International Journal of Geo-Information

MDPI

Article

Open Geospatial Data Integration in Game Engine for Urban Digital Twin Applications

Toni Rantanen [1,*], Arttu Julin [1], Juho-Pekka Virtanen [2], Hannu Hyyppä [1] and Matti T. Vaaja [1]

[1] Department of Built Environment, School of Engineering, Aalto University, FI-00076 Aalto, Finland; arttu.julin@aalto.fi (A.J.); hannu.hyyppa@aalto.fi (H.H.); matti.t.vaaja@aalto.fi (M.T.V.)
[2] Forum Virium Helsinki, FI-00130 Helsinki, Finland; juho-pekka.virtanen@forumvirium.fi
* Correspondence: toni.rantanen@aalto.fi; Tel.: +358-50-348-2172

Abstract: Increasing interest has recently been shown towards the digital twins of cities. These urban digital twins utilize 3D city models together with other data sources, such as open data, which can be accessed, e.g., through application programming interfaces (APIs). A large amount of open geospatial data is already available through various open data portals and APIs, such as REST and MQTT. Game engines, on the other hand, have been seen as highly potential platforms for urban digital twins, with recent development increasing the useability of geospatial data in game engines. This study explores the possibilities of game engines for urban digital twins by integrating and visualizing API-based open geospatial datasets with a 3D city model data set in a game engine environment. Three open data integration approaches are demonstrated, and the performance of these approaches is tested for urban digital twins. Based on the results, modern game engines offer a great platform for creating urban digital twin applications; however, some problems and limitations must be overcome before the use of geospatial data in game engines becomes feasible on a larger scale.

Keywords: geospatial; digital twin; game engine; open data; 3D city modeling; spatial data integration; 3D data visualization

check for updates

Citation: Rantanen, T.; Julin, A.; Virtanen, J.-P.; Hyyppä, H.; Vaaja, M.T. Open Geospatial Data Integration in Game Engine for Urban Digital Twin Applications. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 310. https://doi.org/10.3390/ijgi12080310

Academic Editor: Wolfgang Kainz

Received: 28 April 2023
Revised: 30 June 2023
Accepted: 27 July 2023
Published: 28 July 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

A 3D city model is a digital model representing the geometry and structure of an urban environment [1]. Numerous cities have produced their own 3D city models, e.g., Helsinki [2], Rotterdam [3], and Berlin [4], of which many have been made available as open data to encourage utilization and application development. These openly available city-wide urban 3D models are typically relatively visually and geometrically simple (Figure 1) as they often follow the CityGML standard, emphasizing semantic data and structure instead of visual detail. Some cities have also released photogrammetric mesh models that are better suited for visualization use to complement CityGML models [5]. In some few cases, models with even higher visual quality have been made openly available but with very local city block type scale [6]. As 3D city models are becoming increasingly available, especially as open datasets, it has become increasingly more important to also research and develop their utilization. Three-dimensional city models have been connected to the "smart city" concept [7] and the development of urban digital twins.

While the concept of a digital twin was already introduced in 2002 [9], no consensus has been reached on its definition yet. However, they are generally described as digital representations of physical systems. Their main uses have originally been in the manufacturing industry but have since extended also to the geospatial domain. According to a recent report by the World Geospatial Industry Council (WGIC) on spatial digital twins, most of the advanced digital twin use cases utilize spatial data [10]. Increasing interest has been recently shown especially towards the digital twins of cities. These so-called urban digital twins are typically seen as extensions to 3D city models, containing additional

attributes and properties, such as semantic data and real-time sensor data [11,12]. They can be utilized in various city-related operations, e.g., decision making, urban planning, logistics, and construction [13]. Examples of urban digital twin projects include Helsinki [2], Rotterdam [3], and London [14].



Figure 1. Examples of openly available photorealistic 3D city model datasets. A common citywide CityGML-type model (left, image courtesy of the City of Helsinki) [8] and a local city block-scale model with higher level of detail (right) [6].

Urban digital twins utilize 3D city models together with other data sources, such as open data, which can be accessed, e.g., through application programming interfaces (APIs). A wide range of perspectives are used for defining APIs from technical to broader concepts, but on a general level, they provide the means for software to communicate and exchange information with each other [15]. In the context of urban digital twins, this essentially means that different data providers can make their geospatial data available via APIs for use with various urban digital twin applications. A large amount of open geospatial data is already available through various data portals, such as Helsinki Region Infoshare (HRI) [16], Barcelona's City Hall Open Data Service [17] and open data portals of New South Wales [18] and the U.S. government [19]. These portals contain data sets with varying formats, also via APIs, such as REST (representational state transfer) [20] and MQTT (MQ telemetry transport) [21]. For example, the City of Helsinki provides numerous geospatial datasets as open data, such as urban tree database [22], metropolitan area service map [23], and two different 3D city models covering the entire city area [24]. These data sets often come with varying temporal resolutions depending on their intended purpose, ranging from static to near real-time updated data. For example, the temporal resolution of a city model can, in the worst case, be several years, while other open data sources, like public transport location data [25], can be updated every few seconds.

Many of the most common geospatial data platforms for urban digital twins are browser-based, such as CesiumJS [26], Mapbox [27], ArcGIS Maps SDK for JavaScript [28], and Google Maps Platform [29]. Their strengths lie in their accessibility and comprehensive support for geospatial data; however, their visualization capabilities and customization options can be seen as limited [30]. Game engines can be seen as emerging alternative platforms for urban digital twins. They offer versatile tools for building functional and interactive digital environments with high visual quality and powerful real-time rendering. Compared to browser-based implementations, e.g., [31] they often have higher system requirements and require installations but offer more functionalities for generating interactive and visually immersive experiences. In addition, they are an integral part in the making of virtual reality (VR) and augmented reality (AR) implementations. An overview of the strengths and weaknesses of browser- and game-engine-based platforms are given in Table 1.

Table 1. An overview of the strengths and weaknesses of browser- and game-engine-based platforms for urban digital twin applications [30,32–34].

Platform	Visual Quality	Interactivity	Accessibility	Geospatial Data Support
Web browser	Low	Medium	High	High
Game engine	High	High	Low	Low

Game engines have already been used extensively for visualizing real-world-based 3D data [35–37]. Unfortunately, their support for geospatial data is very limited, as they operate in their own local coordinate system, which makes georeferencing a rather difficult task [32]. Game engines are also somewhat limited in their ability to handle real-time data streams [38]. However, recent game engine development has streamlined the use of geospatial data in game engines to a degree by providing solutions for georeferencing through dedicated georeferencing plugins, origin-rebasing methods, and 64-bit float precision support, but also via the ArcGIS Maps SDK [39] and Cesium for Unreal Engine [40] and Unity [41] plugins. The plugins for Unreal Engine and Unity are especially promising, essentially integrating the Cesium Ion cloud service with the popular game engines, providing a real-world coordinate system via a full-scale WGS84 (World Geodetic System 1984) globe directly in the engine, in addition to web browsers [42]. Cesium is a popular web-based 3D platform, consisting of a virtual globe for visualizing and creating web applications that utilize geospatial data. It has been widely used by many cities, such as Helsinki [43]. While the development is still ongoing, and has its own limitations, it offers new and more manageable ways of utilizing geospatial data directly within game engine environments.

While game engines have been extensively used for static real-world 3D data visualization purposes, the added scope of integrating multiple open geospatial datasets with varying temporal resolutions, both static and near-real-time, by utilizing APIs in a game engine environment, has been explored to a much lesser extent. However, some similarities can be found in various publications. For example, Würstle et al. [30] have an urban digital twin example using Open Geospatial Consortium (OGC) standards and static sensor data through the REST API. However, constantly updating real-time data was not used. From a real-time-data integration perspective, Kuts et al. [44] utilized the Unity game engine and MQTT-protocol for synchronizing a physical factory robot with its digital twin, but its context lies within local-scale industrial applications rather than city-wide urban digital twins. From a geospatial perspective, Lee et al. [33] created a planetary-scale platform with Unity game engine, which can be used for urban digital twins, however only terrain and buildings were implemented. Purely from a visualization perspective, Laksono and Aditya [32] utilized the Unity game engine to visualize topographic data with 3D building models. Maulana and Kanai [45] developed an interactive visualization for agriculture commodity selection and land evaluation purposes with the Unity game engine and large-scale geospatial data. Furthermore, an increasing number of game-engine-based urban digital twin projects exist globally, e.g., in Shanghai [46], Wellington [47], Amsterdam [48], Paris [49], and Helsinki [50]. However, game-engine-based studies that combine the utilization of open geospatial data, demonstration of integration approaches for open data of various temporal resolutions, and visualization of these static and real-time data sets are limited in number. A recent study by Lei et al. [51] has also highlighted data integration as one of the major remaining challenges with urban digital twins.

The aim of this study is to integrate and visualize openly available city-wide geospatial datasets with open 3D city model data in a game engine environment. To achieve this, three data integration approaches are demonstrated by utilizing three types of open geospatial data sources via APIs, available game engine plugins, and custom scripting in a game engine. Also, the performance between these data integration approaches for urban digital twins is tested. Furthermore, techniques to visualize the integrated static and dynamic open geospatial data sets are explored. Additional focus is given to the state of georeferencing

of integrated geospatial data sets in the game engine as it has previously been seen as troublesome. Due to the focus being on the integration of city-wide open geospatial data, the quality aspects, such as visual fidelity and geometric accuracy, is of lesser importance in this study.

2. Materials and Methods

2.1. Open Geospatial Datasets

In this study, four different openly available datasets were used in the data integration, including the city information model of Helsinki; HRT (Helsinki regional transport) high-frequency positioning data; MyHelsinki Open API—places, events, and activities; and the urban tree database of the City of Helsinki. Datasets are listed in Table 2 below.

Table 2. The four datasets used in the data integration with short descriptions of their type and content.

Dataset	Type	Description
The 3D city information model (Section 2.1.1) [8]	Textured 3D model files in OBJ file format [1]	City information model based on the CityGML standard
Public transit dataset (Section 2.1.2) [25]	MQTT-based API	Real-time positioning and attribute data of public transport in the metropolitan area
City events and activities dataset (Section 2.1.3) [52]	REST-based API	Contains coordinate and attribute data of events in Helsinki
Urban tree dataset (Section 2.1.4) [22]	WFS-based API [2]	Contains coordinate and attribute data of planted trees in Helsinki

[1] Data converted from CityGML. [2] Data downloaded manually as CSV.

2.1.1. The 3D City Information Model and Hosting Service

The city of Helsinki has two different 3D city models: a semantic city information model and a high-detail reality mesh model. Both models are available for the public through viewing application [8] and as open data [24]. The city information model of Helsinki is built following the CityGML standard [53] and contains two levels of detail (LoD) for building models. These models are generated using the city plan's base-map-based building footprints, airborne laser scanning, and oblique aerial imaging [54]. According to Ref. [55], the city information model can be used to perform different analyses, e.g., on energy consumption. In this study, only the CityGML-based 3D city information model was utilized, as the CityGML standard is open and popular and model data sets based on the standard are commonly available. While higher in visual quality, the reality mesh model does not include buildings as separate model objects, which would have complicated data integration and visualization process.

In our study, hosting and streaming of the city information model of Helsinki is achieved using Cesium Ion [42], which is a cloud-based 3D geospatial data hosting and streaming service. The service includes several curated 3D geospatial datasets, such as Cesium World Terrain [56], Cesium OSM Buildings [57], and Bing Maps Aerial Imagery [58]. In addition, users can upload their own 3D models into the service. Cesium Ion also automatically converts uploaded assets into a 3D Tiles format [59] for more optimized data streaming

2.1.2. Public Transit Dataset

Helsinki regional transport (HRT) offers real-time WGS84 (EPSG:4326) location and attribute data about public transit (bus, train, tram, metro, and ferry traffic) in the Helsinki metropolitan area through a dedicated API. The API is based on MQTT [21], which is a lightweight publish–subscribe messaging protocol commonly used for the Internet of Things (IoT). The API, once subscribed, provides a continuous data stream in JSON

(JavaScript Object Notation) format, making it useful for data sets with higher temporal resolutions. In this study, the data is used to simulate real-world public transit in near real-time in a game engine environment. From the public transit types, only buses were selected for the simulation due to their quantity and high activity.

2.1.3. City Events and Activities Dataset

MyHelsinki Open API offers an API for real-time WGS84 (EPSG:4326) location and attribute data about events happening in the Helsinki area. The API is REST-based, which means it is accessible with HTTP queries [20]. It is more suitable for data sets with lower temporal resolution, e.g., once a day. In this study, the data is utilized for visualizing different events and their information within the game engine environment.

2.1.4. Urban Tree Dataset

Helsinki Region Infoshare offers a database of planted trees in the streets and parks of Helsinki [22]. The data contains the coordinates in the ETRS-GK25 (EPSG:3879) coordinate system and other attribute information for around 50,000 trees, including a large number of different tree species. The data is updated irregularly and does not include plot-based trees or forest data. It is offered through WFS [60] and WMS [61] interfaces and can be downloaded in CSV, ESRI SHAPE, JSON, KML, and XML formats. In this case, the data was downloaded manually in CSV format with a file size of 8.39 MB and used as static data for populating the game engine environment with urban trees.

2.2. Game Engine Platform

Unreal Engine (Version 4.26.2) is a versatile commercial game engine which, in addition to video games, is commonly used for architecture, film and television, broadcast and live events, animation, automotive and transportation, and simulation [62]. Unreal Engine 4.26.2 was selected due to its better compatibility with all necessary plugins during development. Unreal engine utilizes a visual scripting system called Blueprint, in addition to traditional scripting in C++. Visual scripting essentially allows users less versed in traditional programming languages to utilize Unreal Engine for application development. Besides the game engine itself, several plugins for the game engine were utilized for data integration. The utilized Unreal Engine plugins and their roles are listed here in Table 3.

Table 3. The Unreal Engine plugins used in the integration process with their roles.

Plugin	Role
BP MQTT [63]	Communication between the game engine and MQTT API.
VaRest [64]	Communication between the game engine and REST API.
UEGeoCoordinates [1] [65]	Coordinate conversion from real world coordinates into game engine coordinates.
JSONParser [66]	Providing tools for managing JSON in Blueprint visual scripting system.
Cesium for Unreal [40]	Providing a 3D geospatial ecosystem through a full-scale WGS84 globe and the ability to stream assets from Cesium Ion cloud service into the game engine.

[1] Integrated into Unreal Engine from version 4.27 forward.

2.3. Data Integration in Game Engine

Data integration in Unreal Engine was carried out utilizing the aforementioned plugins and datasets together with the game engine's in-built tools and functionalities and the blueprint visual scripting system. Cesium for Unreal plugin was used as the basis for the integration, as it provides a global-scale, high-accuracy WGS84 globe inside Unreal Engine and the ability to stream 3D-tiled 3D models contained in Cesium Ion cloud service into the game engine. The Cesium World Terrain (DTM) asset and Bing-Maps-based aerial imagery, included in Cesium Ion, were was to supplement the 3D city information model of

Helsinki. More specifically, the 3D city information model of Helsinki contained LoD2-type textured building models of CityGML 2.0 standard [53]. Additionally, the georeferencing components provided by the plugin were utilized for spatial-rebasing operations to form a local origin for the game engine environment to prevent problems caused by decimal precision. An overview of the data integration pipeline is provided in Figure 2. In addition, some minor visual enhancements were made to the environment by implementing realistic sun, clouds, and weather using various game engine tools.

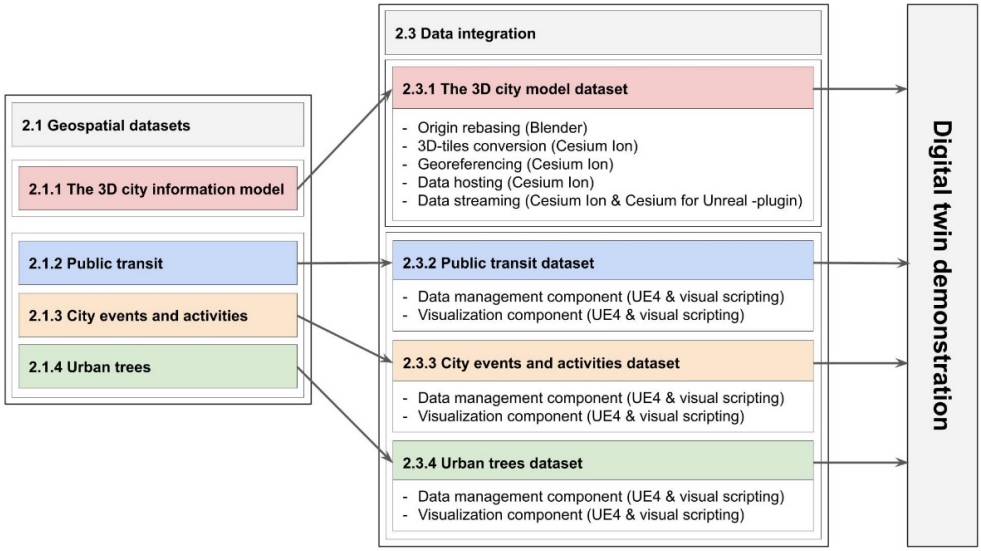


Figure 2. An overview of the data integration process, with the integration of the 3D city model dataset color coded in red and the three data integration approaches in blue, yellow, and green.

2.3.1. Integrating 3D city Models

For enabling the 3D city information model in the game engine, manually selected parts of the 3D city information model of Helsinki were downloaded from the open download service [8] in OBJ format, including model textures in PNG format with the file size totaling ~207 MB. The area formed by the selected parts was limited to cover only the city center, containing 361 buildings (Figure 3), as the download service has a limit on the size of the area that can be downloaded at once via manual selection. The downloaded data set was processed in Blender (version 2.8) to rebase it closer to the origin as Cesium Ion requires models to be uploaded in a local coordinate system with geometry centered around the origin [67]. The rebased model was exported from Blender in OBJ format with PNG textures and uploaded into Cesium Ion (see Section 2.1.1). Cesium Ion automatically converts the uploaded dataset into 3D Tiles format [59] for more efficient data streaming. After uploading the model, manual georeferencing, based on visual alignment, was performed in Cesium Ion to bring the dataset approximately into the correct location on the WGS84 globe. The resulting data set could then be accessed and utilized directly in Unreal Engine by streaming it from Cesium Ion using the Cesium for Unreal plugin.

2.3.2. Game Engine Integration of Public Transit Dataset (MQTT API)

To make the use of public transit positioning data possible in Unreal Engine, a data management and public transit visualization component (i.e., "blueprints") were constructed using the visual scripting system. The data management component was made with the help of the pre-made scripts provided by the BP MQTT plugin and was designated to handle the connection between Unreal Engine and the MQTT API, subscribing to topics in the API, parsing the received JSON data using the JSONParser plugin, and converting the received coordinates into game engine coordinates via the UEGeoCoordinates plugin. An example of the implemented data management component is shown in Figure 4.

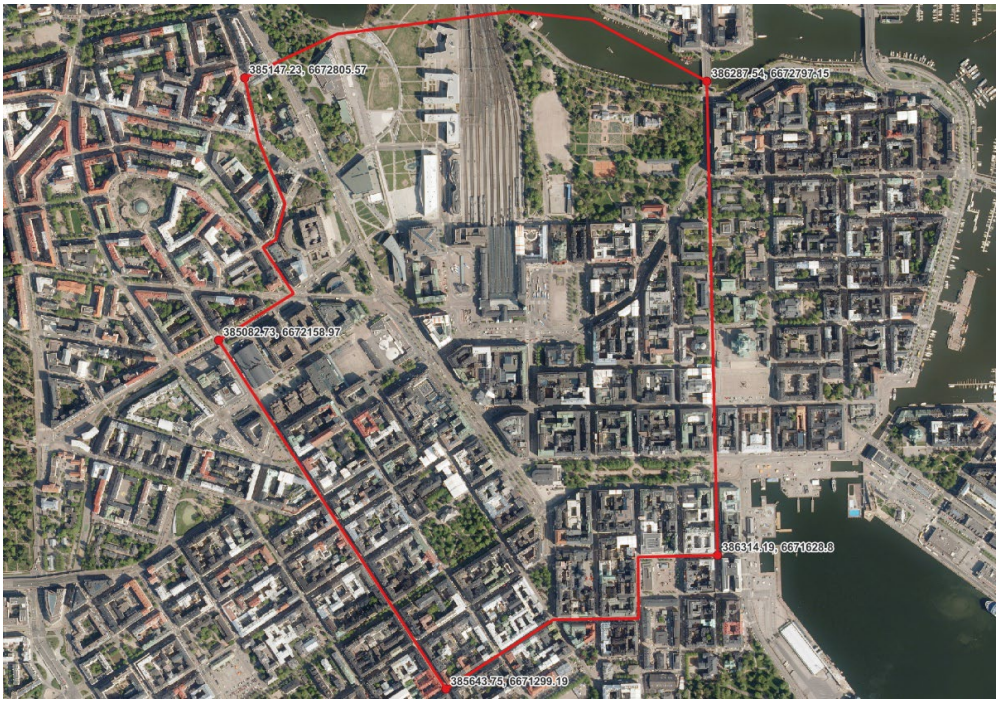


Figure 3. The 3D city information model area of interest with size of approximately 160 ha, containing 361 buildings (Coordinates in ETRS89/TM35FIN, EPSG:3067).

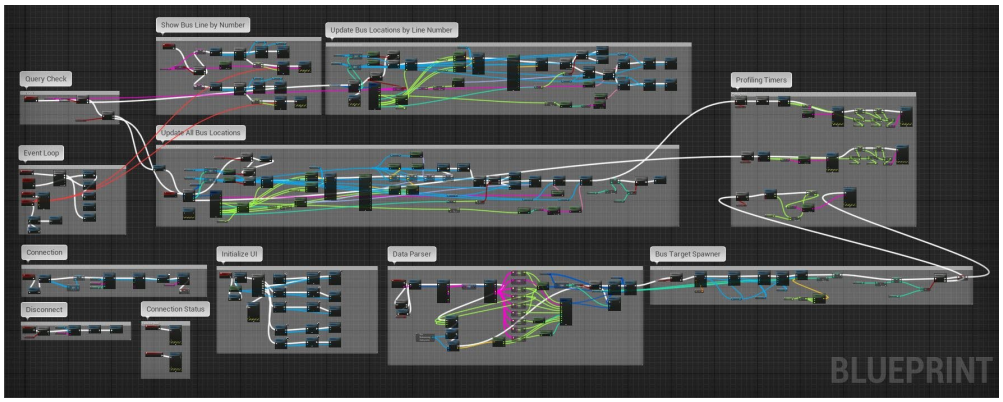


Figure 4. An example of the implemented data management component created using visual scripting.

The coordinate conversion was accomplished by first converting the received coordinates into a projected coordinate system (ETRS89/TM35FIN, EPSG:3067) and then into game engine coordinates. In addition, the data management component was assigned with spawning and data update logic for individual public transit visualization components (buses in this case). The public transit visualization component contains a 3D model of a bus in addition to managing the internal logic of the bus object, based on the constantly updated location and attribute data. To enhance the level of visualization, the buses were manually 3D modeled with animated doors based on their real-life counterparts and to reflect the state of dynamic data. An overview of the data management and public transit visualization components is shown in Figures 5 and 6 respectively.

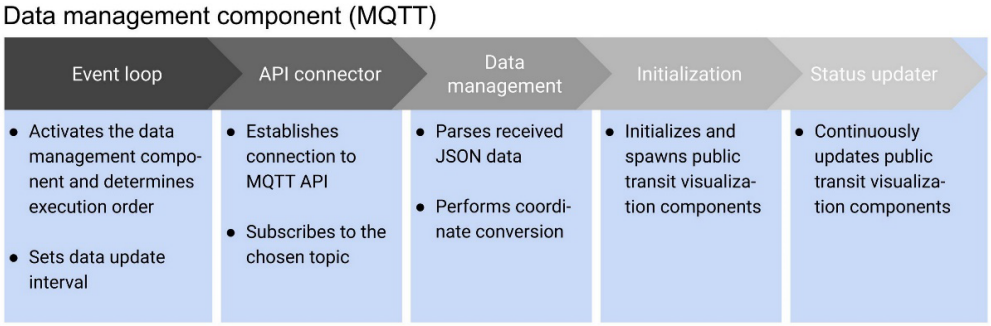


Figure 5. Overview of the data management component and its main tasks (MQTT).

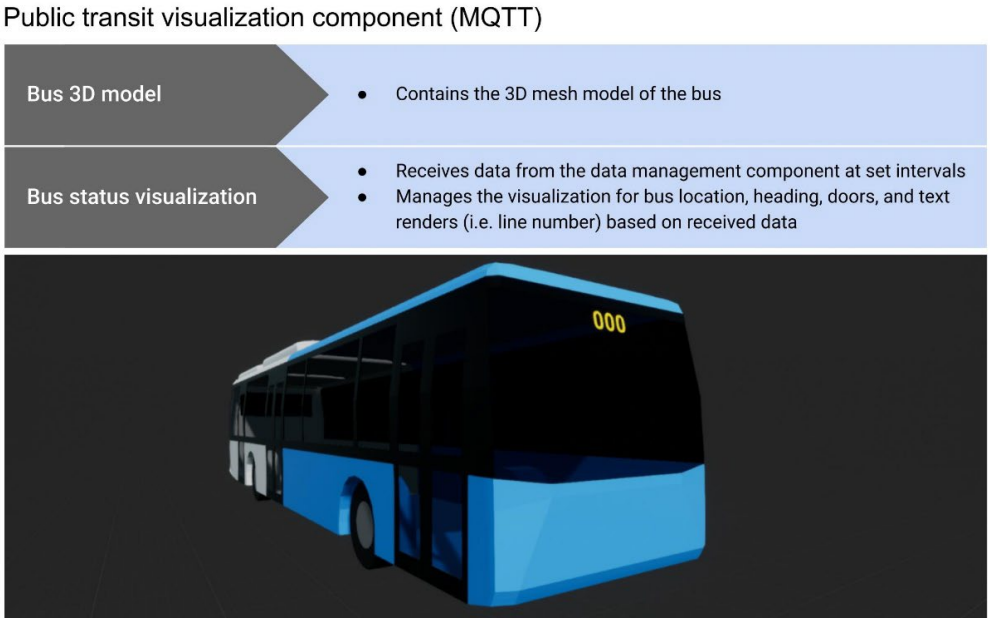


Figure 6. Overview of the public transit visualization component with a 3D model of a bus made to visualize key attributes in the public transit dataset. These attributes include location, heading, speed, door status, and line number. One visualization component is generated for each bus.

2.3.3. Game Engine Integration of City Events and Activities Dataset (REST API)

Integration of the city event data was implemented by generating a data management and an event visualization component. In addition, two user interface elements containing an event marker and an information panel were created for visualization purposes. The data management component was made and is responsible for managing the connection between Unreal Engine and the MyHelsinki Open API and parsing the received JSON object data with the help of pre-made scripts provided by the VaRest plugin. In a similar fashion to the MQTT approach, the component also included coordinate conversion into game engine coordinates via the UEGeoCoordinates plugin, together with spawning and data update logic for individual event visualization components. The event visualization component was assigned with managing and updating the event marker and information panel used for data visualization purposes based on the received data. The event marker was generated to provide an indicator of the event locations, while the information widget was generated to allow the user to see more information about the event when looking towards the event marker from within a set distance. Concise depictions of the data management and event visualization components can be seen in Figures 7 and 8 respectively.

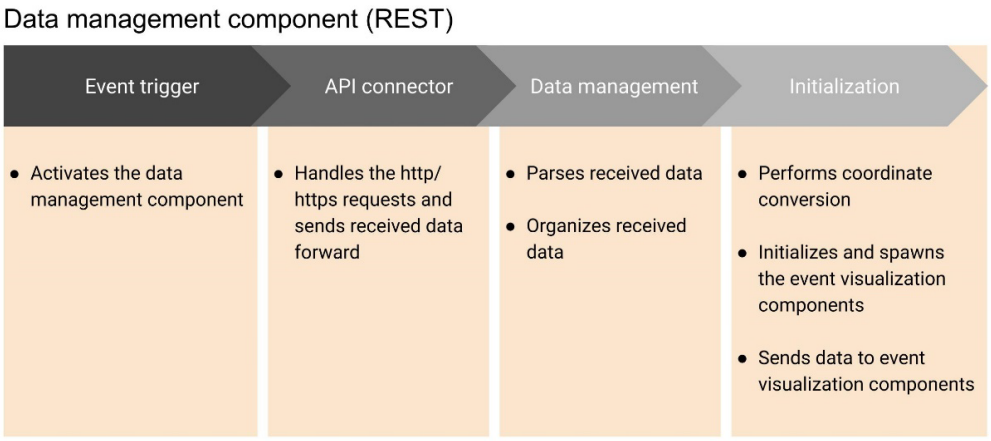


Figure 7. Overview of the data management component and its main tasks (REST).

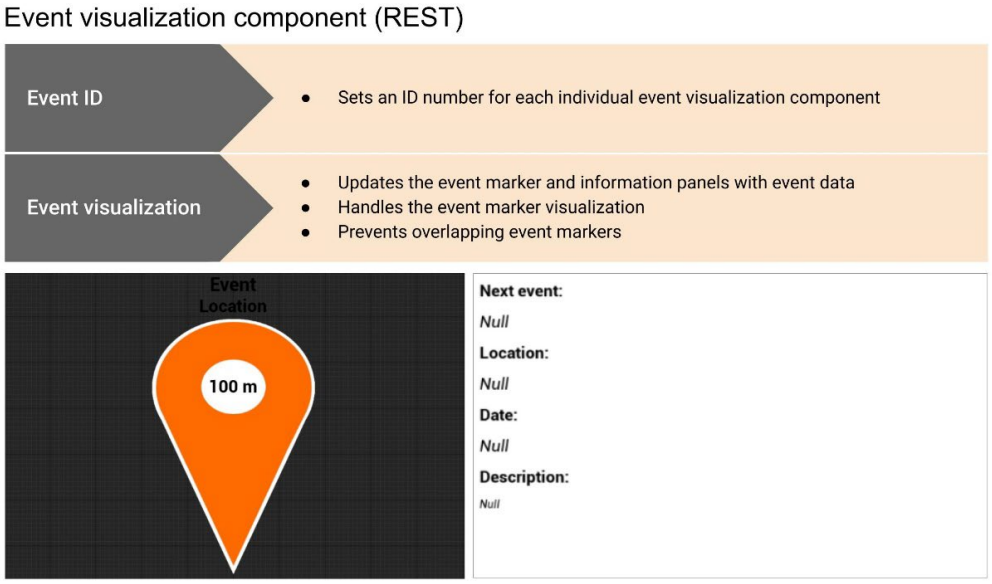


Figure 8. Overview of the event visualization component with the event marker and information panel visualizations. One visualization component is generated for each event.

2.3.4. Game Engine Integration of Urban Tree Dataset (Static Data as CSV)

Urban tree data of the City of Helsinki was downloaded as a CSV file from a WFS-API, and a corresponding struct template (i.e., data table) was generated in Unreal Engine, which allowed the CSV file to be imported directly into the game engine. Similarly, to the other approaches, a data management and a tree visualization component were created, as well as one user interface element containing an information panel. The data management component was assigned to read the data contained in the structs and to convert the real-world coordinates into game engine coordinates using the UEGeoCoordinates plugin (see Figure 9). In addition, the data management component was also set to define a unique ID for each individual tree visualization component and to populate the game environment with said components (i.e., trees). The aforementioned tree visualization component (see Figure 10) was made for individual tree objects, consisting of a 3D mesh model of a tree (assets downloaded from Unreal Engine Marketplace, with model file sizes ranging from 1 to 4 MB) and logic for tree species and size selection based on the data within the data table. Tree sizes were derived from their diameter value, as the data did not contain height information. Additionally, the component was assigned with sending relevant tree information to the information panel. The information panel was set to show tree information when the user is looking at the tree within a certain distance.

Concise depictions of the data management and tree visualization component can be seen in Figures 9 and 10, respectively.

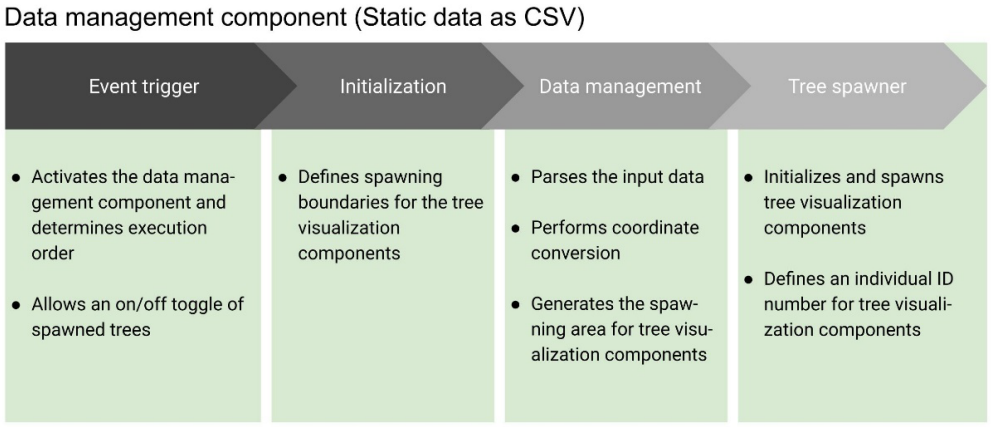


Figure 9. Overview of the data management component and its main tasks (CSV).

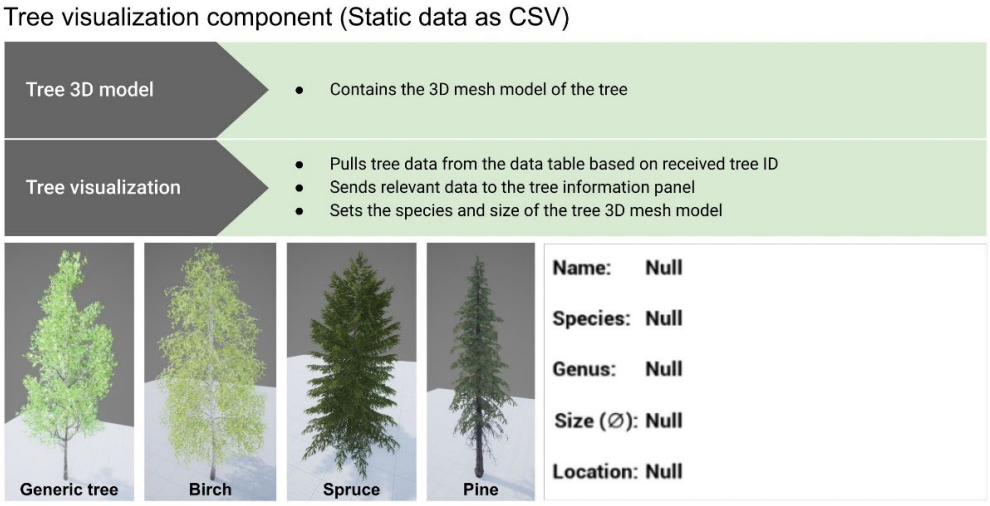


Figure 10. An overview of the tree visualization component with tree 3D mesh models and the tree information panel. Four different tree models were used for visualization: a generic tree, birch, spruce, and pine. Tree size scaling is based on the tree diameter.

2.4. Performance Tests

The performance of different data integration approaches was tested based on their execution times. In Unreal Engine, a simple profiling timer was set up via visual scripting to measure connecting time to the APIs and the time it takes for the game engine to parse and spawn the received data. Additionally, in the case of MQTT, a timer was also set up for measuring data update times due to the near real-time nature of the data. In addition to the execution time measurements, the frame rate (FPS) of the implementation was monitored during the tests. The hardware used for these tests was a high-end gaming PC running Windows 10 OS with Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz, Nvidia GeForce RTX 2080 Ti (11GB) GPU, and 64.0 GB of RAM.

2.4.1. Parsing and Spawning Time Measurements

For obtaining sufficient results on parsing and spawning times, several different data sample sizes were tested, looking for any correlation between measured times and the amount of data. The sample size ranges differ based on the source data, as they contain varying maximum values with 800, 2600, and 50,000 data samples for public transit (MQTT), city events and activities (REST), and urban trees (CSV), respectively. A denser

sample set was gathered on the lower end of the sample size range, from 100 to 800 with intervals of 100, to make a direct comparison between integration approaches possible. The measurements were conducted multiple times, and an average value was calculated to address outliers in the measured time values.

2.4.2. Updating Time Measurements for Real-Time Data (MQTT Only)

Due to the real-time nature of the MQTT approach, additional measurements were taken for the data update times based on the sample size. The update times were recorded from a sample size of 100 up to 800, with intervals of 100. In a similar way to the parsing and spawning time measurements, multiple measurements were taken to eliminate outliers in the measurements.

2.4.3. Frame Rate Measurements

FPS readings were taken with the same static view of the scene to gain comparable results (FPS readings are expected to be lower when moving in the scene). Maximum frame rate was capped to 120 fps, which was the idle reading with only the city model active. As the data sets were varying in size (see Section 2.4.1), the measurements were focused on the lower end of the sample size range to achieve more comparable results. However, additional measurements were taken for the static data-based approach due to the larger size of the dataset. The frame rate was considered satisfactory for desktop and VR use with values over 30 and 90 FPS respectively.

3. Results

3.1. Digital Twin Demonstration

The implementation consists of a 3D city-model-based game engine environment with a functioning real-world coordinate system using WGS84 (Figure 11). The information city model of Helsinki was streamed from Cesium Ion cloud service via the Cesium for Unreal plugin, which appeared to work with no major technical or performance-related issues. The streamed 3D city model also works with typical game engine simulation systems, such as physics and lighting. The global satellite imagery (Bing Maps, Microsoft Inc., Redmond, WA, USA) provided with Cesium Ion was also integrated with the 3D city model data, providing textured digital terrain model not included in the CityGML-based city model.



Figure 11. View of the 3D city model of Helsinki in Unreal Engine, streamed through the Cesium Ion plugin.

Three different data integration approaches were experimented with for enriching the game engine environment with geospatial data. These included public transport (MQTT API), city events and activities (REST API), and urban trees (static data as CSV).

3.1.1. Public Transit (MQTT API)

The implemented MQTT approach features public transit buses in their real-world location in near real time, with an update interval of two seconds (Figure 12). Location data was interpolated over the two-second update window to achieve smoother bus movement between two consecutive locations.



Figure 12. View of the implementation with near-real-time updated public transit embedded in the 3D city model of Helsinki in Unreal Engine.

In addition to location, the public transit components receive information about their speed, heading, door status, and line number, which are visualized using dynamic object color, object rotation, dynamic door movement, and an attached text component, respectively (Figure 13). A query based on the bus line number was also implemented, which allowed only buses of a selected line to be shown at a time.

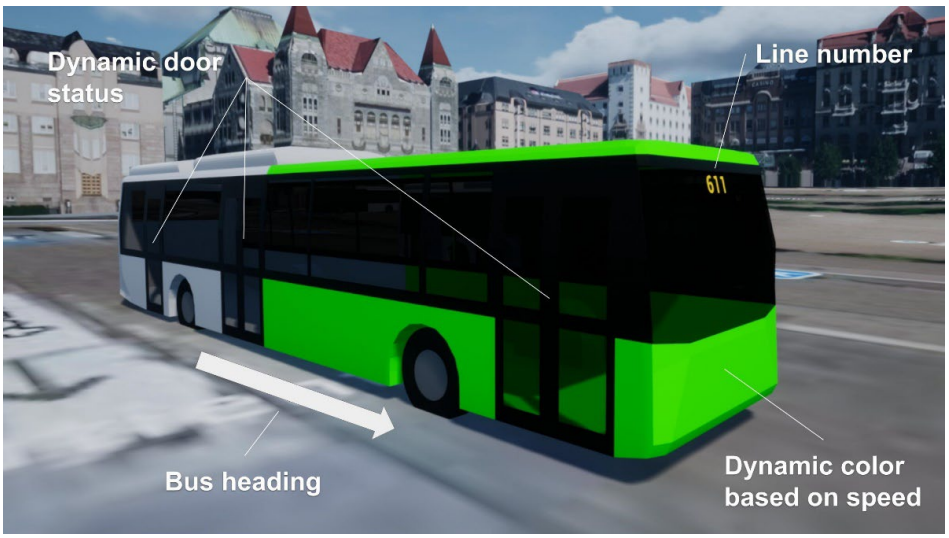


Figure 13. Three-dimensional model of the bus with visualizations based on received attribute data. Line number reflects the actual bus line number, doors dynamically open and close, and the front coloring (blue part) changes based on current speed. Additionally, the bus is rotated based on its heading attribute.

3.1.2. City Events and Activities (REST API)

The second is a REST-based implementation that visualizes location-tied event information by using user-facing 2D markers embedded in the 3D city model-based game engine environment. For each marker, a UI element pops up, which describes the name, date, address, and a short description of the next event happening in the marked location when the user looks towards the marker from a close enough distance (Figure 14). Additional functionality includes a distance counter, which shows the distance to each marker in real-time.

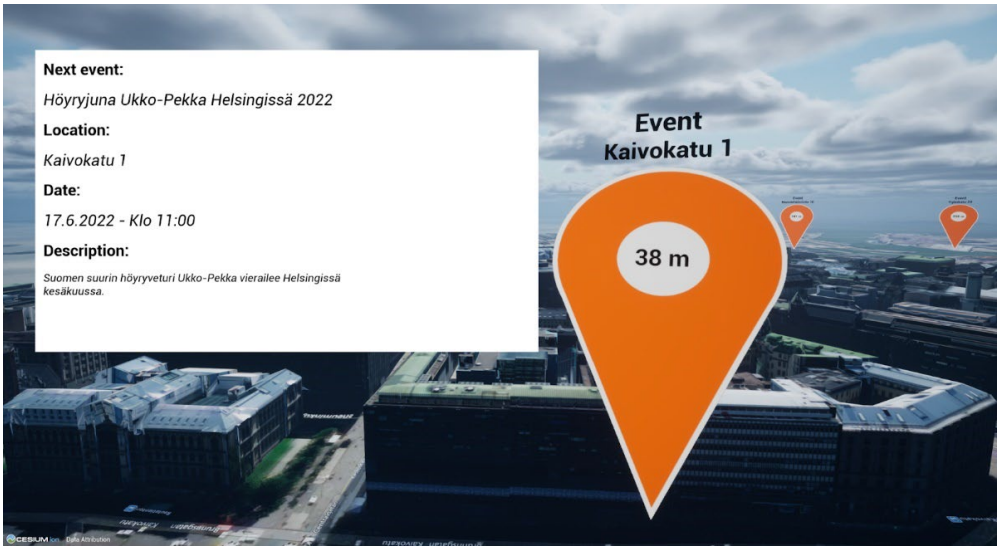


Figure 14. View of the implemented event markers containing information such as names and dates of the events happening in different real-world locations. The event describes the visit of a steam locomotive Ukko-Pekka in Helsinki during the summer of 2022.

3.1.3. Urban Trees (Static Data as CSV)

The third is a static data-based implementation that visualizes trees in the game engine environment relative to their real-world location, with individual tree species and size determined by the input data (Figure 15).



Figure 15. View of the city model-based game engine environment embedded with visualized data from the tree register.

Tree height was, however, derived from the tree diameter and is, therefore, only an estimation (no direct height data was available). Also, due to the vast amount of different tree species contained in the data, only a few different tree models were selected for the visualization: mainly birches, spruces, and pines (the most dominant tree species in Finland), while a generic tree model was used for all the other trees. In addition, a gaze-based interaction was implemented, which allows users to receive information from the targeted tree via a pop-up interface (Figure 16). A spawning radius setting was also included that could be set to make trees spawn only within a selected distance from the user.

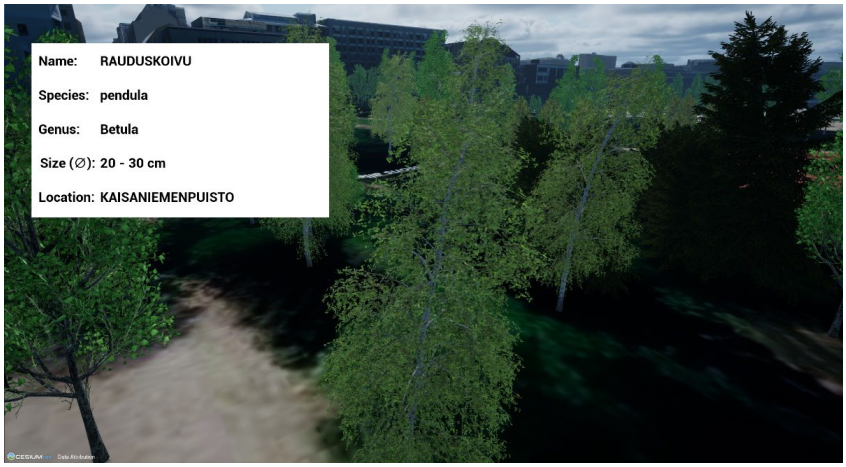


Figure 16. Pop-up interface showing information about the individual targeted tree based on its attributes in the tree register. Tree size is the diameter of the target tree.

3.2. Performance Tests

3.2.1. Parsing and Spawning Time Measurements

The parsing and spawning times for different data integration approaches is shown in Figure 17. Sample sizes and measured times appear to have a linear correlation. Performance-wise, MQTT and static-data approaches seem to be comparable, with static data being slightly better with larger amounts of data. On the other hand, the REST-based approach seems to perform significantly worse, being approximately ten times slower compared to the static-data approach. In the MQTT- and REST-based approaches, the time to form a connection to the APIs was measured to be less than a millisecond for MQTT and between 2 to 5 s for REST, emphasizing the lower end.

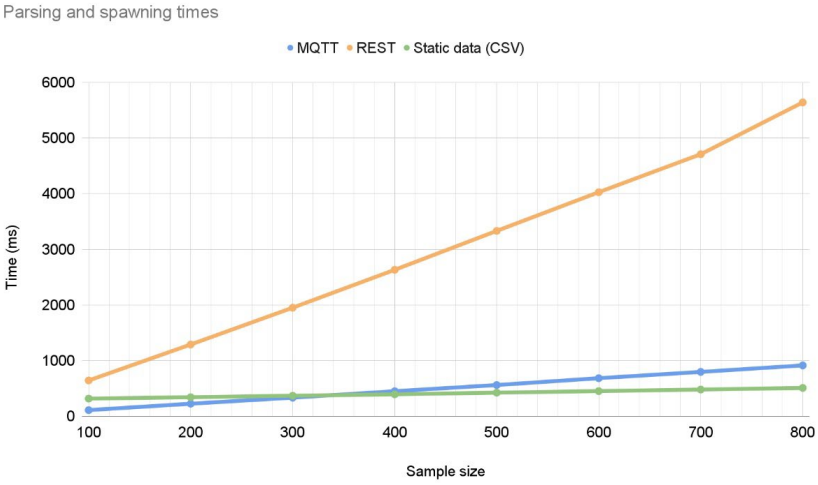


Figure 17. The measured parsing and spawning times for the three data integration approaches.

3.2.2. Updating Time Measurements for Real-Time Data (MQTT Only)

In addition to the parsing and spawning time measurements, updating time measurements were conducted for the MQTT-based approach due to the real-time nature of the dataset. The updating times for the real-time public transit data are shown in Figure 18. The results indicate a linear correlation between sample size and update time.

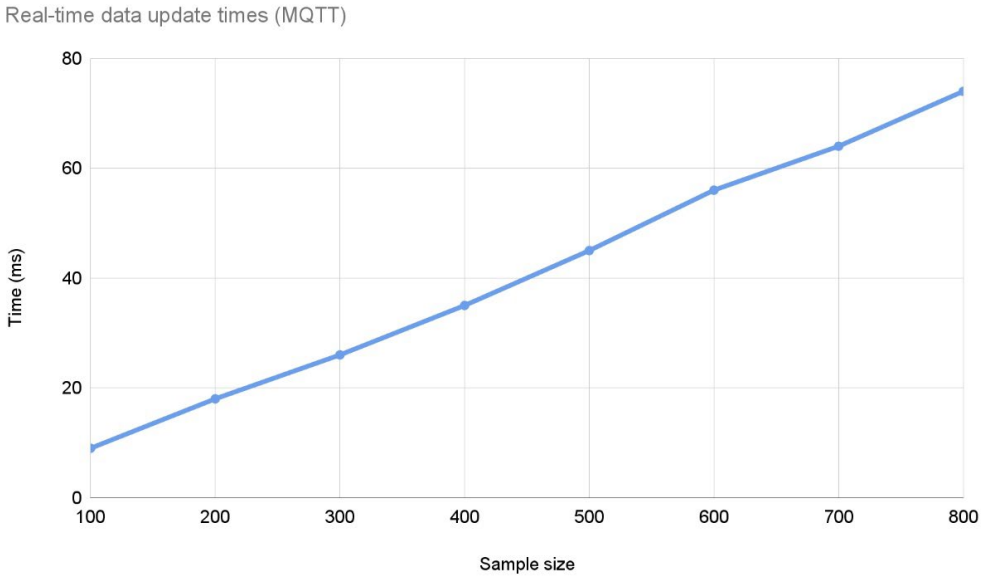


Figure 18. The measured data update times for the near-real-time MQTT approach.

3.2.3. Frame Rate Measurements

Frame rates of the different approaches can be seen in Figure 19. The MQTT- and REST-based approaches follow a similar pattern, with MQTT performing slightly worse with smaller sample sizes. However, the static-data-based approach performs significantly better compared to the other approaches, maintaining a steady frame rate throughout the sample size range. Based on additional measurements (not displayed on Figure 19), the frame rate of the static-data approach starts to drop only after a sample size of 5000, while still achieving a frame rate of 36 at a maximum sample size of 50,000.

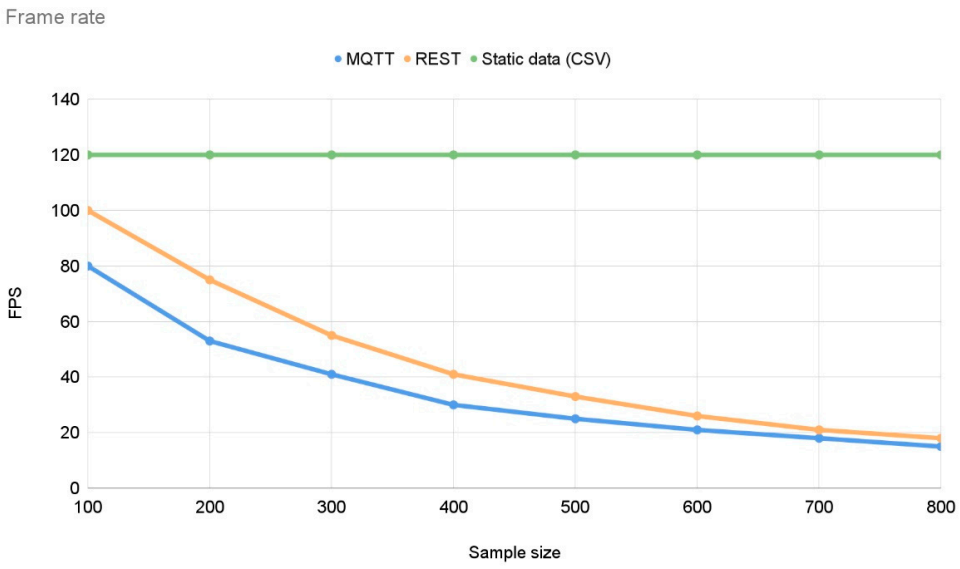


Figure 19. The measured frame rates for different data integration approaches.

4. Discussion

4.1. Digital Twin Demonstration

The 3D model streaming from Cesium Ion, using the 3D Tiles file format, essentially makes the visualization of large-textured 3D city models in game engines less heavy in terms of performance. While streaming the 3D city model into Unreal Engine looks promising, it has some limiting factors. For instance, OBJ models uploaded into Cesium Ion are reduced into a single entity, complicating the implementation of individual building-based interactions. Additionally, most mesh model formats, such as OBJ, do not retain semantic data when converting from CityGML format, which makes displaying semantic data for buildings problematic. However, the recently released 3D Tiles 1.1 (Next) [68] includes support for streaming semantic metadata within 3D models, potentially enabling implementations utilizing semantic data in game engines. Another more universal problem is the lack of direct support for georeferenced models in Cesium Ion, as georeferenced 3D models must be either manually placed, uploaded from supported software, or by using a REST API.

The demonstration was made with data integration as priority and, as such, less attention was paid to the visual quality of the implementation. That said, the resolution of the Bing Maps' aerial imagery, provided with Cesium Ion and the texture resolution of the city information model of Helsinki, are on the lower side, making visualizations from the ground level non-optimal. In addition, the geometry of the LoD 2 model is understandably limited in detail level.

Useability-wise, the frame rate appears to be a limiting factor to the implementation (at least in Unreal Engine 4), restricting the amount of rendered objects. Different display setups also have varying requirements for a sufficient frame rate. Table 4 below shows the approximate maximum number of rendered objects for desktop and VR uses, which is derived from the results. While the FPS values for the MQTT- and REST-based approaches seem low compared to the static-data-based approach, the contained object models are much less optimized (e.g., no pre-made levels of detail), increasing their rendering costs.

Table 4. Maximum amount of rendered objects with different display setups.

	MQTT (Moving Objects)	REST (Static Objects)	CSV (Static Objects)
Desktop (30 fps)	~400	~550	~55,000 [1]
VR (90 fps)	~80	~140	~13,000

[1] Estimated value based on frame rate results (maximum dataset size was ~52,000).

On a general level, the datasets appear to fit visually into the aerial imagery provided via Cesium Ion. However, there are some examples of questionable georeferencing, such as trees located inside buildings or buses drifting out of road areas. These georeferencing errors can originate, for example, from insufficient coordinate accuracy of the source datasets, GNSS blind spots in urban canyons (especially relevant for buses), or even Unreal Engines decimal rounding due to float precision.

4.1.1. Public Transit (MQTT API)

Based on the test results, the MQTT approach seems promising for integrating constantly updating data in near-real-time and, due to fast parsing times, could be used for other data as well. The update times are also quite manageable on a smaller scale; however, they can become problematic with very large datasets, as the update time sets a limit on update frequency. The FPS performance can be seen as problematic, as it limits the number of objects to around 400 for desktop use. The problems involving the FPS with higher amounts of data could be, at least partially, addressed by optimizing the object models as well as utilizing different culling and instancing methods.

The MQTT-based approach should be investigated more in terms of infrequently updated data to better compare its performance against the static-data-based approach. In

addition to its promising performance with near-real-time data, it also has the capability of two-way data transfer, meaning it can also send data. While this feature was not experimented with in this study, it could prove to be a very useful feature especially for digital twins that have been often envisioned to apply two-way communication.

4.1.2. City Events and Activities (REST API)

The results from the performance evaluation suggest that the REST-based approach is more suitable for infrequently updated data, as its parsing and spawning performance is relatively poor, even with rather limited amounts of data. The FPS is comparable to the MQTT approach and would likewise benefit from model optimization and culling methods to ease the graphical load. The aforementioned issues make it a less applicable approach for urban digital twin applications due to the large-scale nature of city-wide data sets. However, it could be potentially used for a more local scale implementation, such as requesting data manually within the "game" to gain relevant information from, e.g., certain areas of interest instead of the whole city. Spatially limited queries are available in certain APIs and have also been implemented in viewer applications [69], proving the functionality of this approach. Further development of the tools for communication between game engine and REST APIs could also prove worthwhile, as REST-based APIs are one of the more common existing APIs currently.

4.1.3. Urban Trees (Static Data as CSV)

Based on the performance evaluation, the static-data approach appeared fastest in handling larger sample sizes, which makes it a more ideal solution for managing large, but more infrequently updated, datasets. It is also the most performance-friendly when it comes to an achievable frame rate and allows significantly more objects into the scene compared to the MQTT and REST approaches. The FPS difference can be explained to some degree by the much greater optimization level of the used-tree models, reducing graphical load for objects. This could be used as a rough reference when evaluating the increase in performance if model optimization were to be implemented for the MQTT- and REST-based approaches. Visual accuracy of the implementation could be enhanced by generating corresponding tree models for each tree type in the database. Additionally, tree height could be adjusted to reflect real-life situations if tree height information was available through the database. Alternatively, the height information for trees could be retrieved from an external data source, such as ALS (airborne laser scanning). However, implementing this in the current processing pipeline would have made it significantly more complex.

4.2. Future Prospects and Developments

Aspects that would increase the applicability of geospatial data integration into game engines are closely tied to the game engine development, including its plugins, as well as the source data itself. On the game engine side, the most obvious improvements would be performance-related, but to also include higher decimal precision and streamlined support for georeferenced datasets and formats. This is applicable especially to the current UE4 implementation. Higher decimal precision would allow for more precise placement of assets in the game engine environment, while streamlined support would make the integration of multiple different datasets more straightforward. For instance, access to city information models' semantic data, contained within the CityGML format through game engines, would greatly benefit their applicability. One of the more limiting factors encountered in this study was the fact that georeferenced 3D models were not fully supported in Cesium Ion when directly uploading data into the service, thus requiring manual georeferencing. Solving this limitation would increase the useability of georeferenced custom datasets within game engines. Other georeferencing options include placement via a REST API or uploading the model from a supported software, but these options were not tested in this study. The potential of these other options could, however, be investigated in a future

study. Additionally, the demonstration highlights the issues that arise from incomplete support for standardized geospatial interfaces in current game engines. For example, while the tree registry data would have also been available over WFS API, it was utilized as a static CSV file instead. In an ideal case, as much of the data as possible would be directly retrieved from APIs.

The main advantages of using game engines as a platform for urban digital twins are their superior visual quality and visualization possibilities, together with real-time rendering capabilities. Even though the visual fidelity and geometric accuracy of the textured 3D city model data was not in the scope of this research, the city model datasets used in this study contained a fairly simple level of detail, common to openly available city model data sets. From a rendering perspective, a more detailed photorealistic 3D city mesh model could have also been used with the demonstrated data-integration approaches. However, CityGML-based models can be more practical for data visualization as they benefit from having an object structure, e.g., buildings as separate objects. Whereas editing textured mesh models can be cumbersome. In addition, being able to exclude trees from the model significantly helped the integration and visualization of individual trees with higher visual fidelity that can, for example, react to the seasonal changes. Game engines would also allow for the use of data sets and 3D assets with much higher visual quality than what is typically used in the city-modeling domain. Game engines are also the go-to platform when creating VR and AR solutions. Like browser-based implementations, they allow for near-real-time updatability as the API-based datasets in the game engine implementation can be automatically updated when the corresponding dataset is updated in the API. Game engines and their plugins, alongside Cesium Ion, are constantly being developed, potentially fixing some of the problems encountered in this study. Additional functionalities are also being introduced, and support for more geospatial data is to be expected. As an example, Cesium for Unreal plugin is receiving support for point clouds, allowing point cloud streaming from Cesium Ion to Unreal Engine (which already supports the use of point clouds). While not utilized in this study due to compatibility concerns during development, Unreal Engine's newest version (Unreal Engine 5) features the Nanite virtualized geometry system, which enables higher geometric complexity and an automatic level of detail adjustment, potentially allowing for the use of more detailed 3D city models. In addition, the change to 64-bit-precision allows for, e.g., greater coordinate accuracy for geospatial assets [70].

From a source data point of view, the main area of improvement would be in 3D city model optimization, to improve its level of detail, texture, data structuring, and data unification. There are problems with, e.g., overlapping 3D city model data and mediocre texture quality. In particular, mediocre texture quality restricts their use for ground level applications (i.e., first person view). For high visualization quality, city mesh models present a feasible alternative. As a recent development, Google has opened the use of their photorealistic mesh models as 3D Tiles to be used e.g., with Cesium-based solutions, enabling them on browser- and game-engine-based platforms [71]. For even more detailed visualization, spatially limited drone-based photogrammetric surveys have been applied for producing highly detailed textured models. However, this approach does not scale well with city-wide applications. Generally, 3D city mesh models are not made with game-engine-use in mind and thus often require a structural overhaul and optimization if intended to be used properly. Optimization would also benefit from having 3D city mesh model datasets in less processed forms to allow for customized workflows. There are also concerns about the quality and reliability of other geospatial source data. For example, the public transit positioning accuracy varies and can cause objects to drift out of their intended areas due to GNSS positioning accuracy in urban canyons.

The utilization of sensor technology can be expected to increase in cities, thus opening new and interesting possibilities for urban-digital-twin applications. Sensor and other geospatial data are also becoming more and more accessible for the general public as open data, which in turn can create new business opportunities. However, this also places

demands on ensuring the usability of APIs (from a developer perspective) and establishing standards for sensor data APIs in the urban digital twin context. For example, the open data provided via HRI is already widely utilized in many applications [72].

5. Conclusions

Urban digital twins have gained increasing interest, with several cities already having initiatives for producing their own. The increasing availability of open 3D city models and geospatial data through various data portals and APIs has pushed the development of urban digital twins forward. Modern game engines could act as potential platforms for such digital twins due to their high-quality visualization and real-time rendering capabilities. Recent developments in game engine technology have also greatly improved their support for geospatial data.

In this study, open geospatial data sets were integrated and visualized with 3D city model data set in a game engine environment. Three open-data integration approaches were demonstrated that utilized three types of geospatial data sources via APIs. Suitable visualization techniques were explored for the data sets with varying temporal resolutions. The integrated datasets included near-real-time public transport data (MQTT API), event and activity data (REST API), and an urban tree register (static data as CSV).

The performance of between the data integration approaches was tested for urban-digital-twin application by measuring their execution times and frame rate. The frame rate was found to be a more limiting factor, as execution times were relatively low for general use. While the static-data approach performed the best, its static nature limits its use for real-time implementations. The MQTT approach on the other hand was found to be promising for implementing real-time applications. MQTT also contained interesting features, such as two-way communication capabilities, which would warrant further research.

The 3D city model data was streamed from Cesium Ion service, which allowed for the use of large-scale city models in game engines. Some limitations were also identified, such as insufficient support for directly uploaded georeferenced models, the singular nature of the uploaded datasets, and the loss of semantic data contained within the CityGML format. Georeferencing-wise, datasets within the game engine appeared to match visually with the aerial imagery. Some georeferencing anomalies were also identified with possible errors originating from inaccurate coordinate data, GNSS coverage issues, and decimal precision within the game engine.

Overall, modern game engines offer a great platform for creating urban-digital-twin applications, however certain problems and limitations must be overcome before the use of geospatial data in game engines becomes feasible on a larger scale. Future developments could include, for example, greater support for CityGML to access building-specific semantic data, game-engine-optimized 3D city models, and improved georeferencing options in Cesium Ion.

Author Contributions: Conceptualization, T.R., A.J. and J.-P.V.; formal analysis, T.R., A.J. and J.-P.V.; investigation, T.R. and A.J.; methodology, T.R., A.J. and J.-P.V.; resources, T.R., A.J., H.H. and M.T.V.; software, T.R.; visualization, T.R.; writing—original draft, T.R., A.J., J.-P.V., H.H. and M.T.V.; writing—review and editing, T.R., A.J., J.-P.V., H.H. and M.T.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The City of Helsinki Innovation Fund project "Helsinki Smart Digital Twin 2025", the Academy of Finland (AoF) through the "Road Distress Mapping Combining Expertise of Sensors and Point Cloud Processing, Surveying and Road Engineering" project (decisions 323783, 323752, and 323750), the Academy of Finland project Profi5 "Autonomous systems" (No. 326246), the European Regional Development Fund (grant number A77489), and the European Union's Digital Europe Programme under grant agreement (No. 101083680).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets utilized in this study that are publicly available can be accessed using the following links: https://kartta.hel.fi/3d/#/ (accessed on 23 December 2022) (3D City Information Model of Helsinki), https://digitransit.fi/en/developers/apis/4-realtime-api/vehicle-positions (accessed on 23 December 2022) (Public transit dataset), https://open-api.myhelsinki.fi/ (accessed on 23 December 2022) (City events and activities dataset), and https://hri.fi/data/fi/dataset/helsingin-kaupungin-puurekisteri (accessed on 23 December 2022) (Urban tree dataset).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Billen, R.; Cutting-Decelle, A.F.; Marina, O.; de Almeida, J.P.; Caglioni, M.; Falquet, G.; Leduc, T.; Métral, C.; Moreau, G.; Perret, J.; et al. *3D City Models and Urban Information: Current Issues and Perspectives*; European COST Action TU0801; EDP Sciences: Les Ulis, France, 2014; pp. 1–118. [CrossRef]
2. Helsinki's Digital Twin and City Models. Available online: https://www.hel.fi/helsinki/en/administration/information/general/3d/3d (accessed on 23 December 2022).
3. Rotterdam 3D. Available online: https://www.3drotterdam.nl/#/ (accessed on 22 December 2022).
4. Berlin 3D—Download Portal. Available online: https://www.businesslocationcenter.de/en/economic-atlas/download-portal (accessed on 22 December 2022).
5. Cousins, S. 3D mapping Helsinki: How Mega Digital Models Can Help City Planners. *Constr. Res. Innov.* **2017**, *8*, 102–106. [CrossRef]
6. Julin, A.; Rantanen, T. Photorealistic 3D Mesh Model Dataset (Helsinki). Zenodo. Available online: https://doi.org/10.5281/zenodo.7599228 (accessed on 30 June 2023).
7. Jovanović, D.; Milovanov, S.; Ruskovski, I.; Govedarica, M.; Sladić, D.; Radulović, A.; Pajić, V. Building Virtual 3D City Model for Smart Cities Applications: A Case Study on Campus Area of the University of Novi Sad. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 476. [CrossRef]
8. Helsingin 3D-Kaupunkimallit (3D City Models of Helsinki). Available online: https://kartta.hel.fi/3d/#/ (accessed on 23 December 2022).
9. Grieves, M. Origins of the Digital Twin Concept. 2016. Available online: https://www.researchgate.net/publication/307509727_Origins_of_the_Digital_Twin_Concept (accessed on 26 July 2023).
10. World Geospatial Industry Council (WGIC). Spatial Digital Twins: Global Status, Opportunities, and the Way Forward. WGIC POLICY REPORT: 2022-01. Available online: https://wgicouncil.org/publication/reports/download-the-wgic-spatial-digital-twins-report/ (accessed on 3 October 2022).
11. Ketzler, B.; Naserentin, V.; Latino, F.; Zangelidis, C.; Thuvander, L.; Logg, A. Digital Twins for Cities: A State of the Art Review. *Built Environ.* **2020**, *46*, 547–573. [CrossRef]
12. Lehtola, V.V.; Koeva, M.; Elberink, S.O.; Raposo, P.; Virtanen, J.-P.; Vahdatikhaki, F.; Borsci, S. Digital twin of a city: Review of technology serving city needs. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *114*, 102915. [CrossRef]
13. Deng, T.; Zhang, K.; Shen, Z.-J. A systematic review of a digital twin city: A new pattern of urban governance toward smart cities. *J. Manag. Sci. Eng.* **2021**, *6*, 125–134. [CrossRef]
14. VU.CITY London. Available online: https://www.vu.city/cities/london (accessed on 22 December 2022).
15. Ofoeda, J.; Boateng, R.; Effah, J. Application Programming Interface (API) Research: A Review of the Past to Inform the Future. *Int. J. Enterp. Inf. Syst. (IJEIS)* **2019**, *15*, 76–95. [CrossRef]
16. Helsinki Region Infoshare. Available online: https://hri.fi/en_gb/ (accessed on 7 February 2023).
17. Open Data BCN. Available online: https://opendata-ajuntament.barcelona.cat/data/en/dataset (accessed on 27 June 2023).
18. Data.NSW. Available online: https://data.nsw.gov.au/ (accessed on 27 June 2023).
19. DATA.GOV. Available online: https://catalog.data.gov/dataset/?metadata_type=geospatial (accessed on 27 June 2023).
20. What is REST. 7 April 2022, by Lokesh Gupta. Available online: https://restfulapi.net/ (accessed on 23 December 2022).
21. MQTT: The Standard for IoT Messaging. Available online: https://mqtt.org/ (accessed on 23 December 2022).
22. Helsingin Kaupungin Puurekisteri (Urban Tree Register of Helsinki). Available online: https://hri.fi/data/fi/dataset/helsingin-kaupungin-puurekisteri (accessed on 23 December 2022).
23. Helsinki Metropolitan Area Service Map APIs. Available online: https://hri.fi/data/en_GB/dataset/paakaupunkiseudun-palvelukartan-rest-rajapinta (accessed on 28 June 2023).
24. 3D Models of Helsinki. Available online: https://hri.fi/data/en_GB/dataset/helsingin-3d-kaupunkimalli (accessed on 22 December 2022).
25. High-Frequency Positioning. Available online: https://digitransit.fi/en/developers/apis/4-realtime-api/vehicle-positions/ (accessed on 23 December 2022).
26. Cesium, J.S. Available online: https://cesium.com/platform/cesiumjs/ (accessed on 19 June 2023).
27. Mapbox. Available online: https://www.mapbox.com/ (accessed on 28 June 2023).
28. ArcGIS Maps SDK for JavaScript. Available online: https://developers.arcgis.com/javascript/latest/ (accessed on 28 June 2023).

29. Google Maps Platform. Available online: https://console.cloud.google.com/google/maps-apis/discover (accessed on 28 June 2023).
30. Würstle, P.; Padsala, R.; Santhanavanich, T.; Coors, V. Viability Testing of Game Engine Usage for Visualization of 3D Geospatial Data with OGC Standards. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **2022**, *X-4/W2*, 281–288. [CrossRef]
31. Virtanen, J.-P.; Jaalama, K.; Puustinen, T.; Julin, A.; Hyyppä, J.; Hyyppä, H. Near Real-Time Semantic View Analysis of 3D City Models in the Web Browser. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 138. [CrossRef]
32. Laksono, D.; Aditya, T. Utilizing A Game Engine for Interactive 3D Topographic Data Visualization. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 361. [CrossRef]
33. Lee, A.; Chang, Y.-S.; Jang, I. Planetary-Scale Geospatial Open Platform Based on the Unity3D Environment. *Sensors* **2020**, *20*, 5967. [CrossRef] [PubMed]
34. Emerging Web and Game Engine Tech for 3D Cities. By Arttu Julin, J.-P.V., Kaisa Jaalama and Enni Airaksinen. 2 October 2020. Available online: https://www.gim-international.com/content/article/emerging-web-and-game-engine-tech-for-3d-cities (accessed on 20 June 2023).
35. Fritsch, D.; Kada, M. Visualisation using game engines. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *35*, 621–625.
36. Greenwood, P.; Sago, J.; Richmond, S.; Chau, V. Using game engine technology to create real-time interactive environments to assist in planning and visual assessment for infrastructure. In Proceedings of the 18th World IMACS/MODSIM Congress, Cairns, Australia, 13–17 July 2009.
37. Buyukdemircioglu, M.; Kocaman, S. Reconstruction and Efficient Visualization of Heterogeneous 3D City Models. *Remote Sens.* **2020**, *12*, 2128. [CrossRef]
38. Clark, T.; Brock, E.; Wu, D.; Liang, Y. Development of Real-Time Smart City Mapping Utilizing Game Engines. In Proceedings of the International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 20–22 October 2020. [CrossRef]
39. ArcGIS Maps SDK for Unreal Engine. Available online: https://developers.arcgis.com/unreal-engine/ (accessed on 22 December 2022).
40. Cesium for Unreal. Available online: https://cesium.com/platform/cesium-for-unreal/ (accessed on 23 December 2022).
41. Cesium for Unity. Available online: https://cesium.com/platform/cesium-for-unity/ (accessed on 20 June 2023).
42. Cesium Ion. Available online: https://cesium.com/platform/cesium-ion/ (accessed on 22 December 2022).
43. Julin, A. Improving the Utilization of Close-Range Photogrammetry and Terrestrial Laser Scanning for Photorealistic Urban 3D Modeling. Ph.D. Dissertation, Aalto University, Espoo, Finland, 2021.
44. Kuts, V.; Modoni, G.; Otto, T.; Sacco, M.; Tähemaa, T.; Bondarenko, Y.; Wang, R. Synchronizing physical factory and its digital twin through an IIoT middleware: A case study. *Proc. Est. Acad. Sci.* **2019**, *68*, 364–370. [CrossRef]
45. Maulana, H.; Kanai, H. Utilizing Game Engine for Development Interactive 3-Dimensional Geographic Information System (GIS) Agriculture Commodity Selection and Land Evaluation. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021. [CrossRef]
46. How China Cloned Shanghai. Published 26 August 2020, by Adam Savage. Available online: https://www.theb1m.com/video/how-china-cloned-shanghai (accessed on 23 December 2022).
47. Wellington Digital Twin. Available online: https://buildmedia.com/work/wellington-digital-twin (accessed on 23 December 2022).
48. 3DAmsterdam. Available online: https://github.com/Amsterdam/3DAmsterdam (accessed on 23 December 2022).
49. Reimagining Paris with the Help of an Urban Digital Twin. 12 November 2021, by Lauretta John. Available online: https://blog.unity.com/industry/reimagining-paris-with-the-help-of-an-urban-digital-twin (accessed on 28 June 2023).
50. Helsinki's Pioneering City Digital Twin. Published 2 June 2022, by George Lawton. Available online: https://venturebeat.com/ai/helsinkis-pioneering-city-digital-twin/ (accessed on 23 December 2022).
51. Lei, B.; Janssen, P.; Stoter, J.; Biljecki, F. Challenges of urban digital twins: A systematic review and a Delphi expert survey. *Autom. Constr.* **2023**, *147*, 104716. [CrossRef]
52. MyHelsinki Open API. Available online: https://open-api.myhelsinki.fi/ (accessed on 23 December 2022).
53. CityGML. Available online: https://www.ogc.org/standards/citygml (accessed on 23 December 2022).
54. How Were the 3D Models Made? Available online: https://www.hel.fi/helsinki/en/administration/information/general/3d/how-were-the-3D-models-made/ (accessed on 23 December 2022).
55. Energy and Climate Atlas. Available online: https://kartta.hel.fi/3d/atlas/#/ (accessed on 23 December 2022).
56. Cesium World Terrain. Available online: https://cesium.com/platform/cesium-ion/content/cesium-world-terrain/ (accessed on 23 December 2022).
57. Cesium OSM Buildings. Available online: https://cesium.com/platform/cesium-ion/content/cesium-osm-buildings/ (accessed on 23 December 2022).
58. Bing Maps Aerial Imagery. Available online: https://cesium.com/platform/cesium-ion/content/bing-maps-imagery/ (accessed on 23 December 2022).
59. 3D Tiles. Available online: https://www.ogc.org/standards/3DTiles (accessed on 23 December 2022).
60. Web Feature Service. Available online: https://www.ogc.org/standards/wfs (accessed on 23 December 2022).
61. Web Map Service. Available online: https://www.ogc.org/standards/wms (accessed on 23 December 2022).
62. Unreal Engine. Available online: https://www.unrealengine.com/en-US/ (accessed on 23 December 2022).
63. BP MQTT. Available online: https://www.unrealengine.com/marketplace/en-US/product/bp-mqtt (accessed on 23 December 2022).

64. VaRest. Available online: https://www.unrealengine.com/marketplace/en-US/product/varest-plugin (accessed on 23 December 2022).
65. UEGeoCoordinates. Available online: https://github.com/ue4plugins/UEGeoCoordinates (accessed on 23 December 2022).
66. JSONParser. Available online: https://www.unrealengine.com/marketplace/en-US/product/jsonparser (accessed on 23 December 2022).
67. Tiling Photogrammetry Models. Available online: https://cesium.com/learn/3d-tiling/ion-tile-photogrammetry/ (accessed on 23 December 2022).
68. Fine Grained Metadata in 3D Tiles Next. 31 May 2022, Dan MacCurdy. Available online: https://cesium.com/blog/2022/05/31/fine-grained-metadata-in-3d-tiles-next/ (accessed on 15 February 2023).
69. Virtanen, J.-P.; Julin, A.; Jaalama, K.; Hyyppä, H. Creating Open Online Applications with Geospatial Interfaces—Case Study "Palvelututka". *Photogramm. J. Finl.* **2021**, *27*, 1–14. [CrossRef]
70. Cesium for Unreal: The First Year. Available online: https://cesium.com/blog/2022/04/05/cesium-for-unreal-the-first-year/ (accessed on 23 December 2022).
71. Google Maps Platform Photorealistic 3D Tiles. Available online: https://cesium.com/learn/photorealistic-3d-tiles-learn/ (accessed on 21 June 2023).
72. Showcases. Available online: https://hri.fi/data/en_GB/showcase (accessed on 7 February 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.