Two Real-World Case Studies on 3D Web Applications for Participatory Urban Planning

Toni Alatalo
University of Oulu & Playsign Ltd.
toni.alatalo@oulu.fi

Matti Pouke
University of Oulu
matti.pouke@oulu.fi

Timo Koskela
University of Oulu
timo.koskela@oulu.fi

Tomi Hurskainen
Playsign Ltd.
tomi@playsign.net

Ciprian Florea
Playsign Ltd.
ciprian@playsign.net

Timo Ojala
University of Oulu
timo.ojala@oulu.fi

ABSTRACT

3D Web is a potential platform for publishing and distributing 3D visualizations that have proven useful in enabling the participation of the general public in urban planning. However, technical requirements imposed by detailed and rich real-world plans and related functionalities are demanding for 3D web technologies. In this paper we explore the maturity of modern 3D web technologies in participatory urban planning through two real-world case studies. Applications built on Unity-based platform are published on the web to allow the general public to create, browse and comment on urban plans. The virtual models of seven urban development sites of different visual styles are optimized in terms of download sizes and memory use to be feasible on browsers used by the general public. We report qualitative feedback from users and present a technical analysis of the applications in terms of download sizes, runtime performance and memory use. We summarize the findings of the case studies into an assessment of the general feasibility of modern 3D web technologies in web-based urban planning.

CCS CONCEPTS

•Applied computing →Cartography; Computer games;

KEYWORDS

Virtual city model, performance, Unity, Emscripten, case study

ACM Reference format:
Toni Alatalo, Matti Pouke, Timo Koskela, Tomi Hurskainen, Ciprian Florea, and Timo Ojala. 2017. Two Real-World Case Studies on 3D Web Applications for Participatory Urban Planning. In *Proceedings of Web3D '17, Brisbane, QLD, Australia, June 05-07, 2017,* 9 pages.
DOI: http://dx.doi.org/10.1145/3055624.3075950

1 INTRODUCTION

Participatory urban planning seeks to engage various stakeholders and community members in urban planning projects. 3D visualizations have proven useful in enabling the participation of the general public in such projects, since they facilitate efficient communication of plans to non-professionals (Bouchlaghem et al. 2005; Wu et al. 2010). The scope of such 3D visualizations can range from individual buildings to entire 3D city models with greatly varying application requirements in terms of spatial accuracy, semantics, aesthetics and historical accuracy (Carrozzino et al. 2009). Moreover, different stages of a planning process can benefit from various abstraction levels in visualizations to keep the participants' attention on relevant aspects (Lovett et al. 2015). Game engines and WebGL have become attractive technology choices for generating visualizations in participatory urban planning projects since they provide aesthetic and interactive visualizations that can be published and distributed efficiently on the web. However, despite recent advancements in 3D Web technology, publishing interactive 3D applications on the web is different from native 3D applications such as games or 3D modeling software, not only in terms of rendering performance and network bandwidth requirements, but also in terms of user expectations on usability and response times.

In this paper we explore the maturity of modern 3D web technologies in participatory urban planning through two real-world case studies. There, applications built on Unity-based platform are published on the web to allow the general public to create, browse and comment on urban plans. The virtual models of seven urban development sites of different visual styles are optimized to have feasible download sizes and memory use for browsers used by the general public. We summarize qualitative feedback from users and present a technical analysis of the applications in terms of download sizes, runtime performance and memory use. The applications focus on delivering the end user experience using state of the art 3D Web technology with available basic compressions and optimizations so that the source data coming from the urban planning systems (BIM, CAD, GIS etc.) is compressed into 3D game engine assets. For this reason, considerations on back-end databases and service system integrations are omitted from technical analysis.

The rest of the paper is organized as follows. Section 2 briefly reviews related work on 3D Web in urban planning. Section 3 introduces the Unity-based platform and overall software architecture and web hosting of the applications and data. Sections 4 and 5 report the two case studies, respectively, covering the functionality of the applications, feedback from users and technical analysis. Section 6 discusses the findings of the case studies with respect to 3D engine choices and selected technical aspects of 3D authoring on the Web. Section 7 concludes the paper.

2 RELATED WORK

In urban planning tools 3D visualizations can be produced with many different technologies ranging from videos to interactive 3D applications (Lovett et al. 2015). 3D data for urban planning applications can exist in multiple forms, for example in form of CAD models, polygon meshes or point clouds. They can be created either by scanning, manual modeling or procedural modeling. Images can be either pre-rendered or real-time rendered. In addition to visualizations, contemporary urban planning software provide many analytical tools that exist mostly outside the game engine realm. These tools can usually produce runtime 3D visualizations and possibly also export visualizations in 3rd party formats (e.g. Sketchup, OBJ, FBX), which can then be imported to custom applications based on game engines such as Unity or Unreal Engine, for example.

Google Earth has been used as a platform for urban planning visualizations. Recently, Web based approaches such as X3DOM, Cesium and three.js have become viable alternatives for many urban planning, 3D city and landscape visualization applications (Krämer and Gutbell 2015; Virtanen et al. 2015).

3D city models on the web are often not only for visual geometry but require additional hierarchy and semantics. In previous work, Chaturvedi et. al. (2015) presented a solution for web-based rich interaction with deeply nested structures of 3D city objects, using additional JSON data format for semantic information. In urban planning, CityGML is currently perhaps the most well-known format for urban 3D visualizations, including not only visual and spatial, but also semantic properties of objects (Kolbe 2009; Kolbe et al. 2005). However, CityGML models are usually not utilized with game engines, web or otherwise.

Although many different technologies for urban planning visualizations have been proposed, any particular universal solution does not seem to exist currently. Instead, technologies as well as design choices should be chosen according to the task in hand (Lovett et al. 2015). Krämer et al. argued that this also holds true for geospatial visualizations in 3D Web (Krämer and Gutbell 2015). Bakri et al. (Bakri and Allison 2016) compared the download sizes and rendering performance of a Unity WebGL build and a native version in visualizing web-based virtual worlds, reporting over three times larger size for WebGL over native version (129 MB for WebGL vs 40 MB for native). In this article we analyze seven scenes, obtaining nearly equal sizes for WebGL and native builds (16-71 MB for WebGL vs 13-60 MB for native).

3 THE TOOLKIT

3.1 Scope and Features of The Toolkit

The toolkit aims to provide a set of user-friendly (non-professional) applications spanning the whole urban planning life cycle. It consists of three applications, *Create, Experience* and *Live*, which cover the building life cycle from early idea gathering to detailed planning and all the way to deployment.

In the early exploration phase, built-in tools of *Create* can be used to sketch ideas, e.g. to draw roads in an area plan. In the design phase, models made by architects and planners with professional tools are imported to the plan. Throughout the design phase, *Experience* allows different stakeholders and the general public to explore and even virtually test the plan in action. Finally, when a building is taken to use, the *Live* service continues to use the models to provide maintenance service UIs and IoT visualization and controls. However, *Live* was not used in the two case studies and is thus excluded from this paper.

The applications are relatively simple and focus on providing a basic set of features while publishing plans created with professional tools. The applications are made with Unity using both 3rd party software components from the Unity asset store as well as in-house components made specifically for the toolkit (Figure 1. The two applications have distinct technical requirements in that *Create* uses user generated content and procedural modeling while *Experience* uses large detailed scenes.

In the case studies, the source materials include area models created with e.g. ArchiCAD, relatively complex polygon models created with e.g. Rhino, simple building masses (boxes) created for area models with SketchUp, similar SketchUp models of surrounding already existing areas exported from city database using Trimble Locus SKP export, and relatively detailed building models created with 3D Studio Max and exported to FBX. All these models are imported to Unity for inclusion in the applications and web publishing, thus they are normalized and optimized with state of the art mesh compression techniques. The focus of the study is on the use of light 3D models for interactive 3D applications on the Web where the models can be explored, configured and even modified on the fly. The applications also utilize pre-rendered 2D images, but technical aspects of still image rendering are omitted from the analysis presented in this paper.

In this paper, data from the first two deployments of Create and Experience is used to evaluate the usability of 3D Web in (collaborative) urban planning. The toolkit is used for the analysis with the reasoning that by being relatively complete it provides good technological coverage. The application domain is certainly demanding, as very high quality visualizations typically obtained with pre/offline rendering are often the norm in architecture. In city models, while building models can be lightweight, their amount can be very large. Nonetheless, useful game-mechanic like interactions in applications require realtime 3D. Finally, the applications should achieve sufficiently high FPS on computers used by the general public while keeping download sizes feasible.

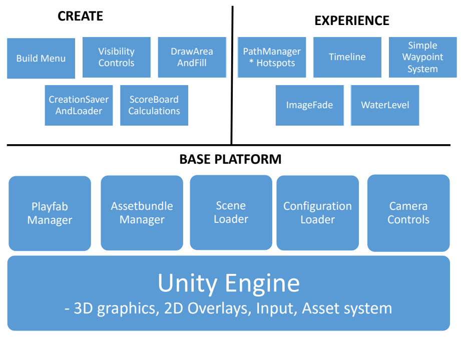
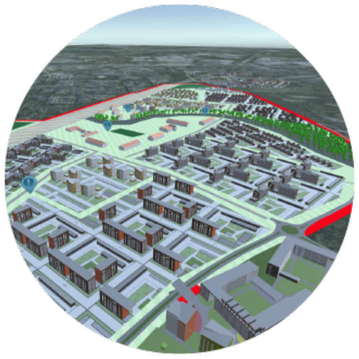


Figure 1: Applications are composed from re-usable components atop the Unity engine

Table 1: The toolkit covers the whole urban planning and building life cycle with three applications

CREATE	EXPERIENCE	LIVE
		
Plan areas in 3D and publish online Build roads, parks and residential blocks Use several map types as the base for all plans Citizens can plan their own versions and comment on others.	Showcase plans online Citizens can visit and experience future areas Hotspots: What kind of services would the citizens prefer here? Open feedback & analytics tools Simulated traffic and ambience to bring the plans to life	Monitor existing buildings Introduction to premises Build a virtual community Share on-time information on problems, local temperatures, free working spaces etc. Dynamic use of BIM

3.2 Web Usability and Design Considerations

WebGL is used to run the 3D applications in a web browser. Unity3D utilizes the Emscripten compiler for web builds, by compiling both the C++ engine and C# written application code via C++ to Javascript (or WebAssembly starting later in 2017). The applications are also published as native builds for mobile devices. Unity was chosen because of good support for importing various 3D assets such as the built-in SketchUp SKP import, ease of authoring with the Unity editor as well as good quality mobile application builds.

Unity WebGL builds face three main challenges when publishing nontrivial 3D scenes for the general public: 1. Large download sizes can result in too long initial waiting times; 2. Detailed graphics may lead to unacceptably slow rendering speed (FPS) on low-end hardware; and 3. The application may not exceed the fixed size memory allocation allowed for applications compiled from Emscripten C++ to Javascript, especially on 32-bit browsers (Trivellato 2016).

For a seamless user experience on the Web, 3D applications should minimize waiting times and not be too heavy to run. A simple strategy using a small initial download followed by larger downloads for additional data is described for a 3D city model in (Alatalo et al. 2016). *Experience* employs a similar strategy by first displaying a light menu for scene selection and loading larger 3D assets on-demand. This is implemented by using Unity's asset bundle system to distribute 3D contents separately from the application. *Create*, however, is simply downloaded and initialized at once. The rationale is that we expect people's tolerance for download times to vary according to context and expectations. *Experience* is for large audiences to view and explore urban plans, perhaps only for a short while, and thus should not be too different from typical fluent web browsing. *Create*, instead, is intended for working on plans for extended periods of time. Therefore, we assume that people accept longer load times, as it is still easier than downloading and installing a separate desktop application.

3.3 Entity-Component Systems for 3D Applications

As described earlier, the toolkit aims to serve different stakeholders through all stages of a planning process. The components of the toolkit should be independent enough so that they can be mixed in various combinations. The development strategy was to utilize re-usable configurable software components which can be selected to be used in a particular planning project. A common technique in complex and 3D game software architectures is to create such composable pieces of functionality as entity-component systems. Besides games such as MMORPGs, they have been demonstrated to be useful for extensible general purpose virtual worlds (Alatalo 2011). Here we work with the assumption that entity-component systems are similarly useful for a configurable toolkit for urban planning. For example, a commenting system requires particular kind of data for a 3D object or scene commented upon, as well as provides a parcitular type of UI. Thus, a general purpose commenting functionality can be implemented as a comment component, attachable to any entity in the scene, with the accompanying software module. Unity uses components both for the built-in core functionality, own additional subsystems such as physics, and for application developer created extensions (scripting). Thus it was deemed appropriate for satisfying the need for extensibility and configurability of these applications.

4 CASE 1: CREATE IN SKETCHING URBAN AREA PLANS

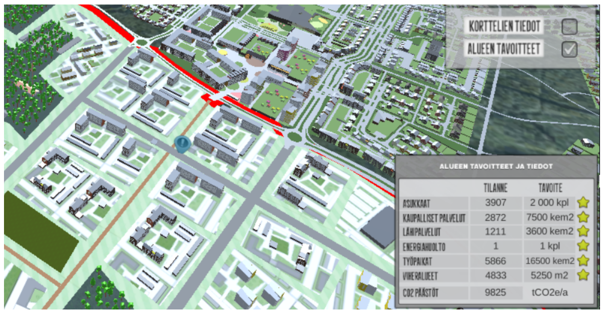
In the first case study, *Create* was used for sketching the area plan of a new suburb to be developed by the City of Oulu. Sketching was conducted in two workshops and at home by voluntary participants.[1]

4.1 Application Functionality

Create supports quick sketching of urban area plans for the purpose of gathering of early ideas. It resembles city management games such as SimCity and Cities:Skylines but is much simpler. Instead of detailed 3D modeling, it features quick and rough urban plan zoning combined with a game-like 3D visualization of the plan. The application has been developed in collaboration with the City Planning division of the City of Oulu in 2016.

A *Create* instance consists of a static 3D model of the existing surrounding areas, various alternative base maps used as ground textures and an empty area for the new plan and suitable building blocks for procedural 3D visualization of the plan. All source materials were provided by the City Planning division in common formats: SketchUp SKP exports from Trimble Locus database for relevant surrounding areas in the city, OBJ files for a new nearby area that has been planned in 3D in more detail, and FBX files for the city block models used in the sketched plan visualization.

Create enables drawing of 2D polygons that can contain different types of buildings or green areas, which is typical for zoning in urban planning. It also allows drawing of streets and walkways as simple polylines. Additionally, *Create* features automatic calculation of several attributes for the current plan including the number of inhabitants, amount of office space and green areas, and estimated CO2 emissions (see Figure 2). For example, when the plan reaches a target of 2000 new inhabitants, a game-like star icon for that target is lit. These calculations and visualizations are done by *Create* without any external services or data.



ALUEEN TAVOITTEET JA TIEDOT

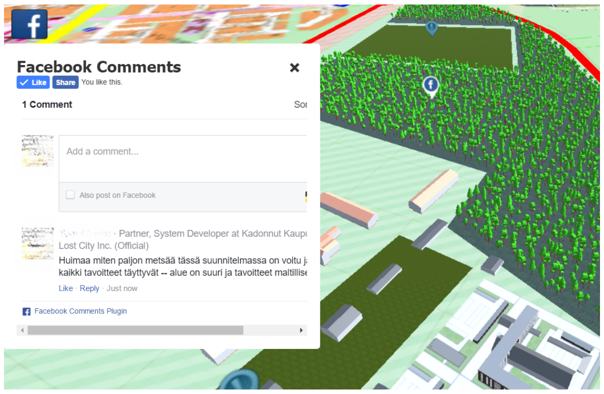
	TILANNE	TAVOITE	
ASUKKAAT	3907	2 000 kpl	★
KAUPALLISET PALVELUT	2872	7500 kem2	★
LÄHIPALVELUT	1211	3600 kem2	★
ENERGIAHUOLTO	1	1 kpl	★
TYÖPAIKAT	5866	16500 kem2	★
VIHERALUEET	4833	5250 m2	★
CO2 PÄÄSTÖT	9825	tCO2e/a	

Figure 2: A star is lit when a design target is met

Each user is provided with a personal account for storing sketched plans that can also be published on the web for anyone to see. Finally, *Create* is integrated with Facebook for authentication, commenting and liking features. Comments and likes can be added both to a whole plan and to individual freely geopositioned pins (Figure 3).

4.2 Feedback from Workshops and Home Use

Before publishing on the web, *Create* was first tested in two workshops. First, the City Planning division organized a workshop for


Facebook Comments ×

Like Share You like this.

1 Comment Sor

Add a comment...

Also post on Facebook

[redacted] · Partner, System Developer at Kadonnut Kaupu
Lost City Inc. (Official)
Huimaa miten paljon metsää tässä suunnitelmassa on voitu ja
kaikki tavoitteet täyttyvät -- alue on suuri ja tavoitteet maltillise
Like · Reply · Just now

Facebook Comments Plugin

Figure 3: Geopositioned URLs for Facebook commenting

professionals working in different divisions within the city organization, e.g. infrastructure, economics, culture etc. Second, a public workshop was arranged for property owners, developers and people living in the area. In both workshops, participants worked in groups, 5 groups in the first workshop and 3 groups in the second workshop. They were given the choice of sketching an area plan using either traditional pen and paper method with a map or using *Create* - all groups in both workshops elected to use *Create*. *Create* was also used for presenting the results at the end of the workshops, which proved to be quick and practical approach.

After the workshops, we published *Create* on the web and solicited home use by voluntary participants who immediately reported practical problems. First, the download size of 80 MB was considered large. In response, the download size of *Create* was optimized down to 36 MB. Second, rendering performance was found to be borderline usable on machines with weaker GPUs – this was mainly due to the inability to use precalculated baked lighting with the 3D scene that the users are creating on the fly. Solving this problem is still an open issue. Third, individuals who were not urban planning professionals reported being at loss about what they were supposed to do. In the workshops, planning professionals facilitated discussion, presented alternatives, asked for possible solutions and refined ideas further. After the first home use test period, the score board with the goals for the plan was added (Figure 2). We have not yet studied how helpful it has been for home users thus it provides an interesting question for future research.

4.3 Technical Analysis

The total download size of 36 MB for *Create* is acceptable taking into account the average download speed in Finland (21 Mbps). This total download size includes base maps, 3D graphics of the surrounding areas, and the block models used by *Create* to fill the areas (i.e. 2D polygons) that people draw.

The data created by users includes a set of objects with the shape of a simple polygon, polyline or a single point together with the type constant. Text descriptions are stored together with the comment pin's 3D position data. The whole document is serialized to a single JSON string and stored on a cloud backend service over HTTP. Commercial cloud service provider, Playfab.com, further

[1]The *Create* instance is publicly available at http://hiukkavaara3d.ouka.fi

uses Amazon’s DynamoDB for storage. After modifications, the document is saved as a whole for simplicity. The minimum storage space for a single key/value pair in DynamoDB is 1 KB. Hence, it is better to combine more data into a single value and not have for example an individual key/value pair for each comment. Following this rationale, Playfab’s HTTP API has a fixed limit of max 10 key/value pairs per request when storing data. *Create* meets this requirement easily by having a fixed set of data fields per document: name, userID, areas (including roads), description and comments. The size limit of a single field is 100 kB so the system works as long both the area polygon data and the textual comment data stay within that limit. Due to the small amount of data, it is stored in human readable unoptimized JSON for convenience. As an example, a relatively complex area plan is in total 15 KB, with 12 KB of area data and 3 KB of comments. HTTP ZIP compression is used to compress data before transmission.

The main finding of this analysis is that simple application specific data, typical to participatory and co-design tools, can be lightweight even though it is used in a rich 3D environment. Thus, common web service backends and technologies such as JSON and NoSQL key/value storages are suitable for this kind of 3D Web applications. It can also be emphasized that the simple GeoJSON standard seems to cover many uses in urban planning with a simple set of geometric primitives: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon, together with Feature and Feature collection for extended types.[2] For interoperability, *Create* could also use GeoJSON instead of its own serialization which currently uses less inefficient 3rd party Unity Vector3 serialization.

Fixed size heap memory allocation for Unity Emscripten builds can be difficult for user created content. Originally, the size of heap memory was set to 384 MB and 512 MB, which were sufficient for *Create* and simple urban area plans created in the workshops. However, in home use people created more complex plans requiring heap memory up to 768 MB, close to the limit of in 32-bit web browsers. Later professionally made official plan alternatives were visualized using Create, with even more detailed and thus heavier plans. However a major optimization in the plan instantiation reduced the memory requirement so that even most complex current plans can now be loaded with a reasonable 512 MB allocation.

For publishing a plan, Playfab’s group feature is used so that a reference to the plan data is added to the group data which anyone participating in the planning project can access. The group model originally developed for games by Playfab suits *Create* in this context very well and the use of 3D graphics does not pose any problems for this feature.

Playfab’s Unity client side SDK supports Facebook login. To implement Facebook Like and Comments, a GUID is generated for each plan and comment pin. The base URL of the application instance is used to form the full URL by adding the GUID. Facebook’s Open Graph supports the use of Like and Comments features on any URL.[3] We use the browser side UI tools, instead of Unity’s WebGL context, for the application’s Facebook GUI. When a 3D element with associated Facebook functionality is activated in the Unity application, it calls browser side Javascript to open a HTML element where the Facebook Like and Comment plugins are embedded and populated with the associated data.

[2]http://geojson.org/

[3]https://developers.facebook.com/docs/sharing/opengraph

5 CASE 2: EXPERIENCE TO PUBLISH THE WINNERS OF THE NORDIC BUILT CITIES CHALLENGE

In the second case study, *Experience* was used to publish the six winners of the Nordic Built Cities Challenge [4]. First we describe the objectives and features of the application, and then briefly discuss site specific design aspects in all six sites. For user feedback we have informal anecdotal reports on positive and negative experiences. Technical analysis of web and mobile applications reports the download sizes of the 3D scenes and runtime memory use and rendering performance. We also describe our solution for enabling deep linking into the scenes.

5.1 Application Functionality

Experience is a cloud based application for publishing urban plans on Web and mobile platforms. It enables game-like exploration of plans with a simple touch-compatible UI targeted for non-professionals. *Experience’s* features include the configuration of hotspots for highlighting selected parts of a plan. A hotspot can be, for example, a location, where users can compare, comment and vote on alternative designs. Also, pre-rendered architectural visualizations can be seamlessly integrated as 2D images into the 3D scene, with additional information and interactions. Currently, *Experience* does not include user created content or any special web service integrations. The comment and like functionalities employed in *Create* can be used similarly. Our analysis focuses on how a variety of basic 3D modeling styles succeed in providing feasible end user performance in terms of download sizes, memory use and rendering.

5.2 Usage

Experience was used to publish the six winners of the Nordic Built Cities (NBC) Challenge architecture competition. The competition was organized by Nordic Innovation and the six regional winners for each individual challenge were selected in June 2016. In fall 2016, software developers worked together with the six architect teams on publishing the winning plans on 3D web and mobile. A generic requirement for *Experience* was to enable publishing basically any area plan. Parallel with software development, separate manuscripts, configurations and 3D scenes were prepared for each site.

We have not yet conducted in-depth user studies for *Experience*. However, we do have convincing anecdotal evidence from various stakeholders involved in the challenge, especially from the organizer and the architect teams, demonstrating that the application and the 3D scenes of all six sites are relevant, useful and actually work in practice in terms of presenting the plans interactively on the Web. By February 2017, the organizer had only heard positive feedback about the application. Several architects involved in the challenge have explicitly stated how the rich interactive exploration of the 3D scene is a new and useful way for them to present their plans. They have also been sufficiently satisfied with usability

[4]Publicly available at http://nbc.playsign.net. Video at https://vimeo.com/199795753

and aesthetics to actually use the 3D Web application. Negative feedback has centered on some users having experienced too long download times and not being satisfied with the visual quality of the 3D scenes. However, based on the feedback received so far, we believe that the application and its 3D scenes provide enough relevant data to analyze how current 3D Web technologies succeed in this domain.

The designs of each six sites had been originally created by different teams of architects and other designers using various tools. Besides 3D models, the designs contained additional data in form of 2D illustrations, diagrams and text to describe various aspects of the designs. The key role of *Experience* is to offer fluent exploration of a design through a coarse overview 3D model and with the possibility to navigate to specific areas containing more detailed information. In this analysis we focus on the 3D models only, providing brief overview of the styles and techniques, and analyzing resulting runtime 3D assets.

5.2.1 Sketching of building masses in an area plan. A common use case in the early phase of urban planning is to sketch building masses for an area model. There, buildings do not have any details such as colors or windows that would distract people from considering how the area works (Lovett et al. 2015). Two sites fall clearly in this category, Kera in Finland (Figure 4) and Karsnes in Iceland (Figure 5). For Kera, the architects demanded a stripped black and white style with outlines of the buildings. Modeling tools, such as SketchUp, can draw such outlines nicely. With Unity, and in our understanding with similar tools in general, it was problematic to produce desired effect programmatically in real-time. An edge detection shader was used, but the results were either unstable so that lines flickered as the camera moved, or with more exact calculations too heavy for common devices, especially with WebGL. Thus, the effect was drawn on the objects beforehand, either by adding outline textures or with additional bevel geometry with black face coloring.

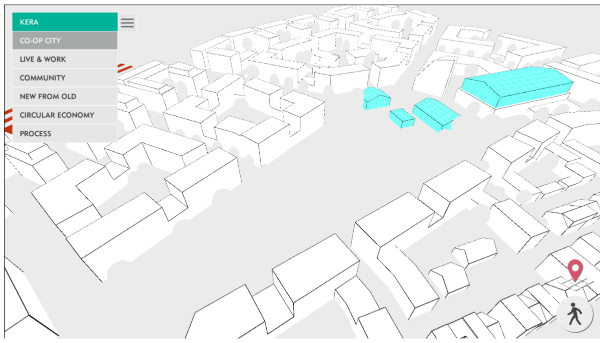


Figure 4: Kera has a very abstract style with outlined boxes

For Karsnes, the SketchUp made box model scene was used as is, just by baking global illumination textures. This kind of lighting makes the shapes nicely visible without outlines.

5.2.2 Illustrate a plan with identifiable semantics for planned services but not building designs. Two sites, Trygve Lies plaza in Oslo, Norway (Figure 6) and Norrebro in Copenhagen, Denmark

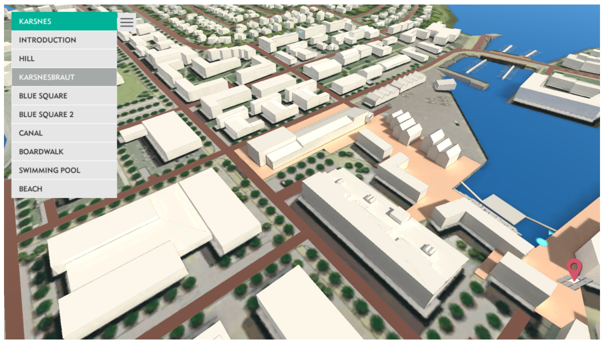


Figure 5: Karsnes simple building mass with baked lighting

(Figure 7), involve closer exploration of design features also at ground level, while surrounding buildings are still white rough shapes. The designs were modeled in 3D, but with low level of detail and utilizing an abstract texturing style. In Trygve Lies, a key part of the design is a small bridge with an entrance to an underground bike garage; the model provided by the architects was detailed enough to explore the route at street level.

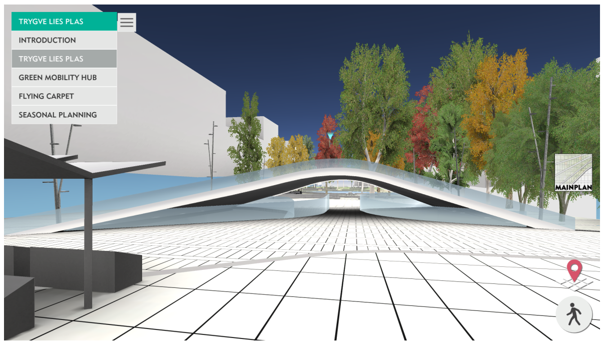


Figure 6: Trygve Lies has models for key features and trees

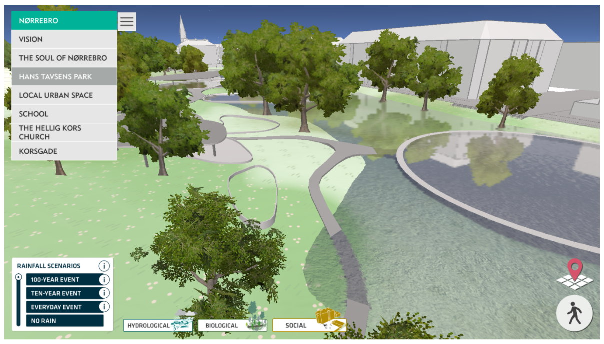


Figure 7: Norrebro excess rainfall examination

The Norrebro park is designed so that it can withstand excess water during heavy rainfalls. For this reason, the 3D scene includes a terrain with an accurate heightmap so that water levels can be correctly visualized. Both scenes include trees made using the SpeedTree component integrated in Unity.

5.2.3 Photorealistic and detailed textured style. The remaining two sites, Runavik in Faroe Islands (Figure 8) and Sege Park in Malmö, Sweden feature photorealistic and detailed textured styles. Photorealism is the default style in architecture visualizations for presenting detailed designs of individual buildings. This is typically especially in marketing, where much emphasis is put on stunning visualizations. Photorealism is challenging for real-time rendering, but we achieved it by pre-rendering baked light textures. The Unreal engine is a popular choice for such architecture visualizations, also targeting VR devices. Unity works similarly when utilizing carefully crafted light bakes. However, such high quality visualizations contain too much data for Web use when covering larger areas. For interior visualizations, however, they already work. Runavik and Sege Park are compromises with medium detail models and limited amount of textures. Runavik features a semi-realistically textured terrain using texture splatting to cover a large area with little texture data. In Sege Park, the scene is a single city block which is composed of modular parts for apartments and other necessary spaces, which can be explored individually in more detail.

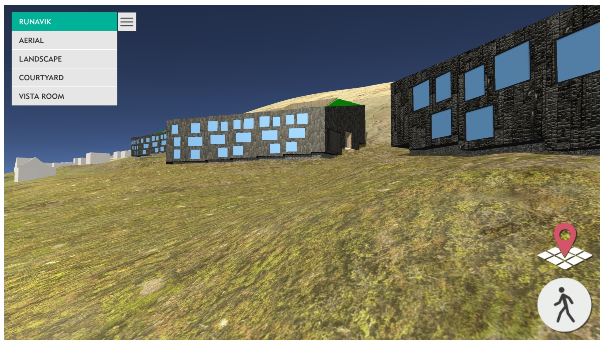


Figure 8: Runavik with a large detailed terrain

5.3 Technical Analysis

Our analysis focuses on download sizes, rendering performance and memory use, relative to the different visual styles used in the 3D scenes of the six sites.

The 3D scene of a site is distributed individually on the web as a Unity asset bundle. A bundle consists of a small text based manifest file with metadata, including the name of the Unity asset in the bundle and the hash of the corresponding binary file. The binary includes all 3D and scene data in Unity's binary format, either uncompressed or with additional LZMA or LZ4 block compression. The bundles are platform specific, so they need to be separately created for e.g. WebGL, Android, iOS and Windows. In this case, the bundles are automatically built using Unity Cloud Build services and uploaded to the CDN used. This automation guarantees that different bundles are comparable.

All the meshes in the scenes are compressed using Unity's built-in compression. Textures are compressed with Crunch with the default quality setting of 50. Regarding additional compression of the unity binary file, the default LZMA compression is the most effective regarding file size. Unfortunately, LZMA is too slow for decompression by Javascript in the browser (Vinson 2017), and it would also require additional memory due to the browser sandbox not having access to a file system where it could store the uncompressed data before loading it to the engine. Therefore, Unity has added LZ4 block compression with which decompressing is both fast and easy on memory. LZ4 is the default compression for WebGL bundles since Unity 5.5, whereas LZMA remains recommended for native builds in typical scenarios. On the web, the built-in GZIP decompression in browsers is used in addition to LZ4 by zipping the bundles and serving them with the "Content-Enconding: gzip" HTTP header. Recently, more effective Brotli compression has been added to some browsers. Unfortunately, the automated publishing of the NBC bundles relies on Google Cloud Storage tools which handle the GZIP compression but do not support Brotli yet, so we could not assess its performance.

The NBC scenes use substantial amounts of 2D imagery in combination with the 3D scenes. However, the 2D data is excluded from this analysis since it is not wholly optimized yet, and because we do not consider it to be relevant for evaluating 3D Web usability.

5.4 Result Data

The data of the six NBC sites is shown in Table 2. For each site we report the style, use of texturing and realtime lighting in the 3D scene. Data download sizes and FPS figures are reported both for WebGL and native versions. Memory use refers to the usage of Emscripten heap.

Not surprisingly, the abstract building masses in Karsnes and Kera are the lightest, even though the areas are the largest. Web downloads of 16-19.5 MB are still substantial but usable. These abstract scenes would be much smaller without light bake textures, but they are necessary for decent visual quality on typical poor GPUs. Norrebro at 23.2 MB is not too large either, and contains only a few 3D objects with more detail. However, it includes a fairly large terrain necessary for water level visualizations. Runavik includes an even larger terrain with somewhat high quality; a main feature on the site is visualizing how the proposed new buildings are shaped to the landscape which is explored by a virtual walk of a few hundred meters on the ground level. However, Runavik does not include much additional complex 3D data, so the overall size is balanced at 33.6 MB. Conversely, Trygve Lies does not have a terrain but includes more detailed mesh models of the landmark bridge as well as other objects on the square and bicycles in an underground bike garage, resulting in similar size of 31.8 MB even though the area is small. Finally, Sege Park is the only site where the 3D data is fairly extensive yielding 71.8 MB web download. There, the geometry includes a whole city block and nine interior models of premises with details such as water faucets in kitchens. The usability of Sege Park could be improved by splitting it into several bundles, so that the overall block model would be loaded and shown first, then interiors would be loaded in the background. However, the total size of Sege Park is already similar to the others because unlike the other sites it does not use any 2D images.

5.5 Runtime Performance

All six scenes are relatively light. Powerful hardware, for example a gaming laptop, can run them fine also with WebGL. Table 2 shows FPS measurements for a weaker GPU device, a 2014 Surface Pro

Table 2: Nordic Built Cities scene analysis

Site	Style	Texturing	Realtime light	Data / Web (MB)	Data / Native (MB)	Ems. Heap (MB)	FPS / Weak GPU	FPS / native build
Karsnes	Abstract building mass	None	No	16.0	13	323	15-60	60
Kera	Abstract building mass	Outlines	No	19.5	15	333	11-54	35-60
Norrebro	Stylized semiabstract, key features modeled	Terrain, trees, not buildings	No	23.2	19.8	423	1-14	46-60
Runavik	Semi realistic mid/low detail	Terrain, new buildings	Yes	33.6	29	452	3-15	6-24
Sege Park	Detailed city block with interiors	Whole block, also interiors	No	71.8	60	608	2	10-12
Trygve Lies	Partly detailed scene, trees	Yes, abstract style	No	31.8	46	350	4-54	9-60

3 with Intel i7-4650U @ 2.30 GHz, 8 GB, Intel HD Graphics 5000 and Windows 10 Pro 64-bit, where Intel HD 5000 is a common graphics card in regular (non-gaming) laptops. The application and all bundles used Unity 5.5.1f1 which uses WebGL 2.0 for rendering when available. The test runs were made with Chrome 56 which uses WebGL 2.0. Same device was used for FPS measurements with a Windows native build of the application for comparison.

The Surface Pro is able to show the very simple abstract 3D scenes well but becomes sluggish with the more complex scenes, and can show the most complex Sege Park scene only at 2 FPS. Although the scene could be optimized further, the data suggests that for audiences with ordinary devices WebGL content must be very light. Culling with Unity and baked occlusion data seems to work well so also the heavier scenes should be usable in most cases. Native Windows desktop build runs the scenes 2-5 times faster so relative performance decrease with WebGL is still substantial.

Regarding memory use, the more abstract city scenes use decent 323-452 MB of Emscripten dynamic heap, whereas the much more complex Sege Park scene uses 608 MB. The heap allocation was configured for 640 MB. The scenes were static with no user created or other dynamic content so static allocation is not problematic.

5.6 Client Side URL Parsing for Deep Linking

All NBC scenes are accessed using the generic client application, *Experience*, which is accessed at a particular URL providing a menu for selecting a scene to be explored. The application has the ability to exit the current scene and select another, which is quick because the Unity engine is already loaded. Alternatively, each scene could be hosted as a separate web page, which would require reloading the Unity engine and initialization of the Emscripten build which is currently a heavy operation. To enable direct deep linking to the scenes, Experience supports including the name of the target scene in the URL as a query parameter. When using the application, it manipulates the URL so that the address of the current scene can be shared and reloaded from the browser normally.

6 DISCUSSION

6.1 Javascript Libraries vs. WebAssembly Compiled C++ Engines

A major choice for 3D Web application authors today is whether to use a typically smaller plain Javascript library or a full C++ game engine via Emscripten compilation. Upsides of vanilla Javascript include typically much faster startup and ease of integration with other web client side code. Unity and Unreal using C++ provide, however, much more powerful graphics capabilities, especially the ease of authoring of 3D applications. It should also be noted that Emscripten engine builds are larger to download and heavier to initialize, but that can still be small in terms of file size when compared to 3D scene data. It is yet unclear how much WebAssembly will help in this respect. Ability to deploy the same 3D application as a native build, especially for mobile devices, was a large contributing factor for the choice of technology in the two case studies presented in this paper – combined with the ease of authoring and familiarity with Unity. Thanks to Emscripten optimizations and occulusion management in Unity, it was also assumed that the rendering performance of Unity Emscripten compile is better than that of for example three.js.

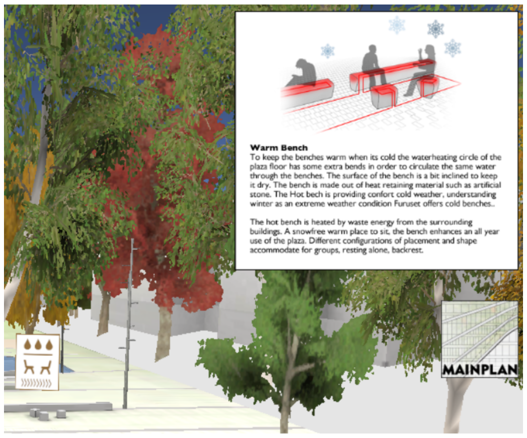
6.2 Styles and Heaviness

It is feasible to publish various kinds of urban plans on WebGL, but they need to be very lightweight in order to guarantee decent performance also on low-end devices. One optimization strategy is to choose carefully, where the more detailed 3D graphics are actually needed, e.g. in terrain or buildings. If the rendering performance of WebGL continues to catch up with the performance of native rendering, this problem will become less significant in the future.

6.3 Web Addressability and Linkability

The Web is a hypermedia system, comprising of documents and media elements with unique URIs, which is woven together by linking. When one encounters an interesting article or an image on the web, the link to it – or even to a specific part of it – can be shared e.g. in a social media post.

By default, the hypermedia capabilities taken for granted on the Web otherwise are lost in typical WebGL applications today. For instance, in a three.js or a Unity application there is no way to refer to a part of the 3D content with a URI. If one encounters an interesting building in a city model it is not possible to refer to it specifically nor share it, other than taking a screenshot, which loses both the 3D model and the information about the origin. Even worse, 2D content such as images and text in 3D Web applications are only available embedded in the application and not as individual elements on the web. This is also the case in the presented applications, illustrated in Figure 9 with the image and text describing a part of an urban design. The architectural design presentation includes hypermedia style icons integrated in the 3D scene, but corresponding 2D content tis not available on the Web – only as a part of the 3D application's data bundle. The data bundles are used to efficiently download and process a large number of small assets.


Warm Bench
To keep the benches warm when its cold the waterheating circle of the plaza floor has some extra bends in order to circulate the same water through the benches. The surface of the bench is a bit inclined to keep it dry. The bench is made out of heat retaining material such as artificial stone. The Hot bech is providing confort cold weather, understanding winter as an extreme weather condition Furuset offers cold benches..

The hot bench is heated by waste energy from the surrounding buildings. A snowfree warm place to sit, the bench enhances an all year use of the plaza. Different configurations of placement and shape accommodate for groups, resting alone, backrest.

Figure 9: Deep linking to the 2D or 3D objects is not possible

For the case studies presented in this paper, two solutions have been implemented to improve addressability. First, the applications support including the ID of the scene in the application URL so that a particular 3D scene can be linked to directly. Also, it is possible to add at arbitrary locations geopositioned comment pins that have unique URLs and can hence be referred to. The comment pin URLs allow the use of the Facebook Like and Comment plugins, because any valid URL on the Web can participate in Facebook's Open Graph. Combined with the approach by Chaturvedi et. al. (2015), this kind of 3D applications can be further integrated with the Web infrastructure. Also the existing 3D Web solutions with DOM integration may well be be relevant for this. For example X3D supports both link anchors in the 3D scene and URI fragment identifiers to refer to a viewpoint in the scene.[5]

7 CONCLUSION

Our case studies show that it is possible to publish 3D city models for participatory urban planning on the Web using Unity so that they are usable with common user devices. WebGL does still however suffer from a very significant performance slowdown of 2-5 times compared to running the same content in a Windows native build. In order to work on weak devices with WebGL the scenes have to have abstract lightweight style and be optimized. Download and initialization of the engine and the 3D contents is slower and heavier than what people are used to on the Web. This can be mitigated by communicating expectations to the users, by informing that they need to wait a while to download an application. As an upside, after download usage is seamless. However, exploiting streaming and progressive loading, common in web applications and with 3D Web, would be interesting solutions for Unity based 3D city models as well. Overall, the 3D Web, including common backend solutions, already works in these applications. Still, extra care and work is needed in creating hyperlinking and 3D application contents so that we can integrate them with the rest of the web.

[5]http://www.web3d.org/x3d-resources/content/examples/Vrml2.0Sourcebook/Chapter28-Anchor/_pages/page01.html

8 ACKNOWLEDGEMENTS

This work has been supported by the 6Aika: Open City Model as Open Innovation Platform (A71143) and the Open Innovation Platform (A70202) projects funded the ERDF and the City of Oulu under the Six City Strategy program, and the COMBAT project (293389) funded by the Strategic Research Council at the Academy of Finland. Playsign application development has been carried out in the Collaborative City Co-design PlatfOrm (C^3PO) ITEA2 project from call 8 (reference: 13016) with financing by the Technology Development center of Finland (TEKES).

REFERENCES

Toni Alatalo. 2011. An entity-component model for extensible virtual worlds. *Internet Computing, IEEE* 15, 5 (2011), 30–37.

Toni Alatalo, Timo Koskela, Matti Pouke, Paula Alavesa, and Timo Ojala. 2016. VirtualOulu: Collaborative, Immersive and Extensible 3D City Model on the Web. In *Proceedings of the 21st International Conference on Web3D Technology (Web3D '16)*. ACM, New York, NY, USA, 95–103. DOI:http://dx.doi.org/10.1145/2945292.2945305

Hussein Bakri and Colin Allison. 2016. Measuring QoS in Web-based Virtual Worlds: An Evaluation of Unity 3D Web Builds. In *Proceedings of the 8th International Workshop on Massively Multiuser Virtual Environments (MMVE '16)*. ACM, New York, NY, USA, Article 1, 6 pages. DOI:http://dx.doi.org/10.1145/2910659.2910660

Dino Bouchlaghem, Huiping Shang, Jennifer Whyte, and Abdulkadir Ganah. 2005. Visualisation in architecture, engineering and construction (AEC). *Automation in Construction* 14, 3 (2005), 287 – 295. DOI:http://dx.doi.org/10.1016/j.autcon.2004.08.012 International Conference for Construction Information Technology 2004.

M. Carrozzino, C. Evangelista, and M. Bergamasco. 2009. The Immersive Time-machine: A virtual exploration of the history of Livorno. In *Proceedings of 3D-Arch Conference*. 198–201.

Kanishk Chaturvedi, Zhihang Yao, and Thomas H Kolbe. 2015. Web-based Exploration of and Interaction with Large and Deeply Structured Semantic 3D City Models using HTML5 and WebGL. In *Wissenschaftlich-Technische Jahrestagung der DGPF und Workshop on Laser Scanning Applications*, Vol. 3.

Thomas H Kolbe. 2009. Representing and exchanging 3D city models with CityGML. In *3D geo-information sciences*. Springer, 15–31.

Thomas H Kolbe, Gerhard Gröger, and Lutz Plümer. 2005. CityGML: Interoperable access to 3D city models. In *Geo-information for disaster management*. Springer, 883–899.

Michel Krämer and Ralf Gutbell. 2015. A Case Study on 3D Geospatial Applications in the Web Using State-of-the-art WebGL Frameworks. In *Proceedings of the 20th International Conference on 3D Web Technology (Web3D '15)*. ACM, New York, NY, USA, 189–197. DOI:http://dx.doi.org/10.1145/2775292.2775303

Andrew Lovett, Katy Appleton, Barty Warren-Kretzschmar, and Christina Von Haaren. 2015. Using 3D visualization methods in landscape planning: An evaluation of options and practical issues. *Landscape and Urban Planning* 142 (2015), 85 – 94. DOI:http://dx.doi.org/10.1016/j.landurbplan.2015.02.021 Special Issue: Critical Approaches to Landscape Visualization.

Marco Trivellato. 2016. Unity WebGL Memory: The Unity Heap. https://blogs.unity3d.com/2016/12/05/unity-webgl-memory-the-unity-heap/. (2016).

Ben Vinson. 2017. Unity WebGL Memory and Performance Optimization. http://developers.kongregate.com/blog/unity-webgl-memory-and-performance-optimization. (2017).

Juho-Pekka Virtanen, Hannu Hyyppä, Ali Kämäräinen, Tommi Hollström, Mikko Vastaranta, and Juha Hyyppä. 2015. Intelligent open data 3D maps in a collaborative virtual world. *ISPRS International Journal of Geo-Information* 4, 2 (2015), 837–857.

Huayi Wu, Zhengwei He, and Jianya Gong. 2010. A virtual globe-based 3D visualization and interactive framework for public participation in urban planning processes. *Computers, Environment and Urban Systems* 34, 4 (2010), 291 – 298. DOI:http://dx.doi.org/10.1016/j.compenvurbsys.2009.12.001 Geospatial Cyberinfrastructure.