Virtanen, Juho-Pekka; Julin, Arttu; Jaalama, Kaisa; Hyyppä, Hannu

CREATING OPEN ONLINE APPLICATIONS WITH GEOSPATIAL INTERFACES - CASE STUDY “PALVELUTUTKA”

Published in:
Photogrammetric Journal of Finland

DOI:
10.17690/021272.1
10.17690/021272

Published: 19/11/2021

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Virtanen, J.-P., Julin, A., Jaalama, K., & Hyyppä, H. (2021). CREATING OPEN ONLINE APPLICATIONS WITH GEOSPATIAL INTERFACES - CASE STUDY “PALVELUTUTKA”. *Photogrammetric Journal of Finland*, *27*(2). https://doi.org/10.17690/021272.1, https://doi.org/10.17690/021272

The Photogrammetric Journal of Finland, Vol. 27, No. 2, 2021 Received 03.09.2021, Accepted 11.11.2021
Doi:10.17690/021272.1

CREATING OPEN ONLINE APPLICATIONS WITH GEOSPATIAL INTERFACES - CASE STUDY "PALVELUTUTKA"

Juho-Pekka Virtanen[1,2], Arttu Julin[1], Kaisa Jaalama[1,2] and Hannu Hyyppä[1,2]

[1] Department of Built Environment, School of Engineering, Aalto University, P.O. Box 14100, FI-00076 Aalto, Finland
[2] Finnish Geospatial Research Institute FGI, Geodeetinrinne 2, FI-02430 Masala, Finland

juho-pekka.virtanen@aalto.fi, arttu.julin@aalto.fi, kaisa.jaalama@aalto.fi, hannu.hyyppa@aalto.fi

ABSTRACT

Three-dimensional city models are an increasingly common data set maintained by many cities globally. At the same time, the focus of research has shifted from their production to their utilization in application development. We present the implementation of a demonstrator application combining the online visualization of a 3D city information model with the data from an application programming interface. By this, we aim to demonstrate the combined use of city APIs and 3D geospatial assets, promote their use for application development and show the performance of existing, openly available tools for 3D city model application development.

1. INTRODUCTION

Three-dimensional (3D) city models have become established GIS data sets in the urban context, with many cities producing and maintaining them as a part of their survey work. Typically, they depict the urban environment reflecting its geometry, semantics and topology in a digital form, most commonly created following the CityGML standard (Gröger & Plümer, 2012). The term "city information model" is occasionally used for 3d city models with semantic properties. The ongoing development of the CityGML standard introduces further support for multitemporal objects and dynamic attributes in city models, further expanding its applicability (Kutzner et al., 2020).

As 3D city models are increasingly available, the scope of research has during the last ten years shifted from their production towards their regular maintenance, application and use. Here, the research activities have included integrating the models to various registers (Eriksson & Harrie, 2021), using the models for analyses (e.g. Virtanen et al., 2021), distributing them online (Alatalo et al., 2016; Julin et al., 2019) and utilizing them for participatory planning activities (Jaalama et al., 2021). 3D city modeling is also affected by developments in neighbouring fields, such as building information modeling and tighter integration of city and building information models (Ohori et al., 2018).

Applying 3D city models also has a significant role in the "smart city" concept, supporting e.g. participatory planning and scenario visualization (Daniel & Doran, 2013). 3D city models are also connected to the development of urban digital twins (Ketzler et al., 2020). As Ketzler et al. (2020) point out, the precise role of the 3D city model in a digital twin is difficult to define as the term is understood differently by different stakeholders. However, both of these

developments (smart city and urban digital twin) are commonly connecting urban GIS assets, such as 3D city models, with online systems and dynamic data sources.

1.1 Online applications utilizing 3D city models

Several possibilities exist for visualizing 3D city models. In addition to commercial 3D GIS and visualization software, even 3D printing has been applied (Oswald et al., 2019). Interactive visualization, where the user is granted more freedom to study the 3D city models, has been seen as a potential tool for e.g. participatory planning (Jaalama et al., 2021). For this, both dedicated systems and more general tools such as game engines or virtual world clients have been applied. In interactive visualization of 3D city models, the use of systems that operate in the web browser has been seen as beneficial for participatory activities and communication - mostly due to the ease of distributing these applications over the Internet (Julin et al., 2018; Virtanen et al., 2018). Here, the potential tools have included browser-based game engines (e.g. Laksono & Aditya, 2019), specific viewer tools (e.g. Ninja, see Vitalis et al., 2020), commercial participation and visualization platforms and virtual globes (Blaschke et al., 2012).

While game engines offer a high visualization capability, their support for geospatial data is often limited (Buyukdemircioglu & Kocaman, 2020). Browser based virtual globes have therefore been a very attractive tool, as they combine the support for geoinformation data and geodetic coordinate systems with the ability to create interactive applications. Several virtual globe platforms have been developed and applied (Table 1). In addition to these, several others have been developed using e.g. CesiumJS as the starting point, such as TerriaJS (2021). A more thorough listing of web mapping and virtual globe libraries can be found from Gábor (2020).

Table 1. Some available virtual globe platforms

Virtual globe platform	***Project website***	***Use cases and reference(s)***
NASA WorldWind	*https://worldwind.arc.nasa.gov/*	*Cultural heritage & tourism (Brovelli et al., 2013); data visualization (Zong et al., 2012)*
CesiumJS	*https://cesium.com/platform/cesiumjs/*	*Participatory urban planning (Lafrance et al., 2019), flood simulation (Kumar et al., 2018), cultural heritage (Scianna & La Guardia, 2018), managing large construction projects (Zhang et al., 2020).*
iTowns	*http://www.itowns-project.org*	*Geoportal (Konini et al., 2019)*

As 3D city models can become fairly large in terms of data amount, their application in online environments requires efficient transfer and loading. While applied widely for storage and file-based data transfer, the CityGML format is not ideally suited for efficient online visualization (Ohori et al., 2018).

For online visualization of city models, the most common approach is applying model tiling in combination with efficient binary formats that allow storing object information via binding it with model vertices (Schilling et al., 2016). This allows both efficient rendering of the model with limited draw calls and inclusion of semantic information in models - essential for many applications (Schilling et al., 2016). This approach has also become standardized as 3D Tiles (2021). As 3D city models are most commonly stored in a relational database, such as 3D City DB, applying them on an online platform usually implies that this tiled, binary compressed model is generated beforehand for online visualization purposes (e.g. Lafrance et al., 2019). Extending the 3D tiles format to cover multitemporal data has also been suggested (Jaillot et al., 2018).

In many of the presented applications, combining city-wide geospatial data with other data sets in visualization is needed. This additional data can, for example, consist of more detailed models (Scianna & La Guardia, 2018) or planning information (Lafrance et al., 2019). We therefore assume that further integration of online visualization, geospatial assets and other data sources in the urban environment is beneficial for developing new services, increasing the adoption of 3D city models and potentially supporting the progress towards smart cities and urban digital twins.

1.2 Development Aims

Currently, the City of Helsinki provides a number of data sets, and maintains open standard interfaces to geospatial data (Helsinki, 2021c). Two different 3D city models are also provided as open data, both of which cover the entire city area (Helsinki, 2021b). In addition, a number of open interfaces provide data from e.g. city services (Helsinki, 2021a). The 2D map services are applied in conjunction with the APIs to produce various applications, such as the “Palvelukartta” (eng. Service map) online application (https://palvelukartta.hel.fi/fi/). However, the combined use of 3D city models with other data sets or APIs is still more rare.

In the following, we present the technical realization of an application built using the CesiumJS framework, utilizing the open geospatial interfaces offered by the City of Helsinki. Out of the potential virtual globe platforms, CesiumJS was preferred as it is already applied in a number of model viewer applications offered by the city of Helsinki and was thus considered to likely offer a high compatibility with existing data sources.

The developed application “Palvelututka” (eng. “Service Radar”) combines the online visualization of the Helsinki city information model with the data from application programming interface (API) providing information about the services offered in the city. By this, we aim to demonstrate the combined use of city APIs and 3D geospatial assets, promote their use for application development by third parties and showcase the performance of existing, openly available tools for 3D city model application development. The produced application is available from the URL: https://foto.aalto.fi/palvelututka/ and its source code can be obtained from: https://doi.org/10.5281/zenodo.4550516 The application can be used to discover available services in the city based on their location, with a rather limited functionality - its main role being that of a technical demonstrator.

2. MATERIALS & METHODS

2.1 Geospatial assets & interfaces

The city information model - Built according to the CityGML standard, the Helsinki city information model contains building models corresponding to two levels of detail (LoD): 1 & 2. The LoD 1 models consist of the building footprint vector extruded, whereas the LoD 2 contains the simplified roof geometry. Following the specification by Biljecki et al. (2016), they would correspond to Lods 1.2 and 2.1. The models are produced from the building footprint polygons originating from the base map with airborne laser scanning used to derive roof geometries for the LoD 2 building models. The facade textures have been obtained from oblique aerial imaging. The model covers the entire city area.

The model is maintained in a postgreSQL database with the 3DCityDB schema. A public viewing application is available at: https://kartta.hel.fi/3d/#/ The model can also be downloaded as open data from: https://hri.fi/data/en_GB/dataset/helsingin-3d-kaupunkimalli

For use in CesiumJS application, the model was utilized as a 3D Tiles data set directly from the city server (kartta.hel.fi). As the model is in a known coordinate system (ETRS-GK25) its use in CesiumJS is quite straightforward, and only requires adding the tileset to the CesiumJS application. After this, the application controls the loading of model segments according to the camera position, as allowed by the 3D Tiles format. The LoD 2 building models are included in the 3D Tiles data set.

Cesium World Terrain - As the 3D Tiles data set used for the Helsinki city information model only contained the building models, a separate digital terrain model (DTM) was deemed beneficial for visualization - otherwise the building models positioned to correct heights would have “hovered” over the CesiumJS WGS84 ellipsoid in the visualization.

The Cesium world terrain is a global DTM with a regionally varying resolution, formed as a combination of several data sources (https://cesium.com/platform/cesium-ion/content/cesium-world-terrain/). For Europe, its reported approximate resolution is 30m. The DTM is offered as a free asset from the Cesium Ion service, a commercial cloud service for hosting and providing assets for CesiumJS applications.

As the heights of the Cesium world terrain do not match the N2000 heights of the Helsinki city information model buildings, it was necessary to manually adjust the height position of the buildings to approximately match the terrain.

The Guide map of the City of Helsinki was used as the texture for the terrain. It is produced by the City of Helsinki, Kaupunkimittauspalvelut (City surveying services) and available in a variety of color schemes. Intended to support e.g. navigation and guidance it is simplified to a scale of 1:10000 and includes e.g. the road network, buildings, street names and some of the main sights in the city (https://hri.fi/data/en_GB/dataset/helsingin-opaskartta) .

In the Palvelututka application, a grayscale version of the guide map was used, being added to the CesiumJS application directly from the city WMS interface. The application controls the loading of the map as raster tiles according to the camera zoom level and viewing position.

The Service Map API of the City of Helsinki was used as the data source describing the services offered in the urban environment. This API offers data about the services and service points in the Helsinki metropolitan area, along with their details (https://dev.hel.fi/apis/service-map-backend-api). The API is "RESTful", (REST, Representational state transfer) meaning that it is accessible with HTTP queries, the directory structure and query parameters being applicable to narrow down the query and provide specific query terms for the API.

2.2 Software components & functions

The Palvelututka application consists of a set of JS applications and HTML and CSS content, that can be hosted on an ordinary web server (Figure 1). The application relies on two external libraries that are included in the application, CesiumJS and MDL. In the following, the major components of the application are described.

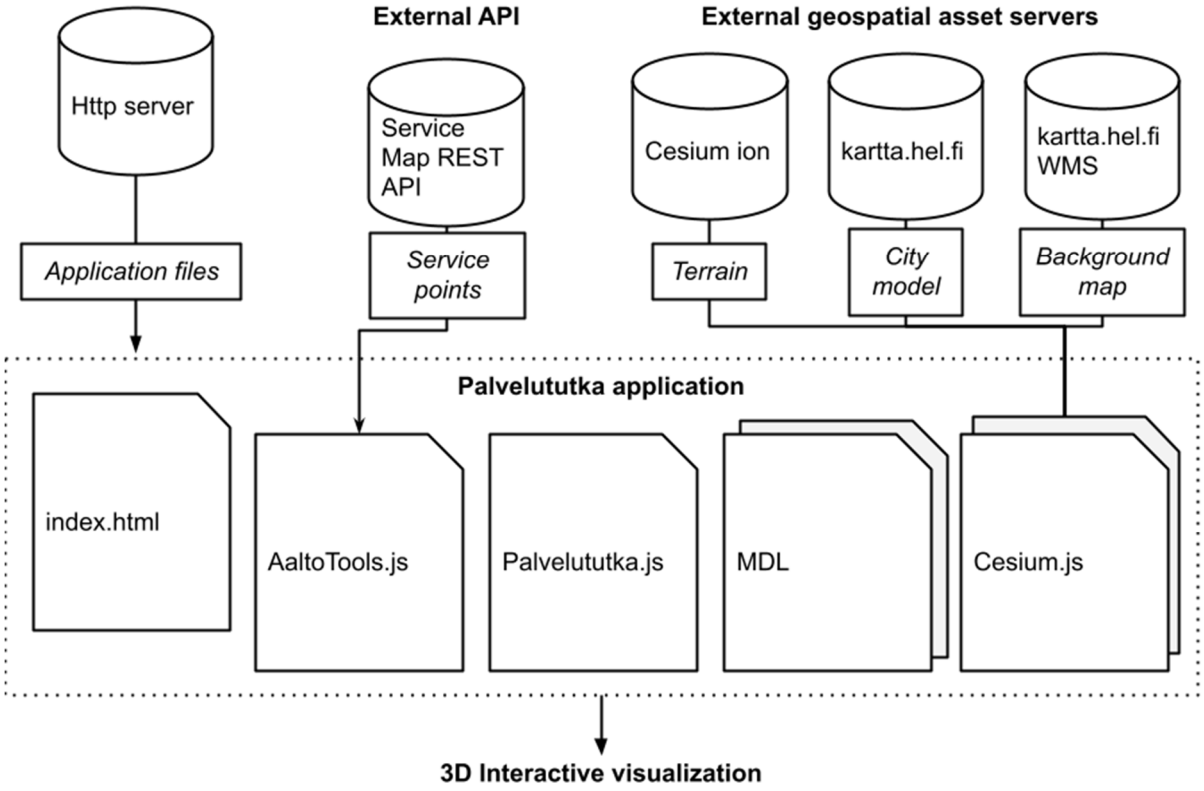

Figure 1. Summary of the application components and external assets used.

CesiumJS library, version 1.73 was used as the core of the Palvelututka application, and for realizing the majority of the functionalities. Realized with JavaScript, the CesiumJS library allows the developer to produce applications that obtain geospatial data from standard interfaces (WMS, 3D Tiles) and render it in a web browser.

To keep the user interface of the demonstrator application as simple as possible, the CesiumJS application is operated with many of its standard UI elements (e.g. timeline, layer selection widget, projection mode selector, home button and geocoding tools) hidden. This was done as it was likely that multiple map levels, projection modes, or geocoding would not be required in the rather minimalistic demonstrator. In addition the CSS definitions of the entity property display elements were modified to alter their appearance by, for example, adding a background image.

In the Palvelututka application, the default tools offered by the CesiumJS library (apart from the UI elements) were used as much as possible. This was to reduce the amount of software development required. In the following, we describe the most important elements and their utilization in the application.

The DTM was added to the CesiumJS scene as a CesiumTerrainProvider element, using the Cesium world terrain directly from the Ion cloud service (https://cesium.com/docs/cesiumjs-ref-doc/CesiumTerrainProvider.html). The city information model buildings were added as Cesium3DTileset, accessing the 3D tiles data directly from the server maintained by the City of Helsinki (https://cesium.com/docs/cesiumjs-ref-doc/Cesium3DTileset.html). For plotting the results of the API query, the EntityCollection element was utilized (https://cesium.com/docs/cesiumjs-ref-doc/EntityCollection.html). It allows the user to create a collection of objects having geographic coordinates and a set of additional information fields, that can then easily be drawn on the terrain. Furthermore, the default CesiumJS UI allows their information to be queried and displayed to the user by simply clicking on the drawn entity in the 3D scene. The Camera element, and its methods were used to realize all camera interactivity in the application (https://cesium.com/docs/cesiumjs-ref-doc/Camera.html). The flyTo-method allows the developer to conveniently request animated camera flights from the application, providing the bounding box of the map area to be displayed. This way, the camera can be efficiently used to zoom into a requested object set by only providing their bounding box.

Material Design Lite (MDL) library was used to create most of the user interface elements in the Palvelututka application. Consisting of a collection of UI elements built using JS and CSS it allows the developer to conveniently specify the needed UI elements in HTML, after which most of their functionality and e.g. responsive rendering according to the screen size is carried out by the library. Out of the components available in the MLD (https://getmdl.io/components/index.html), the menu, cards and buttons were used to create the UI for the Palvelututka application. Compared with the default UI elements of the CesiumJS library, the MDL allows more dialogue styles and menu structures to be built, opening the route for more advanced user interfaces.

Palvelututka.js contains a collection of JavaScript functions for realizing the application. The most significant functions are introduced below. These functions were implemented by authors to realize the application. After the query functionality is requested by the user, the interface is activated to obtain coordinates from the next click on the terrain by setting a binary flag. With this activated, the pickEllipsoid method of the default CesiumJS camera is fired. In practice, this allows the user to click on the displayed terrain, after which the CesiumJS is used to get geographic latitude and longitude for this click event.

The queryServiceMapSpatial function takes the query latitude and longitude (in degrees) as arguments, assembles the http query for the Service map api, and after the query is returned, initiates the parsing of the query results in a separate function. The query result is obtained as a JSON data structure. Prior to plotting, the objects are filtered to remove those that do not have location information as they cannot be easily plotted on map.

The plotServiceMapUnits function parses the query result and creates a CesiumJS entity from the parsed objects. This is then added to the CesiumJS entity collection, containing all of the plotted entities. Prior to appending, the entity collection is checked for duplicate IDs, so individual entities are not plotted twice. Here, the entity collection also provides a list of all

plotted entities without separate maintenance procedures required. After the parsing is complete, a separate function is called to set the UI to the state that allows viewing the results, obtain the bounding box for the current entity collection and order a camera animation to view these.

AaltoTools.js contains a set of self-developed utilities for manipulating 3D Tiles data sets in CesiumJS applications. In the Palvelututka application, the setHeightOffset function is utilized for setting the height offset for the Helsinki city information model buildings to better match the Cesium world terrain.

3. RESULTS

The complete application, when accessed with a web browser, progresses along the following path of functional modes. After loading the MDL, the UI queries the window size, determining the menu styles accordingly. After the CesiumJS is loaded, the city information model and background map are obtained from the respective city servers and added to the scene. The Cesium world terrain is added and the city information model shifted in height to match it better. A simple welcome UI is displayed for the user, drawn as a MLD card (Figure 2). From the welcome UI, the user can initiate the query of the services. This dismisses the welcome card and activates the UI for receiving clicks on the terrain.

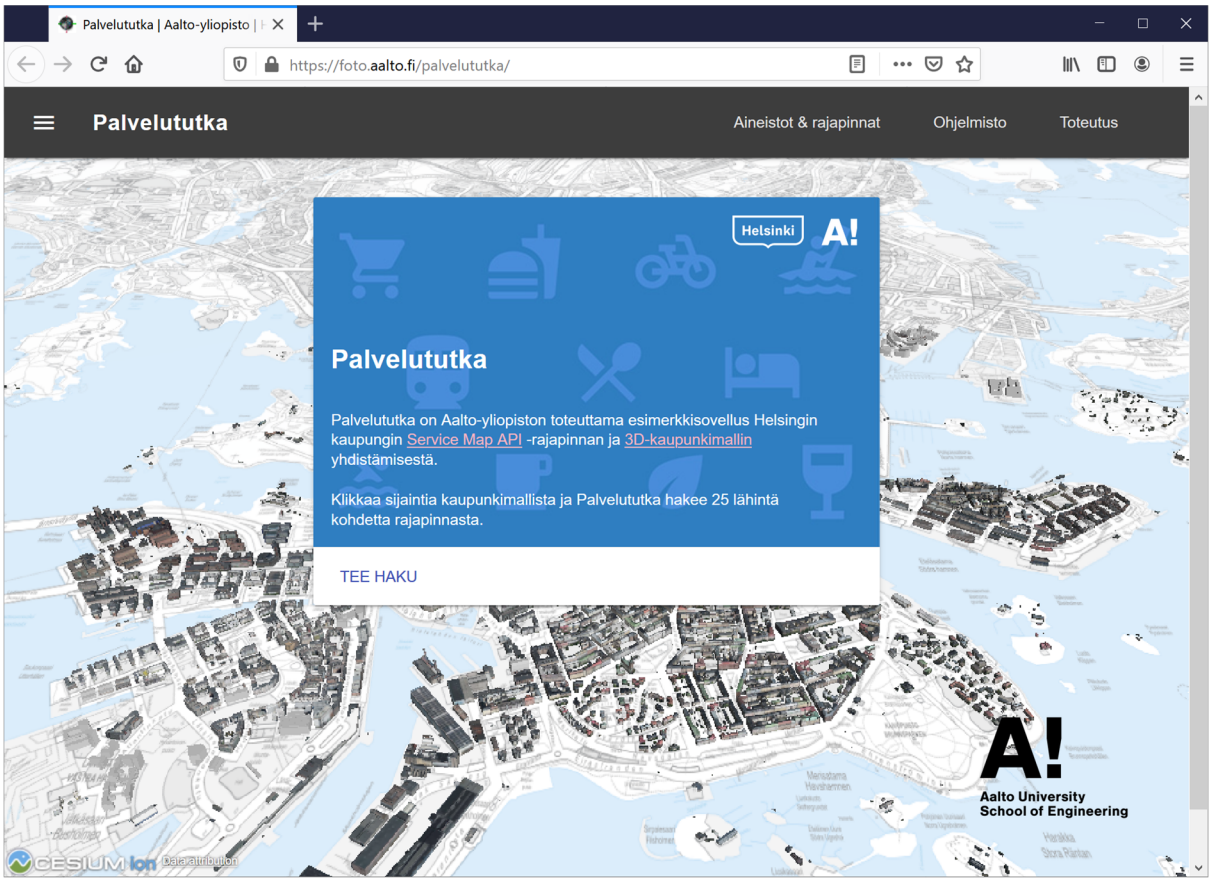

Figure 2. The application in the initial state.

After the user clicks on the terrain, the geographic latitude and longitude of the click are determined and used to perform a spatially limited query from the service map API. The query requests 25 closest services within 1000 meters from the clicked location. After the query is

completed, the camera is flown to display the results and a UI card is shown announcing to the user that query has been completed (Figure 3).

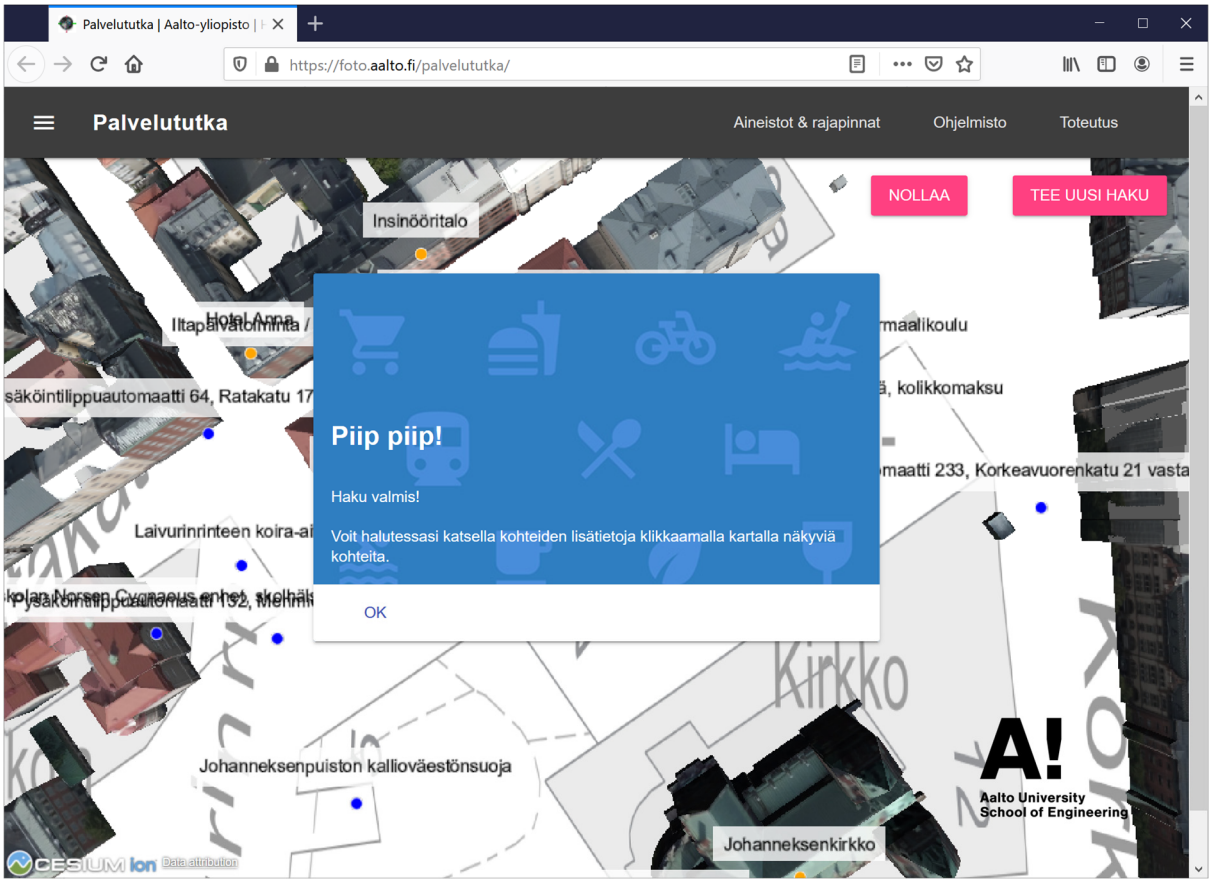

Figure 3. Completed query in the UI.

By dismissing the UI card, the user can progress to navigate in the city model and observe the query results, their points being colored according to whether the services are public or provided by a private company. The labels and their rendering is performed by the default functions of the CesiumJS entity collection entities. By clicking on an individual service on the map, its information is displayed (Figure 4). The information fields displayed originate from the service map API and have been passed with query to the properties of CesiumJS entities. The entity information display is realized with the default tools of CesiumJS, albeit with a slightly modified style.

The user is provided with buttons to either repeat a query, in which case the new resulting entities are added to the current set, or reset the application to its initial state.

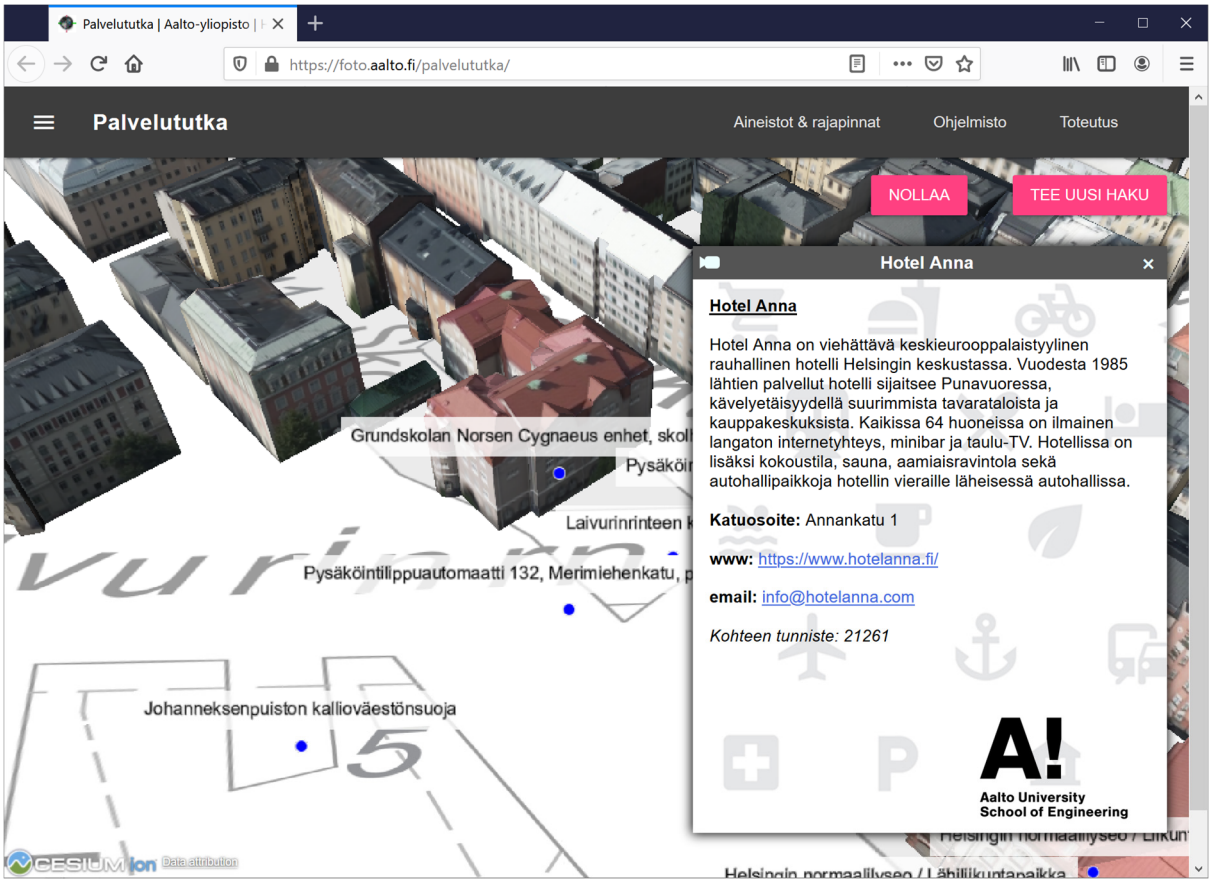

Figure 4. Viewing the details of an individual result item.

4. DISCUSSION & CONCLUSIONS

Several potential tools are available for developing applications that utilize 3D city models, even for browser-based deployments. The combined utilization of standard geospatial interfaces and other APIs is a common practice for producing online applications. This case study, working with the geospatial data assets of the City of Helsinki, set out to demonstrate the use of open APIs with the 3D city model in a browser-based application. As a result, an open demonstrator application was developed using the CesiumJS library, allowing users to retrieve and visualize data from the city's service API on top of the 3D city information model. As a technical demonstrator, it is not feasible as a replacement for the existing 2D applications, as e.g. the query functionalities are far too limited. However, it does successfully combine data from the city service API with the 3D city model in visualization.

Looking at the completed demonstrator application, we can identify several aspects that could be improved in the future. Firstly, concerning the data sets used, there are mismatches between the heights of the city information model buildings and the terrain (Figure 5). This is caused by the limited detail level of the global terrain used and the differences in coordinate systems between the two. As the Helsinki city surveying regularly undertakes airborne laser scanning campaigns and their data is used e.g. for the roof geometries of the LoD 2 building models, it would be the best alternative to produce a DTM from this data set. This would also help minimize the potential temporal differences between the two data sets.

In addition, this would suit the open data approach as currently the Cesium world terrain is a proprietary data set by an individual company. After all, all other data sets in the application

are provided openly by municipal actors. However, if done by an external party, this would require a separate production of a terrain data set and setting up a system to host it for online applications.

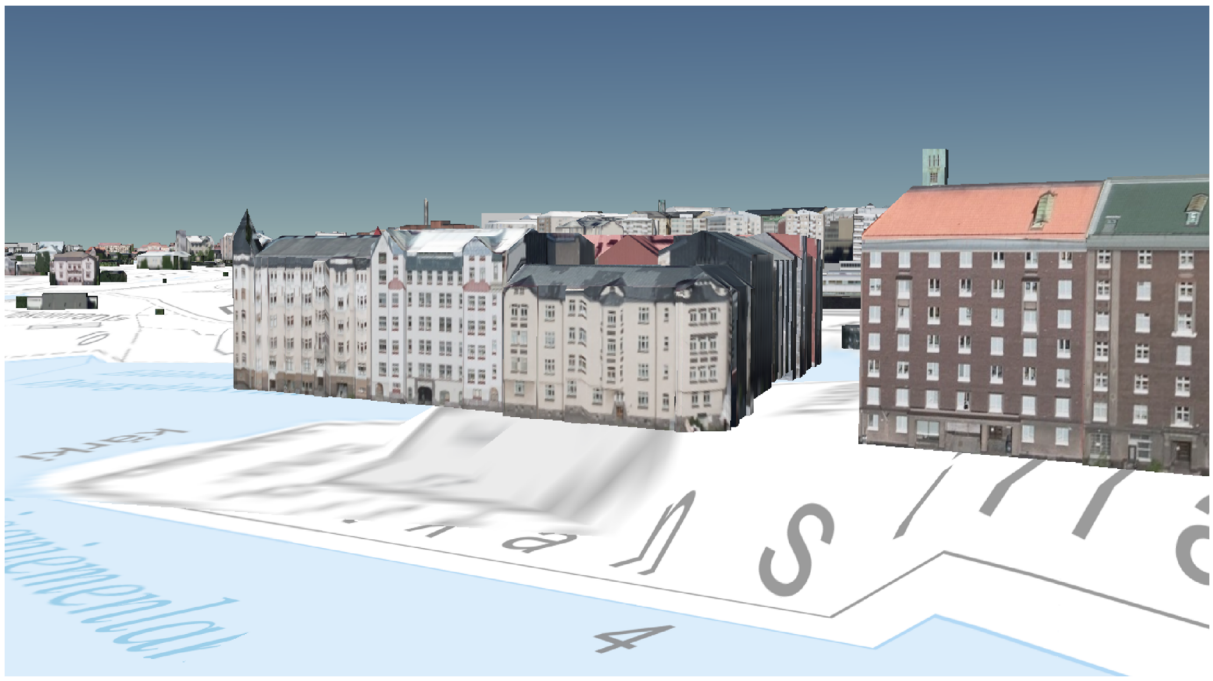

Figure 5. The mismatch between the terrain and the 3D building models is most pronounced near larger terrain forms.

The detail level of the urban visualization is currently sufficient to allow identifying individual buildings and street segments from the urban environment. Nevertheless, including other urban elements in the 3D city model would further support the human interpretation. Inclusion of e.g. trees to CityGML models has been demonstrated in prior research (Virtanen et al., 2021), so this remains a data integration task. An alternative approach would be to apply the textured city mesh model as it offers a higher degree of visual realism than the city information model, and also includes the vegetation elements.

A larger apparent development task would be the tighter integration of service points retrieved from the city API with the building information present in the city information model. Currently, the points are only visualized according to their coordinates as retrieved from the API. By e.g. spatial autocorrelation with building footprint vectors, they could potentially be integrated into the building models where appropriate. Thus, the UI could also better use the building models for visualization, highlighting e.g. the buildings that provide services of the desired nature, such as restaurants. It is questionable whether this data integration could be completed in the browser, or whether it would require separate processing steps beforehand.

A similar task of data improvement would be adding further fields of information to the register behind the service map API. This could include, for example, URLs to images or other documents better describing the service or their classification into sub-categories for more refined queries, for example retrieving all parking services simultaneously. However, this would most likely require changes to the data offered by the used APIs. While the data can of course be processed in the browser prior to visualization, it is most likely unfeasible to perform significant additions or revisions in this stage.

Naturally the application itself could also be refined further. Currently, it merely demonstrates that it is possible to create these kinds of online applications that merge 3D geoinformation with API data. For developing an actually useful tool for e.g. discovering urban services, several aspects of the application would require further design and development. These include at least refining the query UI to allow more complex search criteria and using other search operations than just location with closest points. Here, many of the UI conventions could likely be adapted from the current 2D services. Additionally, only presenting the results spatially is not likely that useful for the user but might be supported e.g. with a listing. In presenting the results as points on the map, their presentation could be improved by applying common conventions from cartography, such as symbols and color coding and revising the camera movement logic when presenting the results. Many of these improvements could likely be realized with the currently applied technologies but their implementation would require significant additional work. As stated, the applied MDL library supports the creation of more complex user interfaces. For more advanced geospatial features, such as managing multiple map layers or temporal data, the default elements of the CesiumJS UI could likely be used. Going beyond the existing data sets and rendering pipeline, alternative styles of 3D presentation emphasizing simplification and readability have also been suggested (Semmo et al., 2015).

On the other hand, even the current version clearly demonstrates the potential and simplicity of building applications that leverage standardized geospatial interfaces in CesiumJS. In this case, the 3D city model, terrain and background map were all brought together without any self-developed processes or middleware. This is highly significant as it greatly simplifies the development and maintenance of the produced applications. Providing novel server-side infrastructure and functionality is potentially time consuming and also results in some operating costs. At the same time, conventional web-hosting is widely available and fairly affordable.

The benefits of directly using the data from city servers via interfaces is also apparent: there are no outdated copies on own servers, nor do these additional content servers have to be maintained. As the municipality officials update the map data, 3D city model or registers that are present via the APIs, all data in the application is automatically up-to-date. In addition to the conventionally used geospatial interfaces, the benefits of “RESTful” APIs are also demonstrated. As the application operates in a browser, using APIs that rely on http requests and data as JSON, their use is extremely simple and can be achieved without additional JS libraries. This also reduces the effort required to maintain these applications. It is also clear that these API queries can be combined with the visualization possibilities of virtual globes rather easily. This is the main technical contribution of the developed demonstrator application.

Combined, the experiences of using the various interfaces are largely positive. They firstly offer some confirmation to the idea that providing open API services could potentially support the creation of new services - even independent of the municipal actors. Secondly, it appears that new services could potentially arise from combinations of existing interfaces and assets and that current technologies can support the creation of these combinations. Finally, they display potential for creating applications that combine data from multiple actors, e.g. different officials, as long as these data are available via similar interfaces.

To conclude, the obtained insight into developing online applications that leverate 3D city models - either as “core data” or simply to provide a 3D scene to support visualization of other data looks increasingly promising. The potential reached by individual developers mostly applying open-source tools and open services is already quite significant and the existing APIs are clearly able to support this development.

5. ACKNOWLEDGEMENTS

The Palvelututka application was realized as a part of “Helsinki Smart Digital Twin 2025” project, funded by the Helsinki Innovation Fund. This report has been prepared with the support of the European Social Fund, project S21997.

6. REFERENCES

3D Tiles. Available online: https://www.ogc.org/standards/3DTiles (accessed on 14 April 2021).

Alatalo, T., Koskela, T., Pouke, M., Alavesa, P., and Ojala, T., 2016. VirtualOulu: collaborative, immersive and extensible 3D city model on the web. In Proceedings of the 21st International Conference on Web3D Technology, pp. 95-103.

Biljecki, F., Ledoux, H. and Stoter, J., 2016. An improved LOD specification for 3D building models. Computers, Environment and Urban Systems, 59, pp. 25-37.

Blaschke, T., Donert, K., Gossette, F., Kienberger, S., Marani, M., Qureshi, S., and Tiede, D., 2012. Virtual globes: serving science and society. Information, 3(3), pp. 372-390.

Brovelli, M. A., Hogan, P., Minghini, M., and Zamboni, G., 2013. The power of Virtual Globes for valorising cultural heritage and enabling sustainable tourism: NASA World Wind applications. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 4, W2, pp. 115-120.

Buyukdemircioglu, M., and Kocaman, S., 2020. Reconstruction and Efficient Visualization of Heterogeneous 3D City Models. Remote Sensing, 12(13), 2128. MDPI AG. 26 pages. http://dx.doi.org/10.3390/rs12132128

Cesium.js (accessed on 14.4.2021). https://cesium.com/platform/cesiumjs/

Daniel, S. and Doran, M. A., 2013. geoSmartCity: geomatics contribution to the smart city. In Proceedings of the 14th Annual International Conference on Digital Government Research, pp. 65-71.

Eriksson, H. and Harrie, L., 2021. Versioning of 3D City Models for Municipality Applications: Needs, Obstacles and Recommendations. ISPRS International Journal of Geo-Information, 10(2), 55, 20 pages.

Gábor, F., 2020. Creating the foundations of a universal client-side Web GIS system. Dissertation, University of Pecs, 24 pages.

Gröger, G. and Plümer, L., 2012. CityGML–Interoperable semantic 3D city models. ISPRS Journal of Photogrammetry and Remote Sensing, 71, pp. 12-33.

Helsinki, 2021a. Available online: https://dev.hel.fi/ (accessed on 5 November 2021).

Helsinki, 2021b. Available online:
https://www.hel.fi/helsinki/en/administration/information/general/3d/3d (accessed on 5 November 2021).

Helsinki, 2021c. Available online: https://www.hel.fi/helsinki/en/maps-and-transport/city-maps-and-gis/geographic-information-data/open-geographic-data (accessed on 5 November 2021).

Jaalama, K., Fagerholm, N., Julin, A., Virtanen, J. P., Maksimainen, M., and Hyyppä, H., 2021. Sense of presence and sense of place in perceiving a 3D geovisualization for communication in urban planning–Differences introduced by prior familiarity with the place. Landscape and Urban Planning, 207, 103996, 14 pages.

Jaillot, V., Servigne, S. and Gesquière, G., 2020. Delivering time-evolving 3D city models for web visualization. International Journal of Geographical Information Science, 34(10), pp. 2030-2052.

Julin, A., Jaalama, K., Virtanen, J. P., Maksimainen, M., Kurkela, M., Hyyppä, J., and Hyyppä, H., 2019. Automated multi-sensor 3D reconstruction for the web. ISPRS International Journal of Geo-Information, 8(5), 221, 21 pages.

Julin, A., Jaalama, K., Virtanen, J. P., Pouke, M., Ylipulli, J., Vaaja, M., Hyyppä, J., and Hyyppä, H., 2018. Characterizing 3D city modeling projects: Towards a harmonized interoperable system. ISPRS International Journal of Geo-Information, 7(2), 55, 18 pages.

Ketzler, B., Naserentin, V., Latino, F., Zangelidis, C., Thuvander, L., and Logg, A., 2020. Digital twins for cities: A state of the art review. Built Environment, 46(4), pp. 547-573.

Konini, M., Devaux, A. and Brédif, M., 2019. iTowns, le nouveau moteur de visualisation 3D de données géospatiales du Géoportail. Annales des Mines - Responsabilité et environnement, 94, pp. 14-18. https://doi.org/10.3917/re1.094.0014

Kumar, K., Ledoux, H. and Stoter, J., 2018. Dynamic 3d Visualization of Floods: Case of the Netherlands. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 6210, pp. 83-87.

Kutzner, T., Chaturvedi, K. and Kolbe, T. H., 2020. CityGML 3.0: New functions open up new applications. PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science, 88(1), pp. 43-61.

Lafrance, F., Daniel, S. and Dragićević, S., 2019. Multidimensional web GIS approach for citizen participation on urban evolution. ISPRS International Journal of Geo-Information, 8(6), 253, 19 pages.

Laksono, D. and Aditya, A., 2019. Utilizing A Game Engine for Interactive 3D Topographic Data Visualization. ISPRS International Journal of Geo-Information, 8(8), 361, 18 pages. MDPI AG. Retrieved from http://dx.doi.org/10.3390/ijgi8080361

Ohori, K. A., Biljecki, F., Kumar, K., Ledoux, H., and Stoter, J., 2018. Modeling cities and landscapes in 3D with CityGML. In Building information modeling, Springer, Cham, Switzerland, pp. 199-215.

Oswald, C., Rinner, C. and Robinson, A., 2019. Applications of 3D printing in physical geography education and urban visualization. Cartographica: The International Journal for Geographic Information and Geovisualization, 54(4), pp. 278-287.

Semmo, A., Trapp, M., Jobst, M., and Döllner, J., 2015. Cartography-oriented design of 3D geospatial information visualization–overview and techniques. The Cartographic Journal, 52(2), pp. 95-106.

Schilling, A., Bolling, J. and Nagel, C., 2016. Using glTF for streaming CityGML 3D city models. In Proceedings of the 21st International Conference on Web3D Technology, pp. 109-116.

Scianna, A. and La Guardia, M., 2018. Globe Based 3D GIS solutions for Virtual Heritage. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42(4/W10), pp. 171-177.

TerriaJS. Available online: https://terria.io/ (accessed on 29 October 2021).

Virtanen, J. P., Hyyppä, H., Kurkela, M., Vaaja, M. T., Puustinen, T., Jaalama, K., Julin, A., Pouke, M., Kukko, A., Turppa, T., Zhu, L., Ojala, T., and Hyyppä, J., 2018. Browser based 3D for the built environment. Nordic Journal of Surveying and Real Estate Research, 13(1), pp. 54-76.

Virtanen, J. P., Jaalama, K., Puustinen, T., Julin, A., Hyyppä, J., and Hyyppä, H., 2021. Near Real-Time Semantic View Analysis of 3D City Models in Web Browser. ISPRS International Journal of Geo-Information, 10(3), 138, 23 pages.

Vitalis, S., Labetski, A., Boersma, F., Dahle, F., Li, X., Arroyo Ohori, K., Ledoux, H., and Stoter, J., 2020. CITYJSON + WEB = NINJA. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 6(4/W1), pp. 167-173. https://doi.org/10.5194/isprs-annals-VI-4-W1-2020-167-2020

WMS. Available online: https://www.ogc.org/standards/wms (accessed on 14 April 2021).

Zhang, S., Hou, D., Wang, C., Pan, F., and Yan, L., 2020. Integrating and managing BIM in 3D web-based GIS for hydraulic and hydropower engineering projects. Automation in Construction, 112, 103114, 13 pages.

Zong, Z., Job, J., Zhang, X., Nijim, M., and Qin, X., 2012. Case study of visualizing global user download patterns using Google Earth and NASA World Wind. Journal of Applied Remote Sensing, 6(1), 061703, 10 pages.