VIABILITY TESTING OF GAME ENGINE USAGE FOR VISUALIZATION OF 3D GEOSPATIAL DATA WITH OGC STANDARDS

P. Würstle[1]*, R. Padsala[1], T. Santhanavanich[1], V. Coors [1]

[1] Centre for Geodesy and Geoinformatics,
Stuttgart University of Applied Sciences (HFT Stuttgart), Schellingstraße 24, D-70174 Stuttgart, Germany
(patrick.wuerstle, rushikesh.padsala, thunyathep.santhanavanich, volker.coors)@hft-stuttgart.de

Commission IV, WG IV/9

KEY WORDS: Game Engine, OGC Standards, Visualization, Urban Digital Twins, 3D Tiles

ABSTRACT:

Urban digital twins have become an essential factor for cities and communities to visualize, simulate and analyze data. The conventional geospatial standards work great with online platforms such as CesiumJS or ArcGIS API for JavaScript. However, their usage in different platforms such as game engines has not been well established yet. Game engines provide an interesting application case because they offer a different approach to visualizing large city models and provide a high graphical fidelity.
This paper aims to answer how the existing standards, such as the API standards GeoVolumes and SensorThings, as well as the 3D model standards 3D Tiles and Esri Indexed 3D Scene Layer (I3S), can interact with game engines. For this purpose, three use-cases were selected and have been used to build applications. These focus on using sensor data in AR and different city development scenarios in a digital environment.
This study shows that different geospatial standard formats such as 3D Tiles, I3S, and GL Transmission Format (glTF) can be used in game engines, either directly or over a GeoVolumes Server. Their implementation makes it possible to use the advantages of game engines with real-world datasets.

1. INTRODUCTION

Game engines are quickly becoming more than just their use in developing computer games. Nowadays, game engines are considered a major catalyst for innovation and creativity, serving various industries. Some key elements that differentiate game engines from commonly used web technologies are real-time rendering engines, the capability to use a variety of input hardware for handling 3D scenes, and physics engines to digital replicate real-world physical laws of gravity, collisions, fluidity, reflex, and more, audio engines for immersive sound effects, artificial intelligence to model complex in-game behaviors, networking ability to connect with different users over internet and much richer graphical user interface for users interactivity and experience. Its recently found interface with the geospatial domain, particularly with the urban digital twins, has used the world of data interactivity and visualization beyond simple and commonly used web programming languages. The use cases of working with urban digital twins within game engines are limited and constantly evolving. Thus, exploring the Open Geospatial Consortium (OGC) FAIR (Findable, Accessible, Interoperable, and Reusable) initiative becomes interesting, particularly the interoperability of different OGC standards with the game engines. This is because urban digital twins are often seen as a medial point where other data-driven technologies can be integrated, managed, and simulated. Its interactivity with high fidelity visualization in-game engines has a game-changing potential as a part of critical software infrastructure for the development of smart, sustainable, and resilient cities.

An urban digital twin is a 3D model replica of a real-world city integrated with static and dynamic information, providing methods for analyzing and optimizing the built environment. Similar to the digital twin technologies used in automobile industries to inform manufacturers on how products work and will react in the future, the urban digital twins are widely used to calculate various urban informatics with which multiple “what-if” scenarios can be simulated to understand its future before it gets constructed in the real-world environment. A significant emerging trend sought to be a game-changer in the urban digital twin domain is its interface with game engines. Though initially focused on developing computer games, engines such as the Unreal Engine and the Unity Engine have demonstrated the unique ability to visualize all the elements of a built environment in real-time and with collaborative interaction. This is due to the fact that such software products can turn urban digital twins into immersive 3D simulations that users can freely explore and understand from different perspectives. The recent implementation of Unreal Engine plugins for Cesium web globe, as well as the Unreal and Unity plugins for Esri’s web globe, have opened up new doors to combining the 3D geospatial capabilities with the rendering power of game engines. Due to its recent implementation, the use cases showing its capabilities are still limited and continuously evolving. With this background, this paper focuses on testing different Open Geospatial Consortium (OGC) standards such as 3D Tiles, Esri Indexed 3D Scene Layer (I3S), SensorThings, and the GeoVolumes API with the Unreal and Unity Engines along with Cesium and Esri’s web globe. The later part of this paper focuses on showcasing the use of OGC standards with game engines in real-world use cases under different themes such as its use in Augmented Reality/Virtual Reality, city development planning, city acoustic analysis, and open-world navigation.

* Corresponding author.

2. STATE OF THE ART

Game engines, primarily used for developing video games, have used digital twins, though mostly fictitious, for a long time. For example, New York City in Marvel Spider-man, Los Angeles in Grand Theft Auto, Dubai in Hitman, and the entire world in Microsoft's Flight Simulator. Putting an actual city in a computer game sounds very interesting. Yet, the game developers rely on fictitious cities to replicate the real world in a digital environment. Historically, besides the costs required to capture the digital twins of the entire computer game scene, the availability of 3D city models has always been a cause of concern.

However, due to the evolution of geospatial technologies to capture and present real-world city objects virtually with accuracy, the connection between game engines and geospatial technologies is gathering pace in the simulation and visualization of urban digital twins. Urban digital twins are attracting attention as a technology that can address challenges related to the smart, sustainable development of the built environment. A three-dimensional geospatial platform is a vital part of a digital twin city. It allows visualizing and analyzing city objects such as terrains, buildings, vegetation, roads, bridges, and other physical structures (Kolbe et al., 2005). Such datasets can be used to study various urban phenomena from multiple perspectives. Web-based geospatial platforms have provided excellent user accessibility and convenience for content sharing. For example, Cesium, Mapbox, and JavaScript for ArcGIS allow users to upload their 3D geospatial data and develop an interactive web application that is accessible without installing additional plugins. However, they are limited in their ability to provide functionality that the web platform does not support. In addition, the execution environment has performance constraints as a web browser. This is where game engines become essential. They can provide far better realistic visualizations and pave the ideal foundation for photo-realistic, immersive simulations of the real world. (Lee et al., 2020) proposed a planetary-scale geospatial open platform developed in Unity3D. Their results provided a geospatial data visualization to the user, which spanned over 30 TB and 71.5 billion tiles. Their platform was developed for the purpose of tiling-based static geospatial data management for rendering on a planetary scale. However, they missed integrating dynamic datasets like that of sensors or moving objects, focusing only on the visualization of static city objects. (Buyuksalih et al., 2017) visualized 3D city models based on Unity3D and also estimated potential solar energy on buildings for Istanbul City. Other use cases are still limited and evolving, (Sihombing and Coors, 2018, Matthys et al., 2021, Edwards et al., 2015, Andaru et al., 2019) have already shown how 3D city models captured using geospatial technologies can be used with game engines for visualization of city objects, use of AR for city development, storytelling/public participation, AEC industry, preservation and visualization of heritage buildings. Such research shows the potential of 3D Unity visualization and game engine for 3D GIS visualization. However, some common issues discussed in the majority of the related research links to issues with coordinate transformation, general interoperability with geospatial datasets, integration of dynamic datasets like sensors, support of databases, and Application Programming Interface (API). Furthermore, no studies have yet tested the use of different frameworks related to the delivery of geospatial datasets developed by the OGC with game engines.

Thus, through different use cases, the present study focuses on answering: How to consume OGC frameworks of 3D Tiles, I3S, SensorThings API, and GeoVolumes API within Unreal and Unity game engines for its use in developing initial prototypes for AR/VR, Mobility, acoustic and city development use cases? In the process, this paper also documents our experience, and the challenges faced while testing the usage of the OGC frameworks as mentioned above in the recently released Cesium (Cesium, 2020) and Esri (Esri, 2020) plugins for popular game engines Unity3D and Unreal Engine.

3. ARCHITECTURE

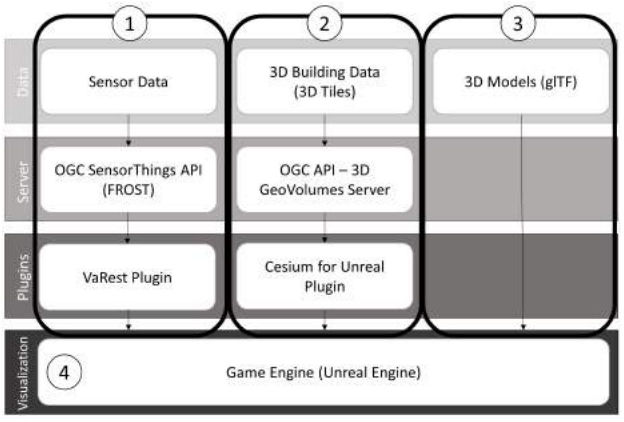
1
2
3
Data
Sensor Data
3D Building Data (3D Tiles)
3D Models (glTF)
Server
OGC SensorThings API (FROST)
OGC API – 3D GeoVolumes Server
Plugins
VaRest Plugin
Cesium for Unreal Plugin
Visualization
4
Game Engine (Unreal Engine)

Figure 1. Architecture

The Architecture is separated into three workflows which are marked in Figure 1. The workflows describe how to use Sensor data, 3D Building data, and 3D Models, respectively. The fourth part describes the usage of the game engine in this research.

① **Sensor Workflow** The rapid development of the Internet of Things (IoT) has become a foundation for the urban digital twins concept, where the interconnection among these devices not only represents the sensor data from the physical world but also realizes the controllable intelligent system (Lv et al., 2021). The SensorThings API (STA) (Liang et al., 2016) is an open-source standard based on the OGC SWE standards, which copes with the interoperability challenge of heterogeneous IoT devices. STA uses JSON as the data encoding and supports REST-like application programming interface (API) via Hypertext Transfer Protocol (HTTP) and Message Queuing Telemetry Transport (MQTT) protocol technologies. Several cities such as Nantes (France), Hamburg (Germany), and Helsinki (Finland) (Fischer et al., 2021, Hernández et al., 2020) have been using STA, including the European Union INSPIRE (Kotsev et al., 2018) which serves thousands of public sector data providers has considered using STA as part of their spatial data infrastructures (SDI).

② **3D Building Workflow** Different 3D formats are utilized for this research and fill different roles in the development process. The base data of the 3D city model is stored in the CityGML format. It is an OGC standard to store and exchange city model information. The most recent version is 3.0, although most of the currently available models still utilize the CityGML 2.0 standard. The CityGML 3.0 standard has been adopted in September 2021 (Kolbe et al., 2021). The missing 3D building datasets are filled in by converting 3D building models from ArcGIS CityEngine to CityGML (Padsala et al., 2020).

3D Tiles is an OGC (Open Geospatial Consortium) standard for 3D geospatial data. It is mainly used for visualization purposes in web applications such as CesiumJS. In accordance with its name, it uses a tiling method that creates a hierarchical structure. 3D Tiles are built on the glTF format (Cozzi et al., 2019). Another format for streaming developed by ESRI is the I3S format. I3S is an OGC standard for 3D objects, integrated meshes, point features, and point clouds (Reed and Belayneh, 2021).

The 3D GeoVolumes API is a geospatial 3D content specification described by an enclosing bounding volume. This specification was developed during the 3D Data Container, and Tiles API Pilot (Miller et al., 2020). In this specification, the data organization of the geospatial data in 3D GeoVolumes is flexible and adjustable according to the provider's needs. Each container in the 3D GeoVolumes may have one or more children containers whose extents are completely contained in the parent volume, and each container contains one particular dataset but may consist of links to the dataset in different formats, e.g., 3D Tiles or I3S, as shown in Figure 2. Later, the 3D GeoVolumes API was successfully deployed and served 3D geospatial contents in various research projects (Interoperable Simulation and Gaming Sprint Engineering Report, 2020, Daly and Phillips, 2021, Deininger et al., 2020). In the "Simple service-based use of 3D data" project by the Runder Tisch GIS, the GeoVolumes technology is used in cooperation with the state departments of Bayern and Baden-Württemberg, Germany. The states provide their 3D building model data for participating regions in 3D Tiles and I3S format via a 3D GeoVolumes server. Later, the participants can use this data to populate their applications.

③ **3D Model Workflow** Game engines support a variety of 3D data formats. Amongst them are FBX (Filmbox), OBJ (Wavefront file format specification), and glTF (open GL Transmission Format). In a study from 2019 by (Nam et al., 2019). comparing the different formats, it is shown that glTF has the fastest loading times, just ahead of FBX (Nam et al., 2019). Therefore, most of the models used for the development are in either the glTF or the FBX file format.

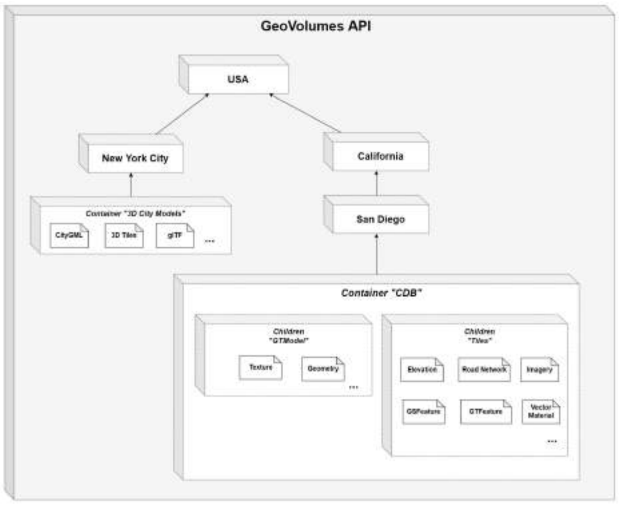

Figure 2. An exemplary hierarchical resource architecture of the 3D GeoVolumes API

④ **Game Engines** Game engines provide a different visualization platform to more common tools in the geoinformation field like CesiumJS and ArcGIS for JavaScript for the web or traditional GIS software. One of the essential features that game engines provide is collisions between the controlled pawn and the environment (Torres-Ferreyros et al., 2016). There is a range of different game engines that provide different features and functionalities. A few of the most common ones are Unreal Engine, Unity, and Cry Engine. There are a few comparative studies that focus on these game engines (Barczak and Woźniak, 2020), (Vohera et al., 2021). They conclude that the engines have advantages in different fields. Unity, for example, is well suited for beginners because of its good documentation and vast user base but only has basic functionalities in regard to animation. According to (Vohera et al., 2021), The Unreal Engine has a higher learning curve but excellent visual output.

For the development in this paper, there are a few special conditions that influence the decision on which game engine to use significantly. There is a need to visualize large building models. In the OGC Interoperable Simulation and Gaming Sprint (Daly and Phillips, 2021) these requirements have been tested, and it has been shown that the Unreal Engine and the Unity Engine provide the functionality to visualize building datasets. The process is described in more detail in Section 3.1.

The Unreal Engine uses C++ as its programming language, but it also provides a blueprint system that does not require programming knowledge (Vohera et al., 2021). The Unreal Engine is currently (since April 2022) on Version 5.0, but the development was done with Version 4.27.

The Unity Engine is based on the C# and C++ programming languages, but also supports scripts in C# and JavaScript(Vohera et al., 2021). It supports different 3D formats such as .FBX and .OBJ (Megha et al., 2018).

There is a solution to include I3S and 3D Tiles into the Unreal Engine. The solution from Esri is called "ArcGIS Maps SDK for Unreal Engine" and is currently still in beta. Cesium provides a plugin called "Cesium for Unreal" which allows the use of 3D Tiles in developments with the Unreal Engine. The Unity game engine has a similar plugin to include I3S layers into the Game Engine, but is currently missing an implementation for Cesium 3D Tiles (Daly and Phillips, 2021).

3.1 Implementation

Three different data sources are used for this development with a game engine. They consist of sensor data, 3D building data, and other 3D models. These workflows are implemented in the following section and are evaluated based on if they can be visualized in general, if there is any loss in information such as metadata and if the performance is acceptable.

3.1.1 Sensor Data The sensor data that is used for this project comes from the Sensor.Community. The Sensor.Community has around 14,000 sensors worldwide. These sensors measure PM2.5, PM10, and recently also noise. These sensors measurements can be accessed through an API. The API only provides real-time measurements, whereas the historical data must be accessed over files. For this reason, a SensorThings server is used in this development. The SensorThings server used in this project is based on the FROST implementation. It is set up with a docker container where the settings are defined inside a YAML file. It directly collects the data via a Python script from the Sensor.Community API by an HTTP Post and stores it. This allows applications to request historical data through the SensorThings server with the filtering functionalities of SensorThings.

The implementation of SensorThings into the Unity Engine still needs to be investigated but should, in theory, be possible.

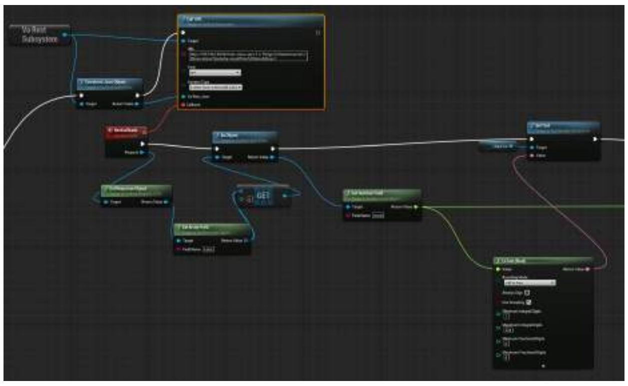

Figure 3. SensorThings access via Unreal Engine

The Unreal Engine does not natively provide the functionality to access the SensorThings server. Since the central part of the development was done with blueprints and not C++, the simplest way to connect to the SensorThings API is with the VaRest plugin. The plugin works by making a request to the SensorThings server. It allows for Get, Post, Delete, and custom requests, and it requires a URL input. The URL has to be constructed with the IP address or domain and the SensorThings standard query. In the request in figure 3, the URL will only collect the latest data point. This calls an event (RestCallback) which contains the result and can then be handled like a regular JSON object. The plugin also provides some blueprint nodes that can be used to manipulate the retrieved data. In the workflow shown in figure 3, the resulting object is retrieved, and the array field with the value name is extracted. The value field contains common information like "phenomenonTime", "resultTime", and "result". To get only the result value of the sensor, the node GetNumber field is used. This value is then able to be displayed in the game engine application. The tools available from the VaRest plugin and the Unreal Engine allow retrieving and displaying the sensor measurements. During the retrieving process, no data is lost. There is also no noticeable performance drop compared to a web application that accesses the same SensorThings server.

3.1.2 3D City Models The 3D city models used in this research come from different sources. This research project already has access to 3D models of the city of Stuttgart. Additionally for the missing parts, 3D city models were generated for this research using ArcGIS CityEngine. Generation of 3D city models using software like CityEngine usually happens by importing vector datasets such as building footprint polygons, road network line geometries, land use parcels, and vegetation points. Additionally, with the CityEngine, the user can procedurally generate these vector datasets and other related attributes such as building height, Floor Area Ratios, and Window to Wall ratio based on local zoning laws using its inbuilt Computer Generated Architecture (CGA) shape grammar programming technique ((Reitz and Schubiger, 2014)). This has been used to generate a generic texture for an existing 3D city model used in the game engine. 3D city models can be visualized in different formats, but in this research, the focus is set on the formats of 3D Tiles and I3S. To get these formats from a city engine workflow, they must be converted. To generate I3S, ArcGIS Pro has been used, and to generate a 3D Tiles dataset, FME has been used. A high-level workflow explaining CityEngine to 3DTiles and I3S conversion is shown in figure 4.

The Unreal Engine provides different methods to include 3D building models in the project. To use the 3D Tiles format with

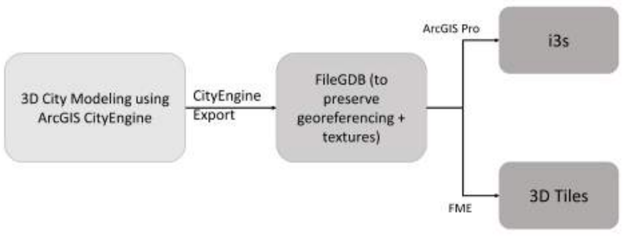
3D City Modeling using ArcGIS CityEngine
CityEngine Export
FileGDB (to preserve georeferencing + textures)
ArcGIS Pro
i3s
FME
3D Tiles

Figure 4. ArcGIS CityEngine to 3DTiles/I3S

the Unreal Engine, the "Cesium for Unreal" plugin can be used. Cesium Ion provides a worldwide 3D Tiles building dataset and a terrain model that can be used. The plugin also allows the possibility to include other 3D Tiles models. This can be done in multiple ways. Building models can be uploaded to Cesium Ion and streamed to the Unreal Engine development. They can also be loaded from a local directory via the "file:///" prefix before the file path. Another possibility is to connect the 3D building model to the Unreal Engine via a URL. In this research, the tilesets are connected via URL. In this case, the URL points to a GeoVolumes server that hosts a 3D tileset of the area of interest. The 3D Tiles model has to have a specific format to be able to be displayed at the correct position on the globe (Daly and Phillips, 2021).

The properties of the 3D Tiles building, like building age, can be accessed through the game engine. Loading the tileset through the GeoVolumes server causes delays in the visualization. This results in missing buildings and terrain immediately after loading the level. If pawns that require collision are placed in the level, this delay can cause them to drop below the ground level and fall endlessly. To fix this performance issue, either a delay has to be built in to let all the datasets load before placing the pawns in the environment, or platform assets have to be placed below the pawns.
3D building models can also be used in the I3S format. Esri developed plugins for the Unreal Engine called "ArcGIS Maps SDK for Unreal Engine". The plugin is currently still in development and not directly available through the Unreal marketplace but has to be downloaded from the Esri Early Adopter website. The plugin is created for C++ projects and cannot be used directly from blueprint projects. However, the tool also offers a GUI that can be used to add layers from files or URLs. Trying to stream I3S directly from a GeoVolumes server into the Unreal Engine resulted in an error. As of now, this problem has not been fixed; therefore, it is only possible to use local data or data hosted on the ArcGIS Enterprise Portal. The ArcGIS Maps SDK for Unreal Engine has been updated to work with Unreal Engine 5 [1]. The"ArcGIS Maps SDK" implementation for the Unity Engine is similar to the plugin for the Unreal Engine. The installation also requires access to the Esri Early Adopter website. The plugin GUI is also very similar and allows for the same data sources, local and by URL. However, it is not possible to stream I3S from an OGC API - 3D GeoVolumes server.

3.1.3 3D Models The other 3D models are in the glTF or the FBX format. The Unreal Engine provides its own tools to integrate data in these formats into the game world. In the current state, the models have to be loaded from a local source. By importing the data in this way, many different settings are available. Importantly, many models have single parts; by importing

[1] https://developers.arcgis.com/unreal-engine-sdk/

them, the user has to choose if they want to import them separately or as one. Web applications, for example, CesiumJS, can load glTF models from an OGC API - 3D GeoVolumes server and display them. Using this approach for the Unreal Engine has been unsuccessful so far. Implementing an OGC API - 3D GeoVolumes server to serve glTF to a Unity project has not yet been tested. Since the models are loaded locally, they don't have performance issues like those loaded from a server. During the import process, it occasionally happens that materials lose their connections. This can be rebuilt by hand without negatively impacting the final model.

4. USE CASES

Three different use cases were developed based on the implementation from Section 3.1. The resulting prototypical applications aim to support participation processes in specific ways. The first use-case uses AR to visualize sensor readings and additional information about the sensor. The created application uses sensors from previous research projects as an example. It can be generally extended to information in the real world and given to citizens so they can walk around an area of interest and inform themselves about the local conditions. The second use case details the creation of an application that supports digital participation processes. It focuses on an area where citizens want to experience their quarter car-free. The resulting application focuses on the noise level in the quarter with and without cars. It was developed as a small game to make it more attractive to participants. The third use case focuses on city development. It demonstrates the use of a 3D gaming world to visualize different building scenarios as an alternative to more widely used 3D web globes.

4.1 AR

Game engines provide a good base for AR/VR applications. In the Unreal Engine, the frameworks of Google (ARCore) and Apple (ARKit) are available to develop AR applications for their respective mobile platforms. This application utilizes ARCore as a base framework. For the testing and the later usage, an Android smartphone is used. The simplest way to connect a sensor from a physical location to the sensor data from a SensorThings server is via a candidate image. The candidate image is a reference image of the physical sensor that is stored in the application and compared with the images from the phone's camera. If the images match, the sensor reading is shown with additional metadata such as the location and date of the measurement (Figure 5). The measurements are loaded from the SensorThings server via the VaRest plugins HTTP Get request (Figure 1).

Additionally, a model of the building or location can be displayed to show the user the current placement (Figure 5). This is done by loading a glTF model and placing it in the environment. The AR applications can detect surfaces and place the model accordingly. The application developed in this use case can be used to retrieve the data of a specific air quality sensor that is measuring the CO2 level inside the rooms of the university building. The application is a prototype and is used as proof of concept.

Additionally, a model of the building or location can be displayed to show the user the current placement (Figure 6). This is done by loading a glTF model and placing it in the environment. The AR applications can detect surfaces and place the model accordingly.

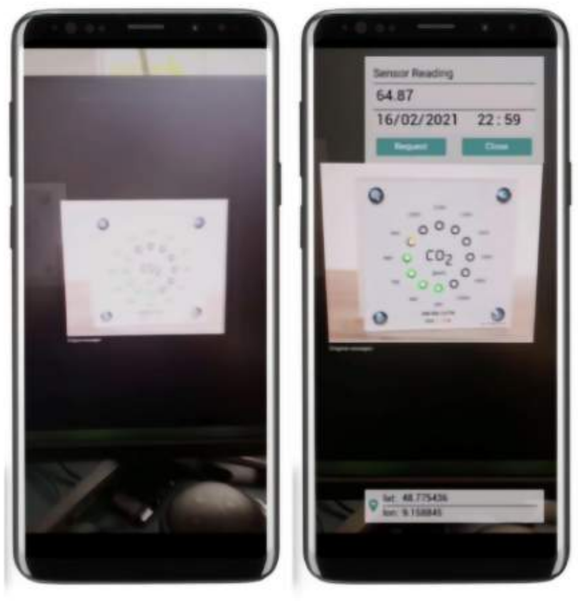

Figure 5. AR Application for Sensor reading

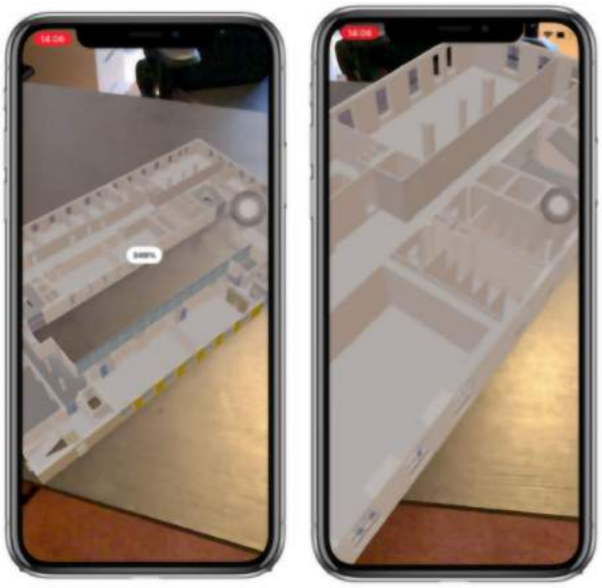

Figure 6. AR Application for Building Placement

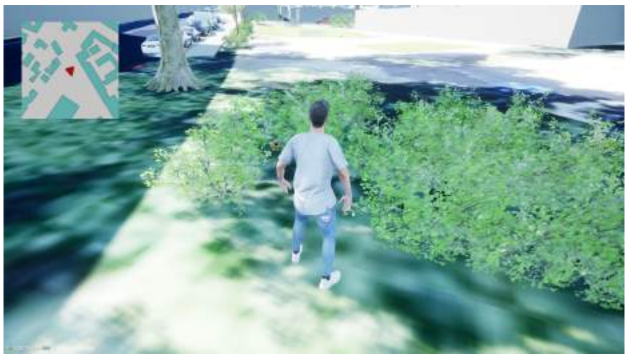

Figure 7. Bird hidden in the Environment

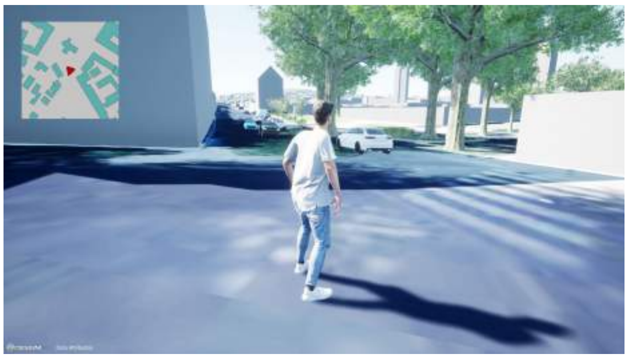

Figure 8. Car Traffic

4.2 Mobility and Acoustic

Game engine technology can be used to aid city development efforts and public participation processes. This application uses a 3rd-Person perspective to portray the city and move around it. The created application incorporates multiple approaches to allow for different gaming scenarios in one area. The selected area is home to multiple research topics. The subjects of these research topics regard mobility and the acoustic in a city planning environment. The goal of the acoustic and mobility topic is to provide an insight into a car-free quarter and show the difference in the soundscape. For this purpose, a small game was developed where the player has to collect birds in the environment of the research area.

The area is reproduced in the game engine using the architecture proposed in Section 3. The environment is built up of the Cesium terrain from Cesium Ion and their orthophoto data. The buildings come from different sources. One is Cesium Ion, and the other is a OGC API - 3D GeoVolumes hosting 3D Tiles of the Stuttgart region. Models of cars and flora are placed in the game to add a more life-like feeling to the development. The standard pawns that the user can control are replaced with more realistic models. These assets are free to use or procured through the Unreal Engine marketplace.

The game is structured through different menus. The player must select the game mode and the character they want to control. In the game itself, the player needs to talk to an NPC (non-player character) to get the quest of finding four different birds in the environment (Figure 7). The quest information can always be found in the quest menu. This is set up, like all menus, as a widget.

The player can navigate freely through the environment and has to listen to the sound. The scenario has three different sounds coming from the quest birds, other birds, and cars. The cars drive around the area on a set route and make engine noises (Figure 8). The car noises make it more difficult for the player to find the birds based on their chirps. The car noises can be turned off, and the cars can be removed from the scenario to emphasize the difference between a car-free quarter and one with cars. This application will give the participant a look into the differences without the need to prohibit cars from entering the area in the real world.

4.3 City Development

This use case aims to support the city development process. The application utilizes a character that can be controlled and moved around different city development scenarios. The user can switch between a first and third-person perspective and use different transportation modes such as walking, driving in a car, or flying with a drone. These modes of transport are included to give the user different ways to explore the area of interest.

The central part of this use case is that different 3D building models can be visualized and switched between with a button press. The 3D models can be from different sources and are streamed to the final application. This allows for storing and managing the 3D building models in a central place. The advantage of this method is that the 3D building models can be replaced or improved at a singular location, and the deployed applications will stream the new dataset.

Switching between the building models should give the user a sense of the differences between the building models and give a better sense of the scale and design. This could help in participation processes by allowing participants to get an impression of newly planned buildings compared to the existing ones. It is especially useful to experience different shadow casts of different height buildings.

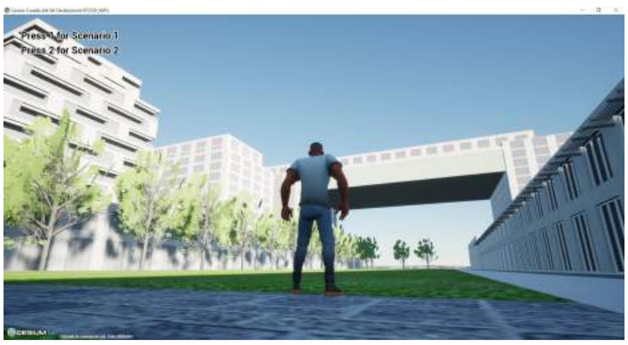

Figure 9. Area with to be developed Buildings

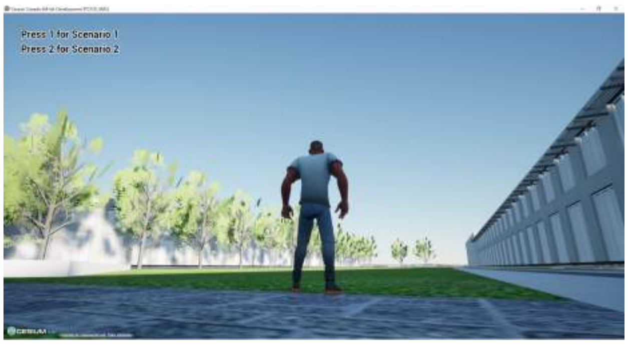

Figure 10. Area with existing buildings

The 3D models are from an area in Canada and procedurally generated as CityGML and later converted to 3D Tiles. The tilesets are hosted on an OGC API - 3D GeoVolumes Server. The newly planned concept contains higher buildings and a bridge between two buildings (figure 9), whereas the original buildings (figure 10) are relatively low.

5. CONCLUSION

The methods presented in this research can be instrumental in developing applications using urban digital twins, for example, in public participation and planning processes, where it is essential to give the participants an understanding of the projects that are as detailed and realistic as possible. The algorithm of CityEngine can be used to support realism by automatically generating building textures. This is especially useful since many of the available 3D city models don't have a texture created through image flights. The automatically created textures provide a much richer view of the surrounding buildings than the standard gray or white models. A realistic digital environment can be beneficial if it is combined with processes that are already in use, like a digital participation platform and in-person participation.

On the technical side, there are still a lot of improvements possible, mainly since the technology to stream 3D city models into game engines is still very young. More generally, there are some questions still to be resolved. One such question is the inclusion of the terrain in formats like quantized-mesh into the Unreal Engine, which has not been tested yet. Another remaining question is how a building in a 3D tileset can be treated as a singular entity. This is necessary to visualize building-specific information on the fly. Cesium for Unreal has added functionality to get metadata for each building by intersecting the line of sight with the buildings. This can provide more additional information that can be used and shown to the user. This also allows applying a style to the tileset. In use cases where it is necessary to switch the color code of buildings based on simulated data like heat demand or photovoltaic potential on the fly, the values of the predefined material have to be changed. There is also the alternative of applying a style to the tileset before integrating it into the game engines. On the I3S side of the development, the first question that needs to be answered is how to provide I3S datasets through an OGC API - 3D GeoVolumes server to Unity and the Unreal Engine.

Using the OGC API – 3D GeoVolumes to provide a tileset, specifically 3D Tiles, to the game engine comes with limitations. The simulated physics in the game engine that is used for collision calculations between different in-game assets causes some issues when the tileset is not loaded immediately. The assets don't have a surface on which they can rest and start falling. Adding additional surfaces below the terrain that is streamed can circumvent this. These have to be lined up correctly to be below the terrain so as not to be seen in the application and high enough so the assets don't clip through the terrain once it is loaded.

These issues vary based on the internet connection. The loading times also appear to be longer than when loading the tileset in a simple web viewer. If this impression can be confirmed will have to be tested in future work. To further develop the use case described in Section 4.2 and gain a better understanding of its practicality, it will be tested with students and voluntary participants.

In conclusion, the research presented in this paper realizes a method to visualize realistic city models with different OGC standards in game engines. The consumption of geospatial datasets in-game engines has become much easier than before. However, some limitations still apply that affect the final product in a negative way.

ACKNOWLEDGEMENTS

The M4_LAB project on which this article is based was funded by the Federal Ministry of Education and Research under the number 03IHS032A. The responsibility for the content of this publication remains with the authors.

REFERENCES

Andaru, R., Cahyono, B., Riyadi, G., Ramadhan, G., Tuntas, S., 2019. The Combination of Terrestrial LIDAR and UAV Photogrammetry for Interactive Architectural Heritage VIsualisation Using UNITY 3D Game Engine. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W17, 39-44.

Barczak, A., Woźniak, H., 2020. Comparative study on game engines. Siedlce University of Natural Sciences and Humanities, 5–24.

Buyuksalih, I., Bayburt, S., Buyuksalih, G., Baskaraca, A., Karim, H., Rahman, A., 2017. 3D Modelling and Visualisation Based on the UNITY Game Engine – Advantages and Challenges. *ISPRS Annals of Pho-togrammetry, Remote Sensing and Spatial Information Sci-ences*, IV-4/W4, 161-166.

Cesium, 2020. Cesium for unreal. https://cesium.com/platform/cesium-for-unreal/.

Cozzi, P., Lilley, S., Getz, G. (eds), 2019. *3D Tiles Specification 1.0*. Open Geospatial Consortium, Wayland, MA, USA.

Daly, L., Phillips, R. (eds), 2021. *Interoperable Simulation and Gaming Sprint Year 2 Engineering Report*. Open Geospatial Consortium, Wayland, MA, USA.

Deininger, M. E., von der Grün, M., Piepereit, R., Schneider, S., Santhanavanich, T., Coors, V., Voß, U., 2020. A Continuous, Semi-Automated Workflow: F rom 3D City Models with Geometric Optimization and CFD Simulations to Visualization of Wind in an Urban Environment. *ISPRS International Journal of Geo-Information*, 9(11). https://www.mdpi.com/2220-9964/9/11/657.

Edwards, G., Li, H., Wang, B., 2015. BIM based collaborative and interactive design process using computer game engine for general end-users. *Visualization in Engineering*, 3.

Esri, 2020. Arcgis maps sdk for unreal engine. https://developers.arcgis.com/unreal-engine-sdk/reference/release-notes/prior-releases/0.1.0/.

Fischer, M., Gras, P., Löwa, S., Schuhart, S., 2021. Urban Data Platform Hamburg: Integration von Echtzeit IoT-Daten mittels SensorThings API. *ZfV-Zeitschrift für Geodäsie, Geoinformation und Landmanagement (zfv 1/2021)*.

Hernández, J. L., García, R., Schonowski, J., Atlan, D., Chanson, G., Ruohomäki, T., 2020. Interoperable Open Specifications Framework for the Implementation of Standardized Urban Platforms. *Sensors*, 20(8). https://www.mdpi.com/1424-8220/20/8/2402.

Interoperable Simulation and Gaming Sprint Engineering Report, 2020. Interoperable Simulation and Gaming Sprint Engineering Report. http://docs.ogc.org/per/20-087.html.

Kolbe, T., Gröger, G., Plümer, L. (eds), 2005. *CityGML: Interoperable Access to 3D City Models*. Open Geospatial Consortium, Wayland, MA, USA.

Kolbe, T., Kutzner, T., Smyth, C., Nagel, C., Roensdorf, C., Heazel, C. (eds), 2021. *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard*. Open Geospatial Consortium, Wayland, MA, USA.

Kotsev, A., Schleidt, K., Liang, S., Van der Schaaf, H., Khalafbeigi, T., Grellet, S., Lutz, M., Jirka, S., Beaufils, M., 2018. Extending INSPIRE to the Internet of Things through SensorThings API. *Geosciences*, 8(6). https://www.mdpi.com/2076-3263/8/6/221.

Lee, A., Chang, Y.-S., Jang, I., 2020. Planetary-Scale Geospatial Open Platform Based on the Unity3D Environment. *Sensors*, 20, 5967.

Liang, S., Huang, C.-Y., Khalafbeigi, T., 2016. OGC SensorThings API Part 1: Sensing, Version 1.0.

Lv, Z., Lou, R., Li, J., Singh, A. K., Song, H., 2021. Big Data Analytics for 6G-Enabled Massive Internet of Things. *IEEE Internet of Things Journal*, 8(7), 5350-5359.

Matthys, M., De Cock, L., Vermaut, J., Van de Weghe, N., De Maeyer, P., 2021. An ”Animated Spatial Time Machine” in Co-Creation: Reconstructing History Using Gamification Integrated into 3D City Modelling, 4D Web and Transmedia Storytelling. *International Journal of Geo-Information*, 10.

Megha, P., Nachammai, L., Senthil Ganesan, T., 2018. 3d game development using unity game engine. *International Journal of Scientific & Engineering Research*, 9, 1354–1356.

Miller, T., Trenum, G., Lieberman, J., 2020. 3D Data Container Engineering Report.

Nam, D., Lee, D., Lee, S., Kwon, S., 2019. Performance Comparison of 3D File Formats on a Mobile Web Browser. *International Journal of Internet, Broadcasting and Communication*, 11(2), 31–42.

Padsala, R., Fink, T., Peters-Anders, J., Gebetsroither-Geringer, E., Coors, V., 2020. From urban design to energy simulation - a data conversion process bridging the gap between two domains.

Reed, C., Belayneh, T. (eds), 2021. *OGC Indexed 3d Scene Layer (I3S) and Scene Layer Package (*.slpk) Format Community Standard Version 1.2*. Open Geospatial Consortium, Wayland, MA, USA.

Reitz, T., Schubiger, S., 2014. The Esri 3D city information model. *IOP Conference Series: Earth and Environmental Science*, 18.

Sihombing, R., Coors, V., 2018. Object-Based Mobile Augmented Reality for a 3D Model. 646-655.

Torres-Ferreyros, C. M., Festini-Wendorff, M. A., Shiguihara-Juárez, P. N., 2016. Developing a videogame using unreal engine based on a four stages methodology. *2016 IEEE ANDESCON*, 1–4.

Vohera, C., Chheda, H., Chouhan, D., Desai, A., Jain, V., 2021. Game engine architecture and comparative study of different game engines. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–6.