types of boolean operations, which are the 3D difference and 3D clip, are based on the calculation of the intersection, as mentioned above. A demonstration of these three operations, or 3D spatial analyses, is given in Fig.13. The overview of the 3DSA component of the application can be found in Fig.14. The clip operation

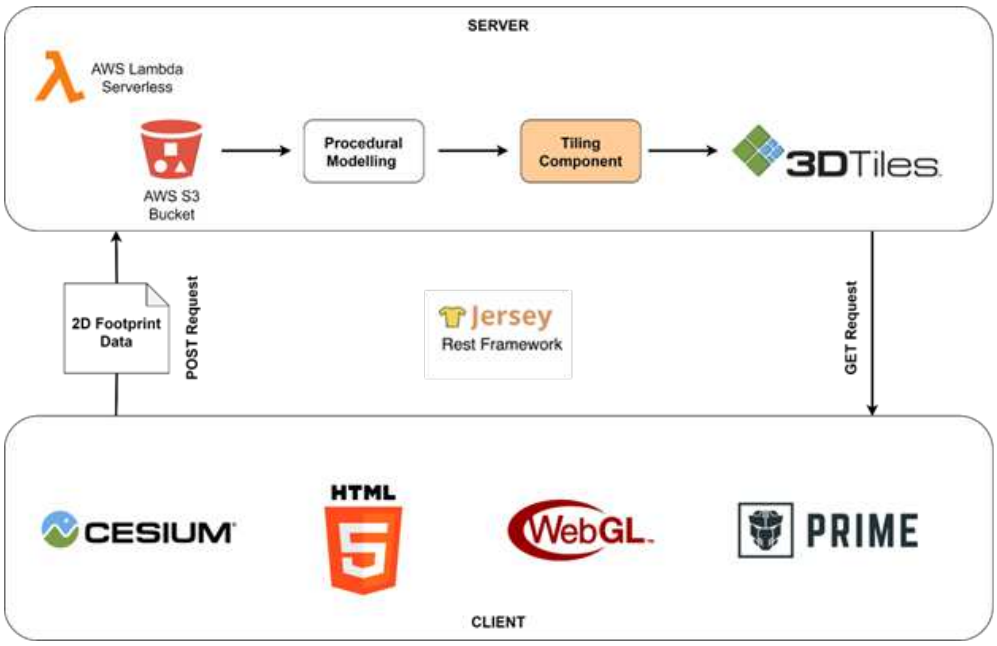

Fig. 10 The overview of the PM component.

occurs the same as the intersect geometrically, but the attributes of the outputs only come from object A in the relevant figure. In the intersection operation, the output includes the attributes of both objects.

3 Results

As a result of the mentioned efforts, a novel web-based software application, which implements the OGC 3D Tiles standard, and provides the modelling of 3DCMs procedurally and 3DSA, is developed. The final application is deployed on AWS and can be accessed through a web interface at http://3dviewer-118y452.s3-website-us-east-1.amazonaws.com/ (Fig.15). Through the interface, the user can convert 2D building data in .shp format into a 3D building model and generate a 3D Tileset dataset on the globe (Fig.16). The 3DSA functionality will be also added to the interface after planned optimizations for the web.

To observe the runtime metrics of the 3DSA, the 3D intersection operation was executed with a test dataset containing 1370 buildings and 21696 vertices. The 3D intersection operation requires two input models: the query model and the search model. Both models were generated procedurally using our component. The query model was extruded by 3 meters, while the search model was extruded by 5 meters. This implies that there will be at least one intersection to detect at each query location, representing the worst-case scenario. To observe the runtime performance, the dataset was partitioned into sequential chunks, each containing a specific number of buildings:

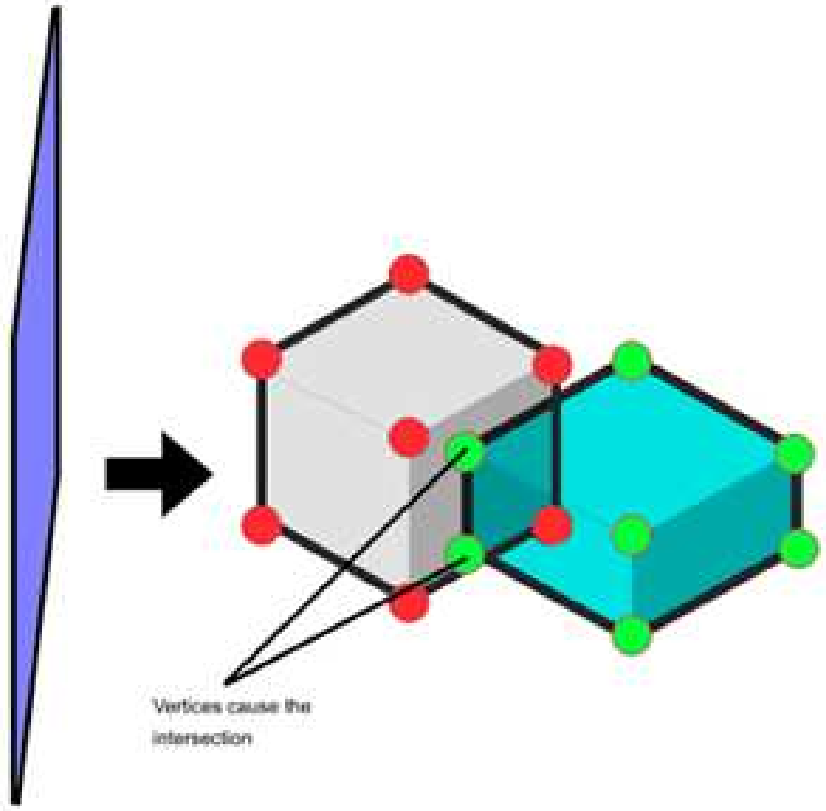

Fig. 11 Sweep Plane and collision detection

200, 400, 600, 800, 1000, 1200, and 1370 buildings. The 3D intersection operation was then executed for each chunk individually. The metrics can be seen in Table 2. These metrics will be used for server-side scaling and to configure the hardware of the cloud server.

Table 2 Runtime metrics of the 3D intersection operation

Object Count	Execution Time (seconds)
200	4,14
400	9,4
600	13,24
800	21,44
1000	29,15
1200	40,82
1370	43,35

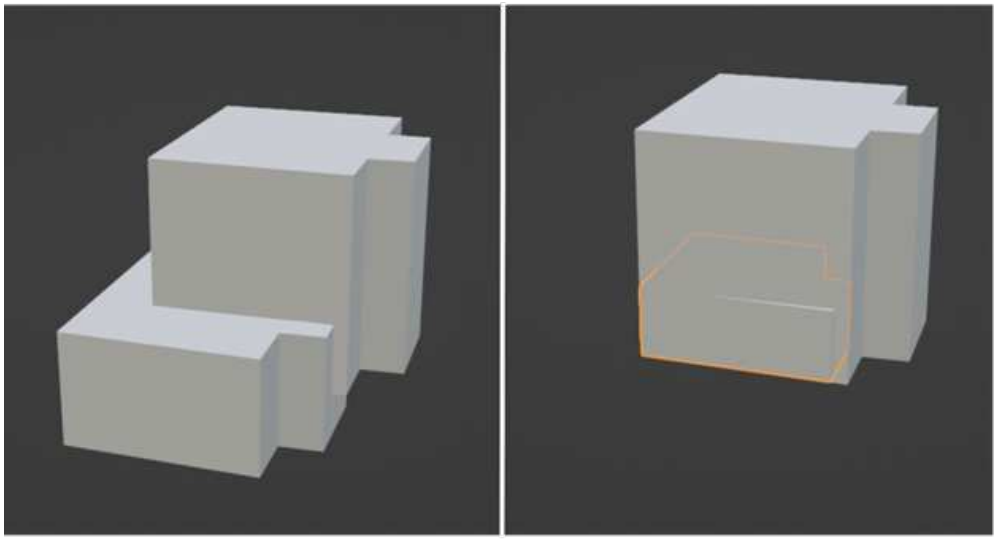

Fig. 12 The intersection volume (right) of two building objects (left) is highlighted with orange colour.

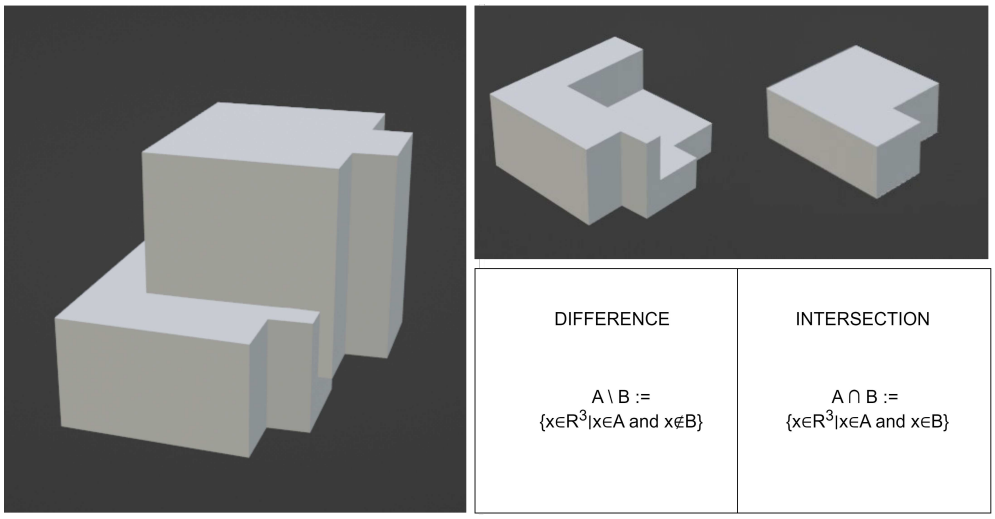

Fig. 13 The demonstration of 3D spatial analyses which are intersection and difference.

4 Findings and Discussion

The findings gained during the development stage of the application are discussed according to the components in this section. Within the frame of the tiling and visualization component, the authors encountered three notable questions. The first one

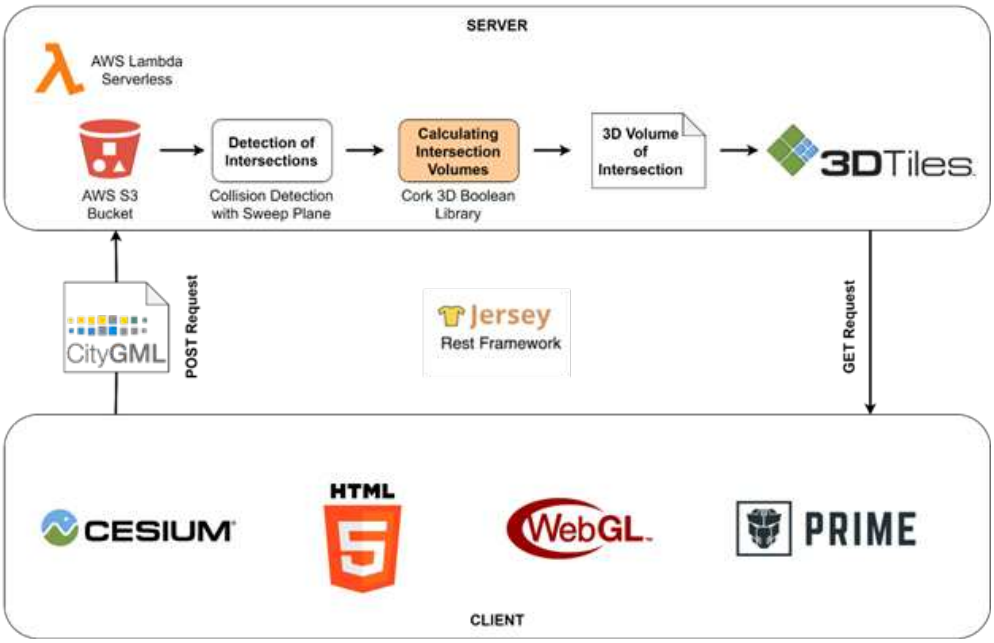

Fig. 14 The overview of the 3DSA component

Fig. 15 The application interface with 3D Tiles Generation Menu and the virtual globe

is the decision on the tiling scheme which can be layer-based or object-based. After generating the tileset, we conducted tests on the render performance during the tile display. It was observed that when zooming or rotating, the render performance dropped below 60 fps. This decrease in performance was attributed to the DTM. Rendering a

Fig. 16 The generated 3DCM on the interface with attributions

textured DTM requires more computational resources compared to rendering untextured 3D buildings. The tile size threshold, which was determined based solely on the 3D objects without considering the terrain, resulted in a drop in rendering performance below 60 fps. To ensure a consistent rendering performance of 60 fps, we took the terrain into consideration when determining the tile size threshold, which was set at 180 KB. The aforementioned situation has sparked a discussion regarding whether tiling should be done on an object-based or layer-based approach. In the case of layer-based tiling, each layer is tiled individually, ensuring that each layer adheres to the tile size threshold and maintains a rendering performance of 60 fps. One advantage of layer-based tiling is that layers can be displayed or hidden individually, providing flexibility in the visualization. However, a drawback of this approach is that when users choose to display multiple layers simultaneously, the rendering performance may drop below 60 fps. In the case of object-based tiling, multiple layers are tiled together. This involves searching for the bounding box of a feature in other layers, and all the relevant data corresponding to the bounding box in those layers are considered for tiling. One advantage of object-based tiling is that multiple layers can be displayed simultaneously without compromising the 60 fps rendering performance. This allows for a richer and more comprehensive visualization experience. However, a disadvantage of this approach is that tiling needs to be done dynamically, meaning that adding new layers requires a re-tiling process to ensure the proper display and performance of the combined layers. A deep dive for evaluating these tiling schemes is also planned as future work.

Another discussion in tiling is the order of the operations during the tiling process. Operations that influence the tile size and the performance of rendering need to be conducted prior to the tiling process. Additionally, certain operations cannot be effectively executed after tiling. The process of clamping to the terrain should be conducted prior to the tiling operation. While it is possible to utilize a modified model

view matrix to adjust the heights of buildings based on the terrain, it is important to note that for a single b3dm file, only one model view matrix can be applied. As a result, individual adjustment of buildings within a tile is not feasible, and the same offset value is applied to all buildings within the tile. Due to this constraint imposed by 3D Tiles, it is necessary to perform the clamping to the terrain for each building prior to the tiling operation. To ensure that the tile size threshold is not compromised, it is essential to perform calculations of vertex normals and triangulation operations prior to the tiling process. These operations contribute to increasing the size of the geometric data of the features. Specifically, the calculation of vertex normals should be done before the triangulation step. This is because, after the triangulation process, although the surface normals remain unchanged, the calculation of vertex normals becomes slower due to the increased number of surfaces. By performing these operations before tiling, the efficiency and performance of the overall process can be optimized.

Another discussion, or encountered problem, in tiling is the high-precision rendering of the tileset. In WebGL, vertex coordinates are typically stored as 32-bit single-precision floats. However, in georeferenced 3D geospatial datasets, coordinates are stored as 64-bit double-precision values. These georeferenced coordinates often exceed the precision limits of the 32-bit single-precision floats, as they can have more than seven decimal digits. As a result, when mapping these coordinates to WebGL, precision loss occurs. This loss of precision leads to an unwanted jittering effect during the rendering stage, which negatively impacts the overall rendering quality. To overcome the precision loss issue, the centre of each tile is stored in the feature table using the earth-centred earth-fixed (ECEF) coordinate system as RTC (Related-To-Center) values. The coordinates within the tile are then transformed into local coordinates, where 32-bit single precision is sufficient, by applying translations based on the RTC values. During runtime rendering, the translated local coordinates are used to ensure a smooth rendering process and avoid the jittering effect caused by precision loss.

The last discussion on tiling has brought attention to the limitations of the 3D Tiles standard. Despite its significance as a prominent standard for 3D data streaming, it is not without notable drawbacks. One particular drawback lies in the absence of an explicitly defined data structure. While the standard requires a hierarchical tree structure, it does not specify a particular implementation such as R-Tree, Quadtree, or KD-Tree. As a result, different applications may employ varying data structures for the same dataset, thereby hindering interoperability. Furthermore, the standard does not provide an explicit guideline for tile sizing. Different applications may adopt different tile sizes for tiling the same dataset, leading to discrepancies in the number of tiles and rendering performance across applications. This variability further hinders interoperability and undermines consistency. This study aims to address and offer a solution to the existing interoperability issues within the 3D Tiles standard.

Considering the 3DSA component, the first remarkable discussion will be about half-edge data structure. One limitation of the 3D intersection implementation is its inability to handle non-manifold 3D shapes. This is because the half-edge data structure can only represent the left and right faces of an edge, while non-manifold shapes have edges that are incident to more than two faces (Fig.17). As a result, the 3DSA

component is unable to accurately calculate 3D touches for such shapes. The replacement of the half-edge structure with another for the detection of 3D touches is also a planned future work at the moment.

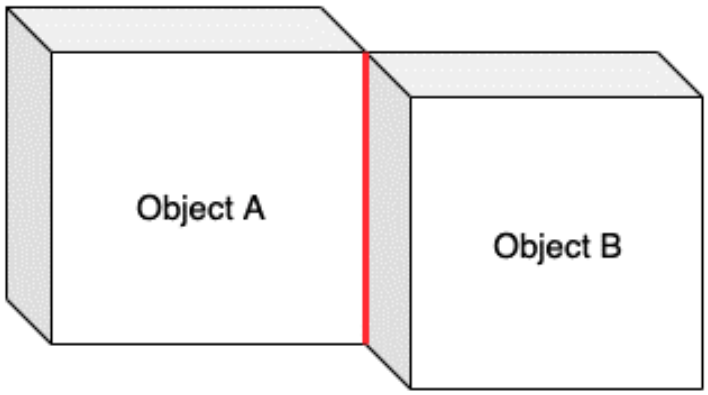

Fig. 17 A non-manifold shape. the red edge is incident to four faces

Another discussion or limitation in the 3DSA component arises in a specific case encountered during testing and debugging the algorithm. If one of the vertices of an object is not entirely inside the other object (Fig.18), the 3DSA implementation cannot construct the intersection as a complete 3D polygonal mesh. This limitation arises because the construction process relies on starting with a vertex that is fully contained within the other object, as described in Section 2.3. While it is possible to address this by increasing the number of ray-surface intersection tests, this scenario is rare in real-world data. Furthermore, increasing the number of ray-surface intersection tests negatively impacts the algorithm's performance, so this solution was not implemented.

Considering the PM component is discussed, the most notable discussion could be about the extrusion. The extrusion function is designed to work independently of coordinate systems. As a result, it successfully operates in coordinate systems where the Z-axis does not represent the elevation from sea level, such as the ECEF coordinate system. This ensures that the function is compatible with the ECEF coordinate system (EPSG 4978), which is the coordinate system used in the 3D Tiles standard. Throughout the work, the only coordinate system used is EPSG 4979. With this function, 2D polygons representing buildings are extruded based on the elevation information found in their attributes through the surface normals. If these elevation values were directly added to the Z values for the extrusion process, the elevated 3D building geometries would not be accurately formed in systems like ECEF. Furthermore, it is important to highlight that there is a notable error in the 3D Tiles standard documentation [12] regarding the definition of the coordinate system. The documentation incorrectly specifies the coordinate system as EPSG 4979. This mistake can potentially lead to confusion and inaccuracies in understanding and calculating coordinates. EPSG 4979

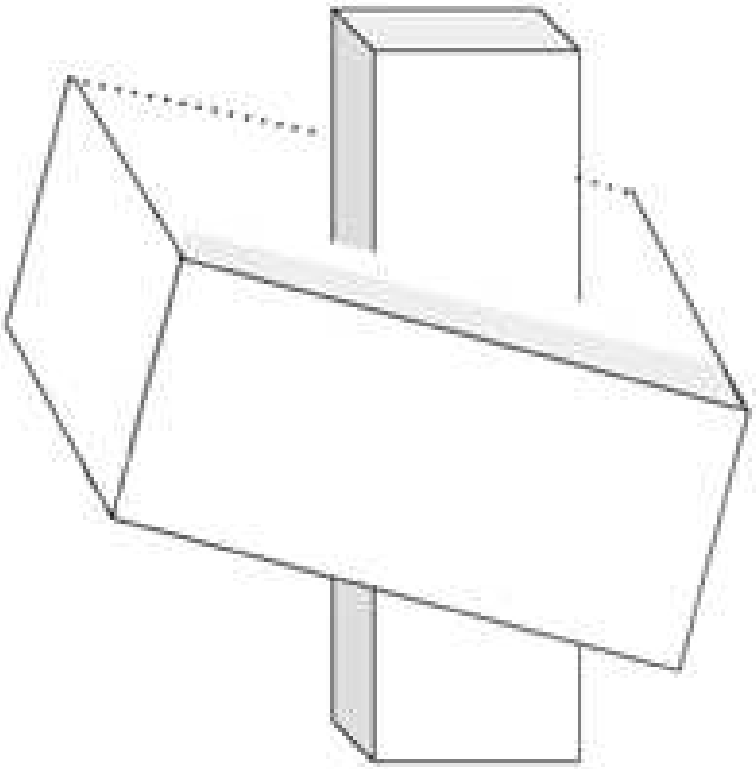

Fig. 18 The special case, none of the vertices of the objects are inside the other object

is a geographical coordinate system which consists of longitude, latitude and altitude and it is not ECEF when EPSG 4978 is a cartesian coordinate system.

5 Conclusion

This study presents the development of a web-based application aimed at facilitating the management of 3DCMs in an interoperable manner. The application enables users to upload their 2D building footprint data, generate 3DCMs, perform analyses on their models, and visualize both the 3DCMs and the results of the analysis. The tiling component of the system addresses the limitations of the 3D Tiles standard by implementing solutions to its drawbacks. However, it should be noted that the textures of 3D models are currently not supported and are not considered in the tiling algorithm. Future work will focus on developing texture support to enhance the capabilities of the tiling component. Besides, our future plans also include the evaluation of the tiling

schemes as mentioned in Section 4. In the PM component, an end-to-end pipeline is implemented to generate 3DCMs from 2D building footprints without the need for a modeling language. This allows users to create 3DCMs efficiently and seamlessly. The 3DSA component handles computationally expensive 3D boolean operations on the server-side, independent of the user's hardware configuration. This approach ensures consistent performance and reliable results. In future developments for the 3DSA component, the goal is to enhance the 3D collision capabilities by replacing or extending the half-edge structure. Additionally, we also plan to optimize tiling and 3DSA processes for efficient web-based data streaming through the implementation of multithreading and multiprocessing techniques, which are widely-known hardware-based acceleration techniques in computer graphics. Although we have not mentioned this kind of process acceleration in this work, we have observed remarkable results in our prior tests and have been about to share these findings.

Acknowledgements

We would like to express our sincere gratitude to the Scientifc and Technological Research Council of Türkiye (TUBITAK) for their support, their assistance has been instrumental in enabling the successful completion of this study.

Author Contribution

Ziya Usta and Alper Tunga Akın performed algorithmic development stages and prepared figures. Ziya Usta, Çetin Cömert and Alper Tunga Akın wrote the main article text.

Funding

This work was supported by The Scientifc And Technological Research Council Of Türkiye (TUBITAK) with the grant ID of 118Y452.

Data Availability Statement

The test data which is used in the 3DSA component for runtime observation is shared at https://mega.nz/folder/Xx9nBZQC#rZxk4ZWrU_VQTNmAXLvN-w

Declerations

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Consent for publication

If the article is accepted by the Editor-in-chief after the review process, all authors consent to the manuscript being published in Earth Science Informatics. The work did not include human participants in order to obtain their consent.

References

[1] Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4), 2842-2889.

[2] A high-performance, feature-packed library for all your mapping needs., https://openlayers.org/

[3] An open-source JavaScript library for mobile-friendly interactive maps, https://leafletjs.com/

[4] Advanced geospatial analysis for browsers and Node.js, https://turfjs.org/

[5] Rodrigues, J. I., Figueiredo, M. J., and Costa, C. P. (2013, July). Web3DGIS for city models with CityGML and X3D. In 2013 *17th International Conference on Information Visualisation* (pp. 384-388). IEEE.

[6] Prandi, F., Devigili, F., Soave, M., Di Staso, U., and De Amicis, R. (2015). 3D web visualization of huge CityGML models. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40.

[7] LOW-LEVEL 3D GRAPHICS API BASED ON OPENGL ES, https://www.khronos.org/webgl/

[8] Chaturvedi, K. (2014). Web based 3D analysis and visualization using HTML5 and WebGL (Master's thesis, *University of Twente*).

[9] Analytical Graphics Inc, 2015., https://www.agi.com/home

[10] Klokan Technologies, 2011. "WebGL Earth- open source 3D digital globe written in JavaScript", http://www.webglearth.org/

[11] Christen, M., and Nebiker, S. (2011). Openwebglobe sdk an open source high-performance virtual globe sdk for open maps. In *1st European State of the Map Conference*, Vienna.

[12] OGC 3D Tiles Spesification, 2018., https://github.com/AnalyticalGraphicsInc/3d-tiles

[13] GeoPortail., https://www.geoportail.gouv.fr/

[14] 3D Macau., http://www.3dmacau.com/

[15] Open3DGIS., www.open3dgis.org

[16] X3DOM, https://www.x3dom.org/

[17] Prandi, F., De Amicis, R., Piffer, S., Soave, M., Cadzow, S., Boix, E. G., and D'Hondt, E. (2013). Using CityGML to deploy smart-city services for urban ecosystems. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, W1.

[18] Mao, B., and Ban, Y. (2011). Online visualization of 3D city model using CityGML and X3DOM. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 46(2), 109-114.

[19] Prieto, I., Izkara, L. J., and del Hoyo, F. (2012). "Efficient visualization of the geometric information of CityGML: application for the documentation of built heritage". *Computational Science and Its Applications–ICCSA* 2012, 529-544.

[20] Gesquiere, G., ve Manin, A. 2012. "3D Visualization of Urban Data Based on CityGML with WebGL", *International Journal of 3-D Information Modeling (IJ3DIM)*, 1,3, pp. 1-15.

[21] Chaturvedi, K., Yao, Z., and Kolbe, T. H. (2015). Web-based Exploration of and interaction with large and deeply structured semantic 3D city models using HTML5 and WebGL. In Bridging Scales-Skalenübergreifende Nah-und Fernerkundungsmethoden, *35. Wissenschaftlich-Technische Jahrestagung der DGPF*.

[22] Koukofikis, A., Coors, V., and Gutbell, R. (2018). INTEROPERABLE VISUALIZATION OF 3D CITY MODELS USING OGC'S STANDARD 3D PORTRAYAL SERVICE. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(4).

[23] Gaillard, J., Peytavie, A., and Gesquiére, G. (2020). Visualisation and personalisation of multi-representations city models. *International Journal of Digital Earth*, 13(5), 627-644.

[24] Lu, M., Wang, X., Liu, X., Chen, M., Bi, S., Zhang, Y., and Lao, T. (2021). Web-based real-time visualization of large-scale weather radar data using 3D tiles. *Transactions in GIS*, 25(1), 25-43.

[25] Xu, Z., Zhang, L., Li, H., Lin, Y. H., and Yin, S. (2020). Combining IFC and 3D tiles to create 3D visualization for building information modeling. *Automation in Construction*, 109, 102995.

[26] Jaillot, V. (2020). 3D, temporal and documented cities: formalization, visualization and navigation (Doctoral dissertation, *Université de Lyon*).

[27] obj23dtiles., https://github.com/PrincessGod/objTo3d-tiles

[28] CityGML23DTiles, https://github.com/njam/citygml-to-3dtiles

[29] py3dtiles, https://github.com/Oslandia/py3dtiles

[30] 3D BAG, https://3dbag.nl/en/viewer

[31] Vitalis, S., Labetski, A., Boersma, F., Dahle, F., Li, X., Arroyo Ohori, K., ... and Stoter, J. (2020). Cityjson+ Web= Ninja. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 6(4/W1), 167-173.

[32] Tsiliakou, E., Labropoulos, T., and Dimopoulou, E. (2014). Procedural modeling in 3D GIS environment. *International Journal of 3-D Information Modeling (IJ3DIM)*, 3(3), 17-34.

[33] Martinovic, A. (2015). Inverse Procedural Modeling of Buildings.

[34] Ledoux, H., and Meijers, M. (2011). Topologically consistent 3D city models obtained by extrusion. *International Journal of Geographical Information Science*, 25(4), 557-574.

[35] Arroyo Ohori, K., Ledoux, H., and Stoter, J. (2015). A dimension-independent extrusion algorithm using generalised maps. *International Journal of Geographical Information Science*, 29(7), 1166-1186.

[36] Biljecki, F., Ledoux, H., and Stoter, J., 2016. "Generation of multi-LOD city models in CityGML with the procedural modelling engine Random3DCity", *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf*, 4,1.

[37] Van Ackere, S., Glas, H., Beullens, J., Deruyter, G., De Wulf, A., and De Maeyer, P. (2016). Development of a 3D dynamic flood WEB GIS visualisation tool. *Flood Risk Management and Response*, 106.

[38] Feng, L., Wang, C., Li, C., and Li, Z. (2011, December). A research for 3D WebGIS based on WebGL. *In Proceedings of 2011 international conference on computer science and network technology* (Vol. 1, pp. 348-351). IEEE.

[39] Chen, W., He, B., Zhang, L., and Nover, D. (2016). Developing an integrated 2D and 3D WebGIS-based platform for effective landslide hazard management. *International Journal of Disaster Risk Reduction*, 20, 26-38.

[40] Xiaoqing, Z., Jixin, L., and Yonghua, X. (2010, April). Architecture and application of 3D WebGIS based on Skyline and ArcGIS. In *2010 2nd International Conference on Computer Engineering and Technology* (Vol. 4, pp. V4-379). IEEE.

[41] Li, B., Wu, J., Pan, M., and Huang, J. (2015). Application of 3D WebGIS and real-time technique in earthquake information publishing and visualization. *Earthquake Science*, 28, 223-231.

[42] Von Schwerin, J., Richards-Rissetto, H., Remondino, F., Agugiaro, G., and Girardi, G. (2013). The MayaArch3D project: A 3D WebGIS for analyzing ancient architecture and landscapes. *Literary and Linguistic Computing*, 28(4), 736-753.

[43] Pispidikis, I., and Dimopoulou, E. (2016). DEVELOPMENT OF A 3D WEBGIS SYSTEM FOR RETRIEVING AND VISUALIZING CITYGML DATA BASED ON THEIR GEOMETRIC AND SEMANTIC CHARACTERISTICS BY USING FREE AND OPEN SOURCE TECHNOLOGY. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4.

[44] Auer, M., and Zipf, A. (2018). 3D WebGIS: From visualization to analysis. An efficient browser-based 3D line-of-sight analysis. *ISPRS International Journal of Geo-Information*, 7(7), 279.

[45] Guttman, A. (1984, June). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data* (pp. 47-57).

[46] Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2), 187-260.

[47] Bieri, H. (1995). Nef polyhedra: A brief introduction (pp. 43-60). *Springer*, Vienna.

[48] Cork Boolean Library, https://github.com/gilbo/cork.

[49] Usta, Z., Akın, A. T. and Cömert, Ç. (2023). Deep learning aided web-based procedural modelling of LOD2 city models. Earth science informatics, 16 (3), s. 2559–2571. doi:10.1007/s12145-023-01053-0