Comparison of Two Workflows for Web-based 3D Smart Home Visualizations

Matti Pouke
Center for Ubiquitous Computing
University of Oulu
University of Oulu, P.O Box 4500
FI-90014 University of Oulu, Finland
Email: matti.pouke@oulu.fi

Juho-Pekka Virtanen
School of Engineering, Aalto University
P.O. Box 14100, FI-00076 Aalto, Finland
Email: juho-pekka.virtanen@aalto.fi

Mahmoud Badri, Timo Ojala
Center for Ubiquitous Computing
University of Oulu
University of Oulu, P.O Box 4500
FI-90014 University of Oulu, Finland
Email: mahmoud.badri@qt.io,
timo.ojala@oulu.fi

***Abstract*—In the built environment, the emergence of Internet of Things and the Smart Building paradigm increase the amount of networked systems that produce data from their environment. 3D user interfaces can help users cope with these systems. A 3D representation of a building can operate as a starting point for creating these interfaces. We experimented with creating a 3DUI for sensor network data visualization in two cases, testing both a manually created game engine model and a BIM model as a basis. Solutions were compared in terms of performance. While BIM model captures both the 3D geometry of a building along with its structural properties, some limitations were encountered in using it for online 3D application development.**

I. Introduction

Internet of things (IoT) refers to the networked interconnection of objects, which are often equipped with ubiquitous intelligence. IoT aims to integrating every object for interaction via embedded systems, leading to a highly-distributed network of devices communicating with human beings as well as other devices [1].

In the built environment context, the emergence of networked systems and sensors has also been acknowledged [2]. Often the adoption of IoT in construction is associated with the term ”Smart Building”. Even though attempts have been made to more accurately define the term, and distinguish it from other terms (such as ”Intelligent Building”) [3], the term remains somewhat ambiguous, with some authors using the term ”Smart Home” in a largely similar context [4].

Commonly, smart buildings are considered to apply automated and networked systems for attaining improved energy efficiency [4] and control of building's operations including heating, ventilation, air conditioning, lighting, security and other systems. A smart building uses sensors and actuators in order to collect data and manage it accordingly to provide its intended function. The controls that work together to bring the smart building to life are called building automation system (BAS). BAS core functionality keeps building temperature within a specified range, provides light to rooms based on an occupancy patterns, monitors various home systems performance and failures, and sends automatic alarms in case of systems malfunctions or other faults. Depending on the fault, alarms could be sent to the homeowner, building maintenance staff, or other parties. Smart building infrastructure leads to improved occupant comfort, efficient operation of building systems, saving in energy use, optimizes how space is used, and minimizes the environmental impact of building.

While several aspects of smart buildings remain topical for further research [5], these networked systems are being installed in increasing numbers. The multitude of data produced by such smart buildings presents a challenge of information visualization. User interfaces that allow the user to observe and control the systems of a building, preferably from a single access point, are needed. 3D user interfaces (3DUIs) have been seen beneficial in landscape and built environment visualization [6], [7], and the maturity of current WebGL technology already allows the development of web-based commercial applications in this domain [8]. In the context of individual buildings, 3DUIs have been utilized, for example for visualizing patient activities for caretakers [9], [10]. In comparison to traditional UIs however, 3DUIs provide additional challenges for designers [11].

The advent of browser-based 3D graphics and the inherently 3-dimensional spatial nature of building data makes browser-based 3DUIs an attractive option for smart home visualization applications as well. However, while real-time IoT data can be obtained relatively effortlessly, the modeling of actual building geometry requires manual work. This makes the utilization of Building Information Models (BIM) tempting, provided they have already been developed for planning and construction phase of the building that is the target of the visualization. In addition, the semantic properties of BIM models have obvious potential for being beneficial in smart home applications.

Several authors have suggested the use of Building information models (BIM) for visualizing data related to buildings [12], [13]. In other experiments, game engine technology [14], or other 3D application development platforms [15] have been applied. In addition, building models have been integrated with 3D city models [16]. A particular research angle has also been the expansion of BIM applications to cover the entire lifecycle of the building, and not just planning and construction [17]. Advantages of using 3D user interfaces utilizing a 3D model of the building have been reported by e.g. [17], [12]. The integration BIM and IoT is aided by the emergence of online storage and retrieval systems for BIM [18].

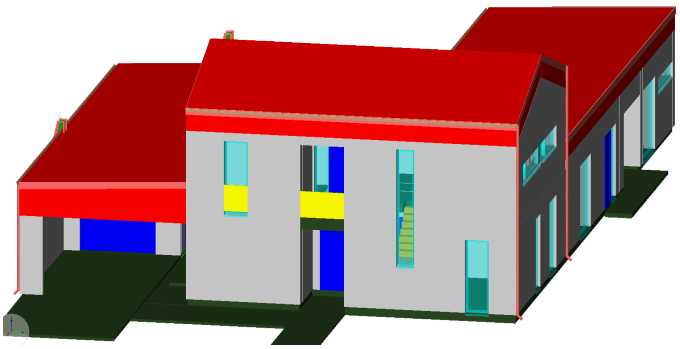

Fig. 1. A BIM model of a detached house shown in FZK Viewer

For accomplishing interactive 3D visualization of building models and associated data, different strategies have been applied: Motamedi et. al [12] utilize existing commercial software for 3D visualization of BIM. Hagedorn & Döllner [16] suggest expanding interfaces commonly used in geospatial information to operate with BIM models. Models can also be produced with indoor 3D mapping [15]. In Virtanen et al. [15], all data sets are converted to game engine compatible format prior to visualization, effectively producing a separate visualization model. Lee et al. [19] present a browser based system with automated model processing and storage in a database prior to visualization.

Especially for larger building complexes or multiple buildings, BIM datasets can become computationally heavy. In addition, the models are not directly suitable for use in 3D game engines due to their inherent differences: most game engines use mesh models, whereas BIM is a parametric model. If a BIM model is converted to a mesh, manual work is often required for filtering out elements not needed for visualization, defining materials, defining surface normal orientations, creating texture coordinates and optimizing the model polygon count. In addition, the semantic properties of BIM models are lost, or at least severely restricted, during the conversion to mesh model. These issues restrict direct application of BIM in browser based 3D systems, and to some extent, in game engines in general.

As a partial solution, BIM models can be processed automatically by dedicated tools intended for the online application (e.g. BIMServer [20]). Performance can be further improved by utilizing models more optimized for browser based 3D engines. However, this creates an additional process step, potentially hindering automation.

In this study, we will present two cases of utilizing 3DUIs in web based smart home IoT visualization. Case A utilizes a server-hosted BIM model, ThingSpeak [21] sensor architecture and three.js [22] WebGL library for visualization. Case B uses a game-engine ready mesh-model and a SOAP webserver using the Unity game engine for visualization. We will compare the two cases in terms of performance and development challenges.

The rest of the paper is organized as follows: Section 2 will explain the details of the two cases that are used for comparison. Section 3 will introduce the results we were able to obtain measuring and analysing both systems. Section 4 will conclude the paper also presenting future research directions.

II. Case details

The two cases serve as systems for visualizing sensor data from a sensor network deployed in a smart building, utilizing a 3DUI. In Case A, the target of the visualization is a two-bedroom apartment hosting a heterogeneous sensor network (the sensors are from multiple manufacturers and not originally providing a unified interface). In Case B, the target is a multistory hospice utilizing a homogeneous commercial sensor network. In Case A, the apartment used was owned by a research facility, allowing it to be utilized for experiments as necessary. Case B was a commercially developed hospice building by a private contractor. While various visualization metaphors and viewport manipulation paradigms have been experimented in both cases (a study is currently in preparation), inherently they both provide the same basic capabilities for orbiting, zooming and exploring the visualization options. The requirements for sensor architecture and visualization features were agreed in co-operation with project partners consisting of researchers as well as four local companies engaged in building services engineering (HVAC, electric installations, etc.) and sensor technology. Several focus group interviews were conducted to gather specifications for 3DUI as well as to identify general interests of the stakeholders. In this study, we are focusing on the following particular questions that were brought up within the focus group interview:

- *"What is the easiest way to produce a building model for 3D visualization, and what is the simplest possible functioning model?"*
- *"Where and in which form should building sensor data be hosted and streamed"*.

In addition to the questions presented above, loose requirements and development ideas were presented for the 3DUIs. Some of them were as follows:

- Browser based: The 3DUI should be accessible anywhere with common online devices.
- Glanceability: The user should not have to perform complex queries to read sensor data. Instead, the 3DUI should quickly inform the user whether or not everything is normal.
- Sensor readings should be easy to interpret according to their meaning. I.e. the 3DUI should automatically provide sensor readings with interpretations, such as "good/bad" or "hot/cold" with corresponding easy-to-understand visualizations.
- For tenants, the system should be used primarily for visualization and less for actually controlling the devices.
- The 3DUIs should be aesthetically pleasing enough to be used as marketing or demonstration material.

Drawing from the results of the focus group interviews, we decided to attempt a development of an IoT platform consisting of sensors and actuators. Pilot studies utilizing the platform should be conducted within multiple real-world buildings. The heterogeneity of the sensor and actuator network should be invisible for the user; IoT devices from multiple manufacturers should be visualized and controlled through an unified interface. Building on top of the built-in interface provided by each device, the IoT device backend would provide input and output capabilities to each individual device through a single channel.

The operation of the backend would be through a 3DUI. The principles of Information Visualization should be utilized in displaying the sensor readings. This would mean, for example, following the Schneiderman's mantra "Overview first, zoom and filter, details-on-demand" [23]. Colours and other metaphors would be used to provide meaning to sensor readings besides displaying their raw data values.

A. Case A

In Case A, it was decided that the 3DUI should utilize a BIM model examining its possibilities in a web visualization application. The apartment in question had no existing BIM-model available, so it was modeled using the Autodesk Revit modeling software. The modeling was done using a floor plan as source material and according to the 3DUI specifications, omitting roof and other details that were obstructing or unnecessary for the visualization. However, furniture and materials were utilized to produce an aesthetically pleasing enough model. The BIM model was then uploaded to the Open Source BIM server [20], in which the model could be hosted, updated and downloaded to the client as necessary. Sensors from multiple manufacturers were installed in the apartment; temperature, humidity, air quality and pressure difference can be measured from multiple locations within the apartment.

A big question at this point was how the IoT data was to be hosted and streamed in relation to the BIM model. It was concluded that while the BIM model can be used to fetch IoT device locations, it is not convenient for hosting the actual sensor data. Thus, it was decided that a separate database is developed for the sensor data while sensor locations are added to the BIM model as necessary; after experimenting with various alternatives, the ThingSpeak architecture was chosen for Case A.

For visualizing sensor readings, it was decided that different alternatives and styles should be implemented for experimentation. These alternatives utilize overlaid 2D icons (text, columns, icons, colors, arrows) as well as 3D geometry (digital and physical meters, fire, steam). These alternatives can be toggled utilizing an overlaid 2D menu. As the focus group interview data stated that "good" and "bad" reading values should be provided automatically by the system, this was taken into account within some of the visualizations. For example, the temperature in the sauna room of the apartment can climb much higher in comparison to other rooms without the 3DUI interpreting it as "bad", see Fig. 3. The 3DUI

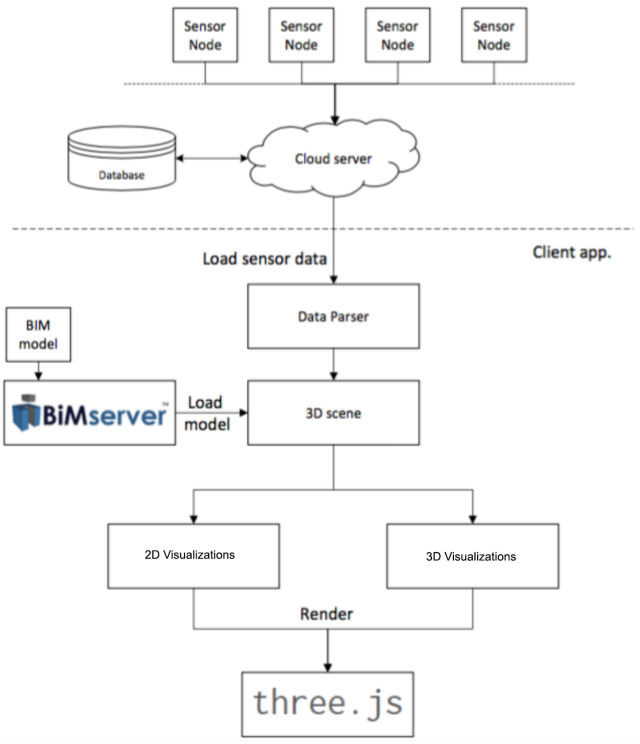

Fig. 2. Architecture overview for the 3DUI in Case A

is currently mostly used for visualization, however a light control interface was also developed as a proof-of-concept. The color of three RGB lights within the apartment can be switched through the interface; this was used to experiment with the principle of utilizing sensor readings from one IoT device to control another. For example, a temperature reading from a room can be hooked into controlling the lights. This concept can later be taken further to do other, more meaningful automated control operations, such as adjusting heating or air conditioning. Finally, it was found that the easiest way to develop an interactive 3D web scene utilizing a BIM model was to convert the BIM data into three.js [22] format. The overall architecture for Case A can be seen in Fig 2 while a screenshot of the application can be seen in Figure 7

B. Case B

In Case B, the target of the visualization is a multi story hospice building operated by a private company. As with Case A, we had no existing 3D models available, BIM or otherwise. This time, it was decided that instead of a BIM model, the building architecture is represented with a non-semantic mesh model. Using floor plans as the source material, the model was produced with 3Ds Max software utilizing a low-poly modeling paradigm similar to producing game assets. Instead of utilizing WebGL libraries, as in Case A, the application was developed with the Unity platform. The mesh model was imported into Unity as an FBX file and used as the basis for 3DUI development.

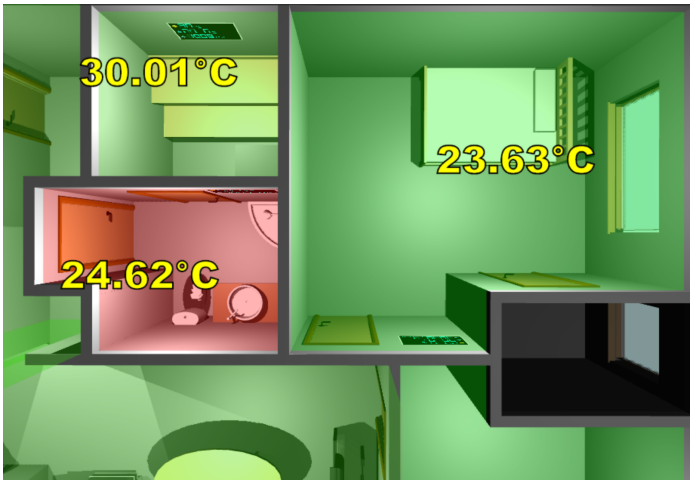

Fig. 3. Case A: Room types having different scales for what are interpreted as ”hot”, ”cold” and ”ok” readings

Fig. 4. A screenshot of the Case A 3DUI (company and project information omitted)

The requirements for the 3DUI came from a company involved in the construction project: only temperature data was to be visualized and no possibility for the controlling of devices was to be developed. The buildings utilized a commercial server software that provided a SOAP interface to sensor readings. While there was only a single data type to be visualized, the UI contains some alternatives for the visualization of the data similarly to Case A (text, bubbles, various icons, floor-overlaid heat maps, fire and ice). A screenshot of the Case B 3DUI can be seen in Figure 7. The building was significantly larger than in Case A. For this reason, no 3D objects acting as sensors were placed in the scene, but instead, the room floors were used for different visualization options (for see Fig 5 for an example). Because the building in question has several floors, the UI also contains a slider to switch between floors that are the target of the visualization. An overview of the architecture of Case B can be seen in Figure 6.

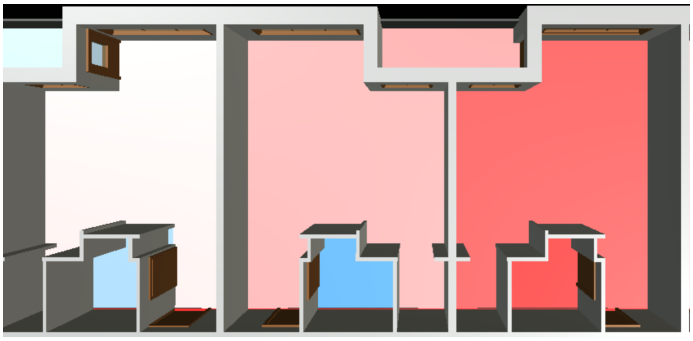

Fig. 5. Red and blue colors displaying too hot or too cold room temperatures in Case B. The stronger the hue, the further the temperature is deviating from optimal

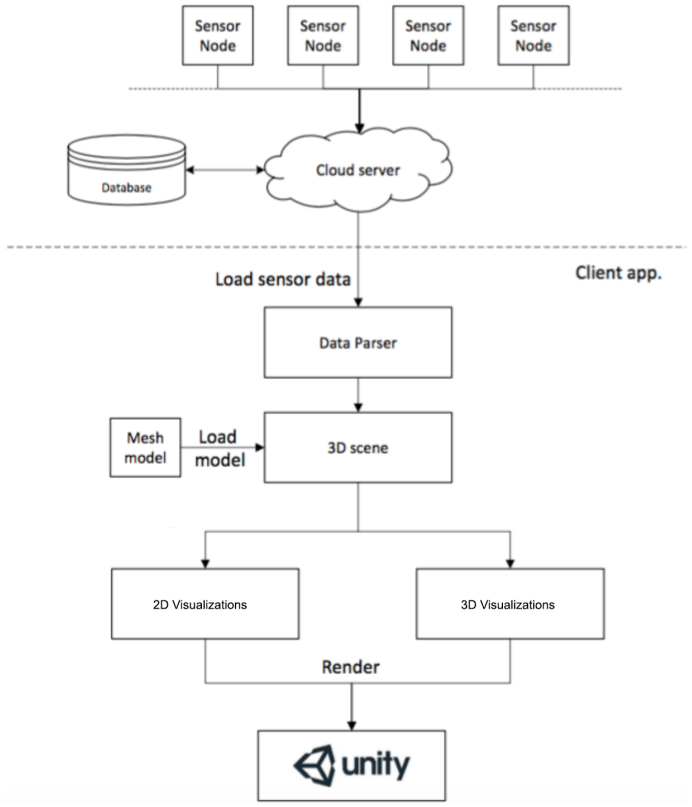

Fig. 6. Architecture overview for the 3DUI in Case B

C. Case architectures and their differences

While both cases provide the same basic functionality, their architectures have differences. The following section summarizes the architectures. The architectures comprise of the following components:

- Sensor Nodes: Each sensor node represents a physical sensor deployed at a certain location within the smart apartment. A sensor node is responsible for capturing the environmental parameters that the sensor can capture within its sensing range and submitting it to the cloud server. In Case A, the sensors were developed and manufactured by two different sources to test the system’s

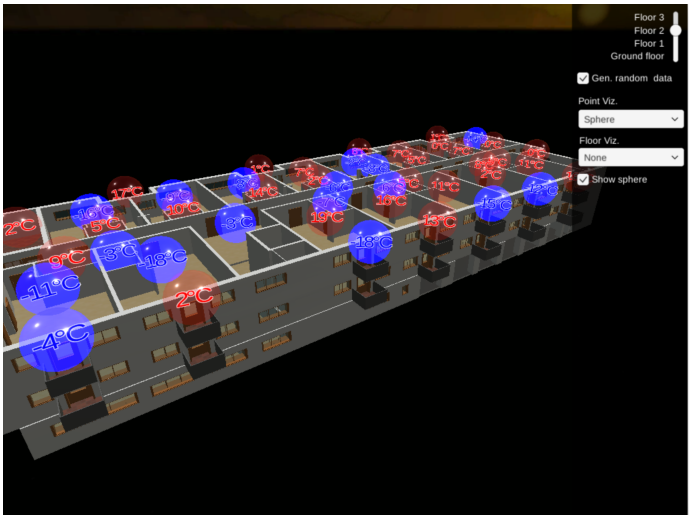

Fig. 7. A screenshot of the Case B 3DUI with randomized sensor data for demonstration

capability to utilize a heterogeneous sensor network. In Case B, the existing sensors of the building were utilized.

- Cloud server: The server acts as the centralized cloud that connects the real environments with the visualization application. The server receives data from the sensor nodes and persists them to the database. The visualization application requests the sensors' data from the server to build the visualizations based on them. The sensors' network transmits the collected data periodically to a centralized database deployed on a webserver. In Case A, the cloud server was deployed on a ThingSpeak IoT platform. The server exposes an API for the visualization application to use to request the sensors' data in JSON format. In Case B, a commercial server was utilized. The webserver utilizes Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP) to enable the visualization frontend application to fetch and create the sensor visualizations. SOAP protocol uses XML based data format to serialize and transmit the data.
- Database: this is the data storage for the sensors' data collected by the sensor nodes network. For security reasons, the cloud server is the only entity that can connect to the database to read and write data.
- Data Parser: This component is responsible for loading the data from the server and parsing it into unified data structures that can be used by the rest of the application using a standard format and API.
- 3D scene: this component is where the 3D environment is constructed. By this stage, the 3D model must be loaded and ready to be positioned in the 3D scene. Materials, lighting, camera, renderers, and the rest of 3D scene construction components are initialized in this stage.
- Building model: This is the model that defines the physical appearance and the properties of the building that is visualized. In Case A, the model was in the BIM format (IFC) hosted in a BIM server. In Case B, a game-engine ready mesh model was utilized (FBX). 2D visualizer: 2D visualizations show the value of the environmental parameters at the location of the sensor using icons or text. 3D visualizer: This component works is a similar manner to the 2D visualizer, but instead of utilizing icons, it creates visualizations laid on the building's physical properties, such as walls or floors.
- Rendering/game engine: In Case A, three.js [22] was utilized to render the final 3D scene for the user. three.js [22] is an open-source JavaScript library used to create computer graphics simulation running in a WebGL canvas in a web browser. In Case B, Unity game engine was utilized and the client application was programmed using the C# language. The application was converted to a WebGL browser application using Unity WebGL player.

Finally, the relevant differences between the two cases can bee seen in Table I.

TABLE I
THE RELEVANT DIFFERENCES BETWEEN CASE A AND CASE B

	Case A	**Case B**
IDE	three.js	Unity
Data format	JSON	XML
Programming language	JavaScript	C#
Data source	ThingSpeak	SOAP web service
Building model	BIM	Mesh
Building type	apartment	multi-story
Sensor data	temperature, humidity, air quality, air pressure	temperature

III. RESULTS AND DISCUSSION

The cases were compared according to performance measurements of each 3DUI application. In addition, interesting findings were found in the experiences gained during the development of both UIs.

A. Measurements

To compare the performance of both 3DUIs, we ran a small experiment in which we started each project and experimented with all visualizations and controls. We monitored the frames per second (FPS) count during the experiment. Both 3DUIs were ran on the same computer having the following specifications: Intel Core i7 6700HQ CPU, 16GB RAM, NVIDIA GeForce GTX 960M, memory 4MB GDDR5 GPU. Results showed that Case A kept a steady frame rate of 59-60 fps with all visualizations enabled. Case B kept a steady 59 fps. Both applications had occasional drops at or below 55 fps; Case A with very fast viewport manipulation and Case B when switching between visualizations or floors. According to this experiment, the 3DUIs did not have significant differences in performance.

TABLE II
DATA TRANSFER RATES FOR DATA REQUESTS IN BOTH CASES

	Request time	Response size	Speed
Case A	0.15s	17.8 KB	118.7 KB/s
Case B	0.04s	0.42 KB	10.5 KB/s

1) Model sizes: Case A utilized a BIM model for the building architecture while Case B utilized a mesh model. As BIM models contain semantic data besides architecture, the file sizes tend to be larger in comparison to mesh models. The ifc file defining the Case A building data was 27.4 MB while the FBX defining the architecturally more complex building in Case B was 3.74 MB. As the 3DUI in Case A utilized three.js [22] and WebGL for rendering, the ifc model was converted to json before loading the scene. The converted json file was 10.3 MB in size.

2) Sensor architecture: To measure the performance of each system in terms of sensor data requests, we measured the periods between the issued requests and successfully returned results. In project A, a GET request with the authentication key as a parameter is required to fetch sensor data. The webserver returns the results in JSON format and contains recent readings from all sensors, 100 records in total. The result was typically fetched between 0.1 to 0.3 seconds, 0.15 seconds in average. The result file is 17.8 KB in size. In Case B, a POST request is sent to a SOAP web service separately for each sensor node. A request parameter contains the sensor's unique ID. The request takes between 0.03 to 0.05 seconds to return a result. While this is a shorter period than in Case A, it should be noted that the request in Case A returns readings for all sensors while in Case B the requests have to be issued for each sensor separately. Dividing the response size by request time we can approximate the data transfer speeds for Case A as 118.7 KB/s and Case B as 10.5 KB/s. The results of the tests are summarized in the Table II.

Examining the responses further, it can be seen that the sensor architecture in Case B is disadvantageous in comparison to Case A. While SOAP format is language and platform independent and utilizes human-readable XML, the response sizes are bloated. Text delimiters and extra XML tags generated by the SOAP standard results in large file sizes, and eventually, slower transfer speeds. An actual response for reading a single sensor value can be seen below:

```
<soap:Envelopexmlns:xsi=http://www.w3.org
    /2001/XMLSchema-instance
  xmlns:xsd=http://www.w3.org/2001/
      XMLSchema
  xmlns:soap="http://schemas.xmlsoap.org/
      soap/envelope/">
  <soap:Body>
    <getPointDataResponse xmlns="http://
        OMITTED.XX/webVision/">
    <getPointDataResult>23.673</
        getPointDataResult>
    </getPointDataResponse>
  </soap:Body>
</soap:Envelope>
```

In Case A, the responses are obtained in JSON format, which can be formatted in a less verbose manner. In principle, a simple key-value pair, such as {"data": 23.673} could be used to transmit the same information as the multi-line SOAP response seen above.

B. Experiences

In Case A, the hypothesis was that the semantic properties of the BIM model can be easily leveraged for the 3DUI. However, as the project progressed, it became evident that the semantic properties will remain largely unused. The only semantic BIM property utilized was the *"IfcSpace"* property that was used to define the volumes of each room. The property was then used to generate translucent overlays for the rooms for color visualizations of temperature, humidity or carbon dioxide levels (see Fig ref. ColorOverlays). During specification gathering, it was also assumed that 3D locations of each IoT device would be convenient to fetch from the BIM model itself. However, as devices were placed and removed frequently throughout the project, it became easier to simply place visualizations in the 3DUI at room granularity according to the ThingSpeak channel specifications prepared for the purpose. While the Open Source BIMServer [20] provides a rich interface for a multitude of runtime operations, it was only necessary to utilize it for downloading the ifc file from the server. However, later in the project, even the ifc download was uncommon rendering the BIMServer largely unused. Because the BIM model itself changed infrequently, it became common to not utilize the BIM server to download the building data, but instead to simply use a previously converted three.js [22] json file for scene loding to save bandwidth and download time.

In Case B, the building model was produced with 3DsMax as a low-polygon model common among game assets. In this way, it was possible to produce a useful 3D model of a far more complex building while keeping the file size much smaller in comparison to Case A.

1) Programming languages: As stated earlier, three.js [22] and JavaScript was used to development in Case A, and Unity with C# programming language in Case B. According to the experiences using both SDKs in the cases, Unity was more productive and easier for development. The visual editor provided by Unity (and game engine platforms in general [24]) is indispensable for any non-trivial 3D project development. Besides, C# is a strongly typed object-oriented language that is suitable for writing extensible and maintainable code. JavaScript neither strongly typed nor object-oriented language. This makes writing maintainable code with it more challenging. In addition, Unity provides a rich framework of component-based features that are easy to integrate with existing code. Considering these aspects, we claim that Unity currently provides a more productive pipeline for developing

Fig. 8. The "IfcSpace" property used for color overlays for visualizing sensor data in Case A

3D applications compared to three.js. However, we acknowledge that this might change as tools and platforms for WebGL development mature.

IV. Conclusion

While BIM models provide potential to be utilized through building lifecycle, the disadvantages associated with BIM may outweigh the benefits in simple smart home or IoT visualization and/or control applications. BIM models might be too complex for real-time utilization or contain elements that must be removed for the models to be useful in visualization. The effort needed to finalize a BIM for 3D application development may in simple cases exceed the efforts needed for producing the models with conventional 3D content production means. For these workflows, existing 2D documentation (which typically is available) reduces the need for measuring activities thus speeding up otherwise manual model building. Further, it should be noted that BIM is not a ubiquitous solution for a large amount of building stock erected prior to emergence of digital workflows in AEC. The selection of the most efficient method is therefore dependant on the available documentation of the building, its size and complexity.

Currently BIM-models must be converted to other formats, such as FBX to be utilized in game engines, however there are web-tools and third-party libraries that can visualize BIM models real-time in WebGL. However, current game-engines provide visual editors and other additional tools that make the development of 3D visualization applications easier in comparison to WebGL libraries. This makes it attractive to utilize simple mesh-models with game engines instead of BIM. The emergence of fully browser based game engines (e.g. Babylon.js) is reducing the effort needed in WebGL app development. In addition, the tools allowing browsed based application development of BIM are also being developed (e.g. 4BIM [25]). According to our experiences so far, we can answer the questions formulated in the focus group interviews, *"What is the easiest way to produce a building model for 3D visualization, and what is the simplest possible functioning model?"* and *"Where and in which form should the IoT data be hosted and streamed"* as follows. A game-engine compatible low-polygon mesh model is the simplest working model; it can be quickly produced using floor plans by anyone familiar with basics of 3D modeling. For hosting and streaming sensor data, the ThingSpeak architecture was superior to the SOAP Webserver.

In future work, it might be useful to perform a similar study to significantly more complex apartment models to further investigate the differences reported in this paper. Regarding the current cases, we will be concentrating on the usability aspects of 3DUIs presented above, for example, viewport manipulation, mobile device usability and the performance of the systems as Information Visualization tools. In addition, we will be introducing Case C, which will be a suburban house. The current 3DUIs were focusing mostly on visualization, the simple light controls in Case A being the only way to control the IoT devices in the buildings. However, the future work will bring increased focus into controlling of IoT devices through a 3DUI in building systems. Finally, other web platforms besides three.js should be included in the analysis.

Acknowledgments

This work has been supported by the TEKES VIRPA-A project, as well as the 6Aika: Open City Model as Open Innovation Platform project (A71143) 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.

References

[1] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012.

[2] B. Firner, R. S. Moore, R. Howard, R. P. Martin, and Y. Zhang, "Poster: Smart buildings, sensor networks, and the internet of things," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 337–338.

[3] A. Buckman, M. Mayfield, and S. BM Beck, "What is a smart building?" *Smart and Sustainable Built Environment*, vol. 3, no. 2, pp. 92–109, 2014.

[4] D.-M. Han and J.-H. Lim, "Design and implementation of smart home energy management systems based on zigbee," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, 2010.

[5] F. Zafari, I. Papapanagiotou, and K. Christidis, "Microlocation for internet-of-things-equipped smart buildings," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 96–112, 2016.

[6] A. Lovett, K. Appleton, B. Warren-Kretzschmar, and C. Von Haaren, "Using 3d visualization methods in landscape planning: An evaluation of options and practical issues," *Landscape and Urban Planning*, vol. 142, pp. 85–94, 2015.

[7] P. Paar, "Landscape visualizations: Applications and requirements of 3d visualization software for environmental planning," *Computers, environment and urban systems*, vol. 30, no. 6, pp. 815–839, 2006.

[8] T. Alatalo, M. Pouke, T. Koskela, T. Hurskainen, C. Florea, and T. Ojala, "Two real-world case studies on 3d web applications for participatory urban planning," in *Proceedings of the 22nd International Conference on 3D Web Technology*. ACM, 2017, p. 11.

[9] N. M. Boers, D. Chodos, J. Huang, P. Gburzynski, I. Nikolaidis, and E. Stroulia, "The smart condo: Visualizing independent living environments in a virtual world," in *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*. IEEE, 2009, pp. 1–8.

[10] M. Pouke and J. Häkkilä, "Elderly healthcare monitoring using an avatar-based 3d virtual environment," *International journal of environmental research and public health*, vol. 10, no. 12, pp. 7283–7298, 2013.

[11] M. Pakanen, L. Arhippainen, and S. Hickey, "Designing for 3d user experience in tablet context," *Design and Early Phase User Evaluation of Four 3D GUIs, International Journal on Advances in Intelligent Systems*, vol. 5, no. 3.

[12] A. Motamedi, A. Hammad, and Y. Asen, "Knowledge-assisted bim-based visual analytics for failure root cause detection in facilities management," *Automation in construction*, vol. 43, pp. 73–83, 2014.

[13] J. Irizarry, E. P. Karan, and F. Jalaei, "Integrating bim and gis to improve the visual monitoring of construction supply chain management," *Automation in Construction*, vol. 31, pp. 241–254, 2013.

[14] W. Yan, C. Culp, and R. Graf, "Integrating bim and gaming for real-time interactive architectural visualization," *Automation in Construction*, vol. 20, no. 4, pp. 446–458, 2011.

[15] J.-P. Virtanen, M. Kurkela, H. Hyyppä, S. Niemi, S. Kalliokoski, S. Vanhatalo, J. Hyyppä, H. Haggrén, S. Nenonen, J.-M. Junnonen *et al.*, "Visualization of building models and sensor data using open 3d platforms," in *International council for research and innovation in building and construction conference*, 2016.

[16] B. Hagedorn and J. Döllner, "High-level web service for 3d building information visualization and analysis," in *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. ACM, 2007, p. 8.

[17] B. Becerik-Gerber, F. Jazizadeh, N. Li, and G. Calis, "Application areas and data requirements for bim-enabled facilities management," *Journal of construction engineering and management*, vol. 138, no. 3, pp. 431–442, 2011.

[18] J. Zhang, Q. Liu, F. Yu, Z. Hu, and W. Zhao, "A framework of cloud-computing-based bim service for building lifecycle," in *Computing in Civil and Building Engineering (2014)*, 2014, pp. 1514–1521.

[19] W.-L. Lee, M.-H. Tsai, C.-H. Yang, J.-R. Juang, and J.-Y. Su, "V3dm+: Bim interactive collaboration system for facility management," *Visualization in Engineering*, vol. 4, no. 1, p. 5, 2016.

[20] J. Beetz, L. van Berlo, R. de Laat, and P. van den Helm, "Bimserver. org–an open source ifc model server," in *Proceedings of the CIP W78 conference*, 2010.

[21] "Thingspeak," https://se.mathworks.com/help/thingspeak/, accessed: 2017-29-06.

[22] R. Cabello *et al.*, "Three. js," *https://github. com/mrdoob/three.js, year=2010*.

[23] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Visual Languages, 1996. Proceedings., IEEE Symposium on*. IEEE, 1996, pp. 336–343.

[24] J. Gregory, *Game engine architecture*. crc Press, 2009.

[25] "4bim," http://www.openbim.org/case-studies/4-bim, accessed: 2017-29-06.