FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

**Mahmoud Badri**

# COMPARISON OF TWO APPROACHES FOR WEB-BASED 3D VISUALIZATION OF SMART BUILDING SENSOR DATA

Master's Thesis
Degree Programme in Computer Science and Engineering
May 2017

**Badri M. (2017) Comparison of two approaches for web-based 3D visualization of smart building sensor data.** University of Oulu, Degree Programme in Computer Science and Engineering. Master's Thesis, 54 p.

# ABSTRACT

**This thesis presents a comparative study on two different approaches for visualizing sensor data collected from smart buildings on the web using 3D virtual environments. The sensor data is provided by sensors that are deployed in real buildings to measure several environmental parameters including temperature, humidity, air quality and air pressure. The first approach uses the three.js WebGL framework to create the 3D model of a smart apartment where sensor data is illustrated with point and wall visualizations. Point visualizations show sensor values at the real locations of the sensors using text, icons or a mixture of the two. Wall visualizations display sensor values inside panels placed on the interior walls of the apartment. The second approach uses the Unity game engine to create the 3D model of a 4-floored hospice where sensor data is illustrated with aforementioned point visualizations and floor visualizations, where the sensor values are shown on the floor around the location of the sensors in form of color or other effects. The two approaches are compared with respect to their technical performance in terms of rendering speed, model size and request size, and with respect to the relative advantages and disadvantages of the two development environments as experienced in this thesis.**

**Keywords: 3D visualization, smart buildings, sensor visualization**

# TABLE OF CONTENTS

# FOREWORD

This thesis was done at the Center for Ubiquitous computer at the University of Oulu in the VIRPA-A project funded by the Finnish Funding Agency for Technology and Innovation (TEKES). I would like to thank my supervisors Dr. Matti Pouke and Prof. Timo Ojala for their continuous support during my studies and thesis. On a wider perspective, I would like to express my gratitude to the Center for Ubiquitous Computer for granting me the opportunity to study and work. I learnt a lot while working there, and it will always be remembered as an important milestone in my professional career. I would like to thank my wife for her support, my friends Mounib Mazouzi and San Günes for being great team members, study mates and supporters during our joint two years master's journey. Finally, I am grateful to Finland for giving me the opportunity to pursue my master's degree.

Oulu, 26.5.2017,

Mahmoud Badri

# ABBREVIATIONS

| | |
|---|---|
| API | Application programming interface |
| AR | Augmented reality |
| ART Renderer | Autodesk Raytracer Renderer |
| BAS | Building automation system |
| BIM | Building information modeling |
| CGI | Common gateway interface |
| CMYK | Cian magenta yellow key |
| CSS | Cascading style sheets |
| CPU | Central processing unit |
| DOM | Document object model |
| FPS | First person shooter |
| fps | Frames per second |
| GDDR | Graphics double data rate |
| GIS | Geographic information system |
| GLSL | OpenGL Shading Language |
| GPS | Global positioning system |
| GPU | Graphics processing unit |
| HEX | Hexadecimal |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext transfer protocol |
| HSV | Hue saturation value |
| HVAC | Heating, ventilation and cooling |
| ICT | Internet communication technology |
| IDE | Integrated development environment |
| IFC | Industry foundation classes |
| IK | Inverse kinematics |
| infovis | Information visualization |
| IoT | Internet of things |
| IP | Internet protocol |
| JSON | JavaScript object notation |
| Lab | Lab color space (lightness, a and b are color opponents) |
| MIB | Management information base |
| NURBS | Non-Uniform Rational B-Splines |
| OWA | Ordered weighted averaging |
| pH | potential of hydrogen |
| RGB | Red green blue |
| SDK | Software development kit |
| SH | Smart home |
| SNMP | Simple network management protocol |
| SOAP | Simple Object Success Protocol |
| TCP | Transmission control protocol |
| UI | User interface |
| VE | Virtual environment |
| VR | Virtual reality |
| VS-CaSP | visualization system of context-aware application scenario planning |
| WiFi | Wireless fidelity |
| WSANs | Wireless sensor and actuator networks |

| | |
|---|---|
| WSDL | Web Services Description Language |
| WSN | Wireless sensor network |
| WWW | World wide web |
| WYSIWYG | What you see is what you get |
| XML | Extensible markup language |

# 1. INTRODUCTION

## 1.1. Background

Internet of things (IoT) refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence. IoT will increase the ubiquity of the Internet by integrating every object for interaction via embedded systems, which leads to a highly distributed network of devices communicating with human beings as well as other devices [36]. IoT is becoming increasingly more popular recently. This is made possible due to several factors like the widely spread high-speed internet we witness today. The cost of using the internet is dropping constantly also. Additionally, more and more internet-enabled devices are being manufactured as time goes by. Technology costs are going down constantly, and at the same time, the use of smart internet-enabled devices is increasing at a very high rate. These factors besides others, created a perfect atmosphere for the flourishment of IoT era. Although IoT is a hot topic recently, the idea is not actually new. Considered the first IoT implementation, John Romkey created a toaster that could be turned on and off over the Internet in 1990 [44]. The toaster device was connected to a computer with TCP/IP networking. It then used an information base (SNMP MIB) to switch the power on and off. Internet enabled devices became popular at the beginning of the millennium due to the aforementioned factors. IoT has many applications in smart homes (SH), wearables, smart cities, smart grids, industrial internet, connected cars, and many more. We are concerned here in the applications of IoT for smart buildings.

In this context, we refer to the deployment of a wireless sensor network (WSN) in a building with the goal of sensing some environmental parameters such as temperature, humidity, air quality, air pressure, lighting, and more. Smart home sensor networks pose new challenges compared to regular sensor. A smart home sensor must consume low energy, be relatively cheap and most importantly achieve high levels of security and privacy for apparent reasons. Deploying and enabling the sensor network is one side of the process. The other side is how to visualize the values collected by the sensor network and present them to the user in a user-friendly manner, which is the topic of this work. Typically, user interface (UI) for graphical visualization applications is implemented using common user interface elements like panels, icons, menus, and images. Standard user interface elements work fine is most of the cases to deliver the intended function of the visualization. Yet, many studies suggested that the use of 3D interfaces provide users with faster and more usable experience [45]. We believe that visualizing the sensors data in a 3D virtual environment (VE) will provide an intuitive interface for the end user and thus it is greatly more usability compared to the standard visualization techniques. The 3D virtual interface also provides a sense of presence to the user improving the situational awareness allowing the user better control and giving him an immersive experience [50].

## 1.2. Thesis overview

This thesis compares two different approaches dubbed as project A and project B for visualizing sensor data in 3D virtual environments. Each project visualizes sensor data in a specific smart building and is built using a specific technology and model format. Project A visualizes data in a small smart apartment located inside VTT research center in Oulu, Finland. Project A visualizes temperature, humidity, air quality, and air pressure values measured by sensors installed in the research smart apartment. Project A was developed using standard web technologies, JavaScript, HTML5 and CSS. The smart apartment model was built in Autodesk Revit using the BIM format [37]. More explanations on this in the design chapter. In project B the 3D model of a large 4-floored hospice was built using the Unity engine. We exported a web application build so that both projects run in a web browser. In the rest of this section, we present definitions for the software and formats used in the two projects.

## 1.3. Research objective

In this thesis, we study visualization techniques for graphical representation of sensor values in a 3D virtual environment. More specifically, our intention is to investigate various visualization techniques and categories, and compare them in terms of usability and look. In addition, we seek to compare Project A and Project B in terms of performance, ease of development, and visualizations quality. We also highlight the benefits and shortcomings of each project settings. Moreover, we compare BIM and mesh models and their suitability for this sort of visualization applications.

## 1.4. Contribution

This work contributes to the field of data and information visualization in 3D virtual environments. More specifically, it contributes to the smart home sensor visualization research domain. We create and test numerous visualization techniques and compare them. We also use two independent technologies: Unity and three.js, two model formats: standard mesh and BIM. Results from this work provide a constructive knowledge highlighting the benefits and drawbacks of different techniques for creating sensor visualizations in 3D virtual environments.

## 1.5. Thesis structure

The rest of the thesis is structured as follows: Chapter 2 provides a comprehensive look at the background of IoT and sensors visualization. This includes surveying four areas relevant to our study: urban sensing, smart buildings, data visualization and information visualization. Chpater 3 presents detailed information about the design and implementation of the two projects (Project A and project B). Chapter 4 presents the evaluation of the two projects and the comparison between them. It shows also the lessons learnt, pros and cons of different approaches and draws recommendations for 3D visualizations based on our research. Chapter 5 provides a discussion of the findings from the evaluation chapter and our recommendations based on the it. Chapter 6 lists suggested future work to extend the current projects work, provides a summary of the work done and draws a conclusion.

# 2. LITERATURE SURVEY

This section provides a thorough look at the previous research work done in four relevant fields to this research work: urban sensing, smart houses, data visualization, and information visualization. We order and survey the four research areas based on relevance to our study starting from the broader topic of urban sensring. We finish by the tightly related topic of information visualization.

## 2.1. Urban sensing

Urban Sensing is the research area investigating the use of environmental wireless sensors in urban environments for measuring various environmental variables. Urban Sensing uses many of the ubiquitous computing paradigms by deploying a multitude of data collection devices to collect data about the environment and use it to contribute to improving the urban life. We discuss in this subsection some of the work done in this research domain. Dutta et al. [2] presented 'Common sense', a mobile participatory sensing system that allows individuals and groups to measure air quality. The system works by using handheld sensor-enabled devices to capture, process, and disseminate sensor data. The visualization of the sensor data is implemented in a browser-accessible web portal. A similar work by Campbell et al. [4] proposed a people-centric urban sensing application 'MetroSense' which collect large scale data. The main quest this work tried to answer is how to move urban sensing from small-scale application-specific network, into largescale sensing system serving people's everyday lives. The main challenge for such systems is how to build such generic urban sensing system for the public on a largescale and allowing it to fit a large set of applications in urban environments. MetroSense used opportunistic sensor networking approach to allow the system to scale to very large areas. Another related work by Resch et al. [6] is the Common Scents project. A flexible, interoperable and portable real-time data integration and analysis system for air quality assessment. The system aims to establish an open and modular infrastructure. The paper demonstrated common issues for bringing out pervasive urban sensor networks and associated concerns to fine-grained information provision. The system was deployed on top of a sensor network called CitySense in the City of Cambridge, MA US.

Voogt et al. [5] presented a review of the use of thermal remote sensing over an urban environment in the study of urban climates, with main focus on the urban heat island effect. The review revealed that the progress in thermal remote sensing of urban areas has been slow due to some factors like the tendency to use qualitatively based land use data to describe the urban surface rather than the use of more fundamental surface descriptors. In addition, advances in the development of new satellite based sensors would contribute to improving the sensing process. The review suggests advancing three research areas to improve thermal remote sensing: (1) determining appropriate surface properties, (2) Join canopy radiative transfer models with sensor view and surface energy balance models, (3) doing more observational studies to get better knowledge of surface properties.

Ono et al. [8] explained an urban sensing system built as part of an experiment in downtown Tokyo. The study also evaluated the effective placement of sensors within the city and the relation between the trend of temperature and the environment. While Ono evaluated the effective placement of sensors within the city, the roles of people in a sensors network was discussed by Lane et al. [7]. Lane defined two important factors of human involvement with sensor networks: participatory and opportunistic sensing. Those two factors are on two end points of a scale. At one end, people are actively interacting with the system (participatory) while on the other endpoint people are providing the data to the system unconsciously (opportunistic). The authors argue, based on the evaluation results that opportunistic sensing can scale up to bigger applications easily compared to participatory sensing. Coming to the security and privacy of urban sensing system, De Cristofaro et al. [1] investigated the privacy of urban sensing systems in details. They illustrated that privacy issues appear due to the presence of multiple actors, such as infrastructure operators, device owners, queriers, and so on. They addressed the problem of protecting secrecy of reported data and confidentiality of query interests from malicious entities. They proposed two adversarial approaches: resident and non-resident, and two strategies available to malicious attackers depending on whether the attacker is randomly distributed or local to a specific region of the network. For each setting, a security technique was developed that trades off between privacy level and potential communication overhead.

## 2.2. Smart buildings

A smart building or home (both refer to the same concept) is any structure that uses automated processes to automatically control the building's operations including heating, ventilation, air conditioning, lighting, security and other systems [61]. 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 buildings. We discuss in this subsection some of the research work done pertaining to smart buildings. Figure 1 shows a sample design for a smart building.

Farias et al. [9] presented a decentralized control and decision-making system for smart building applications called 'CONDE'. Farias claims that the use of decentralized wireless sensor and actuator network (WSAN) saves energy compared to using the common centralized WSAN. Experiments on CONDE showed that it has advantages compared to its peers in response time, overall system efficiency and energy saving for both the building and the WSAN. CONDE promotes the integration of different applications within the WSAN in order to achieve the energy efficiency.

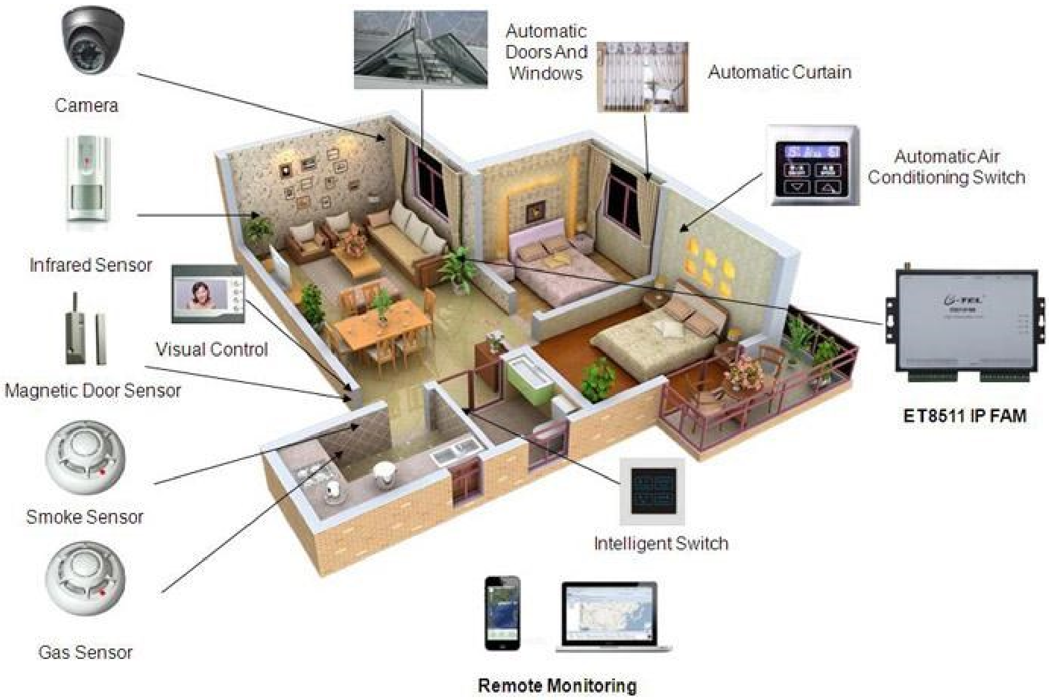


Figure 1. Example smart building design [62]

Meyer et al. [10] claims that wireless networks and ubiquitous devices powered by intelligent computation will blend into people's life in the near future. Meyer provided a multitude of futuristic scenarios for the context-aware home vision and demonstrated how context-aware home will allow its occupants to live a safe, supportive, convenient, pleasant, enjoyable, entertaining, and relaxing life.

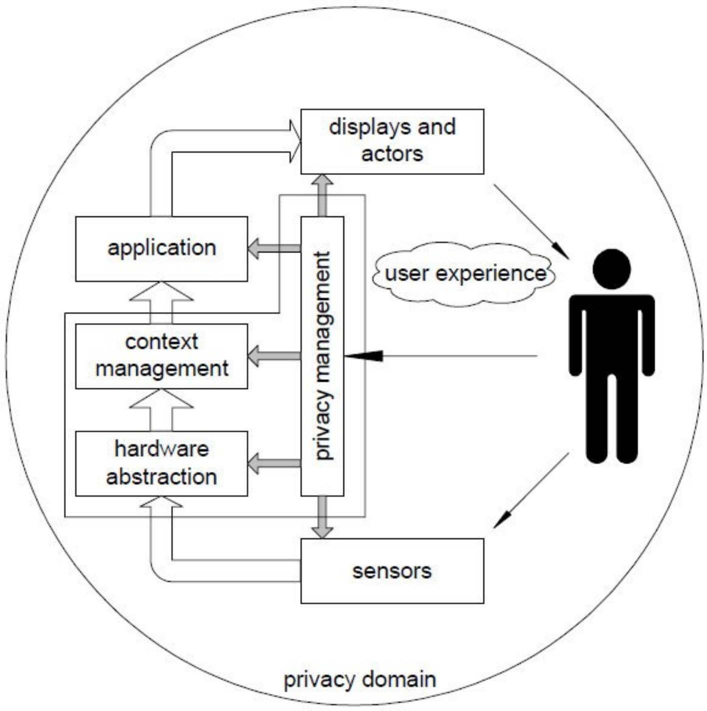


Figure 2. Basic components of a context-aware system

Robles et al. [11] presented a review of the context-aware tools for smart home development. Robles defined Context awareness as the idea that computers can both sense, and react based on their environment. He also defined a smart home as a home that is equipped with special structured wiring to enable occupants to remotely control or program an array of automated home electronic devices. Additionally, he gave some examples of smart home products and their functions like cameras, lamps, door phones, motion sensors and more. Likewise, he highlighted some benefits of a smart home like making life easier, peace of mind, security, energy efficiency and the promise of tremendous benefits for elderly people living alone, for example: notifications of medicine time, automatic hospital alert in case of resident fall, tracking food and water intake, and many more. The paper concluded by listing some challenges of installing smart home systems like balancing the system complexity versus usability. Some factors to consider are the system size, system components, how intuitive the system will be, number of system users, system administrators, and feasibility of making changes to the system interface.

Kleissl et al. [12] carried out a study on smart buildings with the main goal of optimizing energy consumption. The study treated a modern building as a cyber-physical energy system and examined opportunities presented by optimizing energy use by its owners and information processing equipment. The paper provided a classification of different types of buildings and their energy consumption. It also pointed out some opportunities available to improve energy efficiency through applying various strategies from lighting to computing. A similar work aiming for energy savings in smart buildings was done by Lu et al. [18]. They presented “Smart thermostat”, an approach to save energy by sensing occupancy and sleep patterns in a home and automatically turning off the home’s heating, ventilation and cooling systems (HVAC) when occupants are sleeping or away. The study stated that heating, ventilation and cooling are the largest contributors to a home’s electricity bill. Smart thermostat system used cheap and simple motion and door sensors installed throughout the home. Evaluation results indicated that the smart thermostat could provide large energy savings. On average, it was demonstrated that using smart thermostat could save about 28% of residential HVAC energy consumption. The low cost of this system which is estimated to be 25$ per home, makes it potential for a large impact on nations’ improvements in the energy efficiency sector.

Morvaj et al. [13] studied smart buildings as a basic building block for the wider vision of a smart city. The paper presented an overview of the features of a smart city with focus on energy efficiency. Morvaj demonstrated the basic features of smart buildings and built a draft model using system dynamics in ‘Anylogic’ to highlight those features. Morvaj stated that the core infrastructure of smart city are its citizens, water and energy, communication, business, transport and city services. Morvaj provided another definition of a smart building as a building that uses subsystems for managing and controlling renewable energy sources, house appliances and energy consumption using most often a wireless communication technology. According to Morvaj, a smart building consists of sensors, actuators, controllers, central unit, interface, network and smart meter.

Wang et al. [14] proposed a building indoor energy and comfort management model based on information fusion using ordered weighted averaging (OWA) aggregation. As the main advantages, a smart building include high-level comfort and high power efficiency, Wang built a multi-agent control system with heuristic intelligent optimization to achieve those two essential requirements. As the name implies, the multi-agent system is made up of multiple agents. Each agent has a specific function but all the agents share some common characteristics. Wang stated that thermal comfort, visual comfort and air quality are the main comfort factors in a smart building. The temperature, illumination level and the $CO_2$ concentration are used to indicate these three main comfort factors. The study presented several case studies and simulation results indicating the effectiveness of the control system in achieving the control goal in different operating scenarios.

Främling et al. [16] described a distributed information architecture that makes it possible to implement smart environments on a large scale by integrating information access to and control of different building automation systems. Främling used the term “smart space” to signify a geographical space where information is available about the space itself, the devices and services available in it, the people present in it and about other potentially useful information or services. The implementation involved the deployment of an experimental building automation facility connecting together various building automation protocols.

Stefanov et al. [17] provided an analysis of the building blocks of smart houses, with focus on elderly and disabled people. Stefanov highlighted two approaches for realizing intelligent houses for the people with physical limitations: Special architecture solutions and Particular technological innovations. According to Stefanov, smart houses have a strong, positive, and emotional impact on persons with physical disabilities and elderly. Stefanov classified smart houses for elderly and disabled to five groups according to the type of disability: people with movement disabilities, elderly, people with low vision, hearing impaired people and cognitively impaired people. Each group has a primary focus when designing a smart house for it. Smart home-installed technology performs movement-independence of the user, health monitoring, entertainment, security, and physical rehabilitation. Stefanov presented a thorough survey of recent work on the smart homes for elderly and highlighted important issues for futuristic intelligent house models.

A similar yet more general concept to smart buildings is intelligent buildings, a concept that goes beyond smart buildings. Martins et al. [15] stated that there are many definitions for an intelligent building. One definition by the institute of the United States defines an intelligent building as “a building that provides a productive and cost-effective environment through optimization of its four basic elements including structures, systems, services and management and the interrelationships between them”. The European Intelligent Building Group provided a similar yet slightly modified definition: “Intelligent buildings provides a productive and cost-effective environment through optimization of its four basic elements including structures, systems, services and management and the interrelationships between them”. The three main features that an intelligent building must have are automatic control, learning abilities and occupancy trends incorporation. A relevant concept discussed by Martins et al. is “smart grids”. A smart grid is an electrical grid where all parties (suppliers and consumers) are digitally connected. Smart grids have two characteristics that are important from the intelligent building perspective. Those are consumer's active involvement and distinct generation and storage technologies.

### 2.3. Data visualization

Generally, visualization can be defined as the ability to make images within the mind's eye. Visualization is a natural function of the human brain. It is something that we are doing all the time effortlessly. Today when we refer to visualization, we usually mean a graphical representation of data or concepts with the purpose of getting useful insights from the data. We first look at the history of data visualization. Friendly [19] presented a comprehensive survey of the history of data visualization. He divided data visualization history into epochs, each of which is describable by coherent themes and labels. The earliest seeds of visualization arose in geometric diagrams, tables of the positions of stars, and navigation maps. Ancient Egyptians used coordinate system at least by 200 BC.

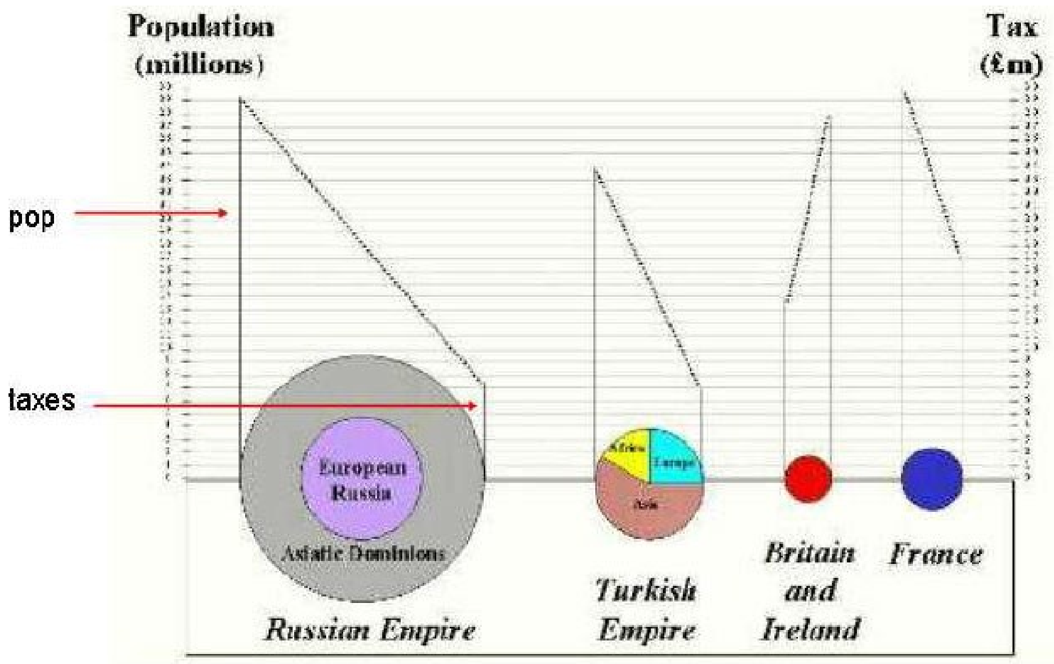


Figure 3. Pie-circle-line chart drawn in 1801 comparing population and taxes in several nations

The 17th century witnessed the rise of analytic geometry, coordinate systems, probability theory, demographic statistics and "political arithmetic", the study of population, land, taxes, value of goods, etc. The idea of graphic representation somewhat established in the 18th century. Maps started to show more data than just the geographical location, like contours and isolines. Several technological innovations of the 18th century like the three-color printer invented in 1710 helped the production and dissemination of graphic works. Furthermore, common modern graph types like line graph, bar chart, pie chart, and circle graph were first introduced in the 18th century by Willian Playfair. Figure 3 shows a redrawn graph by Playfair combining multiple graphing elements: circles, pies and lines.

Modern graphics started in the 19th century with a massive rise in statistical graphics and thematic mapping. New graph type were invented, for example the continuous shadings graph type. Graphs started to be used in economic and state planning. The late 1800s were considered the “golden age” of statistical graphics and thematic cartography with significant advances in inventions and usage of data visualization techniques. On the contrary, the early 1900s can be labeled the “modern dark ages” of visualization according to Friendly. This is due to few new graphical inventions, and less motivation towards graphical visualizations during this historical era. From 1950 to 1975, data visualization usage started to rise again. From 1975 until present, data visualization has blossomed into a mature, vibrant and multi-disciplinary research area. With the massive technological advances, advanced interactive and dynamic data visualization became possible.

Fan et al. [20] presented an approach to process and interpret the data gathered by sensor networks. The built system is a sensor networks consisting of a group of sensors and communication units deployed to monitor rare plants or other endangered species. The environmental data are collected and sent by the wireless sensor networks to an internet-enabled base station. Collected data is then visualized on a map created by a GPS system. Voronoi Diagrams are then used to partition the map into areas holding different weather attributes based on the data collected by the sensor network. Each partition is colored with a different color so that it is easy to visually distinguish between the different partitions. Generated visualization maps are made accessible on the web using a CGI program. The project has several visualization tools; one of them emphasizes the spatial distribution using. Another one shows the chronological charting of the data. In addition, many images were taken by cameras embedded in the sensor network to track the growth of the studied plants.

The choice of colors play an important role in making data visualization easier to understand. An interesting technique for choosing multiple effective colors for use during data visualization was described by Healey [22]. Literature suggests that we need to consider three separate effects during color selection: color distance, linear separation, and color category. The proposed method is a simple approach for measuring and controlling those three effects. Experimental results showed how it is possible to control color distance, linear separation, and color category to pick seven isoluminant colors that satisfy user requirements. Another work aiming to help designers in the color selection process is ColorBrewer.org, built by Harrower et al. [53]. ColorBrewer.org is an Online Tool for selecting effective color schemes for maps. The most prominent features of ColorBrewer include: (1) multi-hued perceptually ordered color schemes; (2) ‘learn more’ features that advise designers on some of the theory behind color usage and map making; (3) the ability to check how well different colors schemes match when additional map information is present; (4) the ability to check how well different colors schemes will perform when the mapped patterns are both ordered and complex; and (5) supporting color output specifications in five commonly used color systems (CMYK, RGB, HEX, Lab and HSV).

Su et al. [21] carried out a study that focuses on the application-layer simulation to propose a visualization system of context-aware application scenario planning (VS-

CaSP) for assisting non-technical developers in easily designing the application scenario of smart buildings and in performing the acceptable and predictable simulation and evaluation. The goal is to verify, and modify the desired context-aware application scenario in created virtual smart buildings before actual deployment processes start. To evaluate this approach, a complete case, including six scenarios, thirteen rule classes, and twenty-eight rules, to support the smart home services was created and evaluated by non-technical users. Three criteria were established for evaluation: Ease-of-Use, Professionalism, and Supportability. Results revealed a high Ease-of-Use and satisfied Supportability degrees indicating that the VS-CaSP is easy to use for the support of quickly design smart building applications, but not professional enough for developments of 3D modeling, animation, and rule expression due to the low degree of Professionalism. Thus, the system is more suited for non-technical users.

Buschmann et al. [23] presented a modular and extensible visualization framework for wireless sensor networks named “SpyGlass”. SpyGlass follows the typical sensor network data flow architecture (Figure 4) which consists of three major functional entities: The sensor network, the gateway nodes located in the sensor network and the visualization software. Visualization of SpyGlass consists of three main layers: background layer, relations layer where relations between the sensor networks are visualized by connecting nodes together, and the nodes layer where each sensor node is visualized. The system was tested for scalability and the results showed that it scales to thousands of nodes without problems.

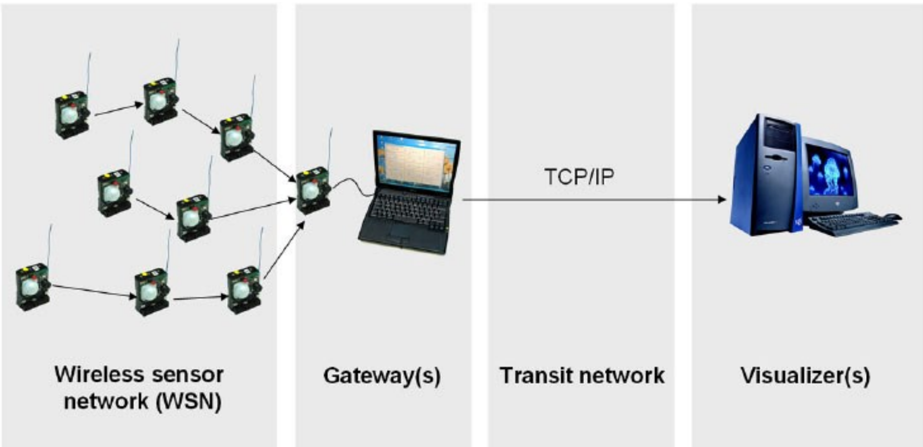


Figure 4. Typical sensor visualization architecture

A similar work was done by Hu et al. [24] to visualize a sensor network that monitors wetland. The sensor network collects environmental data, such as temperature, humidity, illumination intensity and pH around the wetland and sends it to the base station. The base station is connected to the visualization application on a computer. Network monitoring and visualization happens in real time. The system also visualizes the network topology, which can provide users with more direct view of the environmental and routing data around the studied wetland.

Another sensor visualization project is the smart condo [25]. The project aims to provide a comprehensive affordable high-quality healthcare to the elderly by recording important events and environmental parameters and sending them to a web-based system for analysis and visualization. The system provides both a 2D as well as a 3D visualization interfaces. The 2D visualization relies on a GIS component to visualize sensor network nodes in the form of markers on a map. The 2D visualization also displays the sensors readings over time. System users can filter the sensor network nodes and only view a subset of them. It is also possible to view a comparison between nodes of choice. The 3D visualization displays the subject in a 3D virtual model of the building. The system allows users to create the 3D environment and add walls and furniture. In addition, users can add their avatars to the 3D environment and then the avatar's location is updated according to the user's sensed location in reality.

Saito et al. [26] studied a two-tone pseudo coloring technique for effective visualization of big one-dimensional dataset. By painting with two colors at each scalar value in the pseudo coloring process, it is possible to visualize precise details within overview display without distortion. This technique provides several advantages. Firstly, it requires very small screen space for drawing the visualization. In addition, it allows the overview and details of the dataset to be present in one image simultaneously without distortion. Implementation of this technique is also straightforward. The presented technique proved to be effective in several applications, such as meteorological observation data and train convenience evaluation data.

### 2.4. Information visualization

Information visualization is defined as: the use of computer-supported, interactive visual representations of data to amplify cognition [27]. In this context, Information refers to abstract items, entities, things that do not have a direct physical correspondence. For example, baseball statistics, stock trends, connections between criminals, query results, text of books, etc. Figure 5 shows the relation between information and data visualization. Information visualization is a sub-set of data visualization. Information visualization process involves gathering relevant information and giving it a visual representation that is useful for analysis, decision-making, gaining insight, and more.

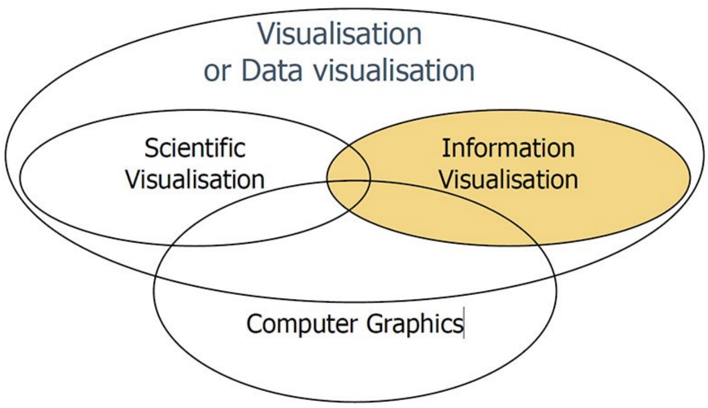


Figure 5. Relation between data and information visualizations

A relevant newer topic is knowledge visualization, which is defined by Burkhard and Meier (2004) as the use of visual representations to transfer knowledge between at least two persons. Chen et al. [29] conducted a meta-analysis study on information visualization aiming to find invariant underlying relations suggested collectively by the empirical findings in order to form an overview of the design and practices of information visualization. The study focused on three aspects of information visualization: users, tasks and tools. The study concluded the following results: (1) it is challenging to apply meta-analysis methods to information visualization studies, (2) individual differences should be investigated systematically in the future, (3) users tend to perform better with simpler user interfaces, (4) to extract conclusive results, larger study sample should be used. The study also recommended the use of standardized testing information, task taxonomies, and cognitive ability tests in order to improve the quality of similar future studies.

Cañas et al. [28] presented “CmapTools” software as an example of how concept maps, a knowledge visualization tool, can be combined with recent technology to provide integration between knowledge and information visualizations. Concept map is an effective way to represent a person’s understanding of a topic visually. Cmap Tools is a software that allows its users to represent their knowledge using concept maps domain of knowledge. The software was deployed on public servers to allow people from everywhere to access and use it on the internet. The user interface of CmapTools was designed to be very simple and easy to use by everyone even children. CmapTools provides another type of integration between knowledge and information representations by allowing users to search the internet from within a concept map. The concept map generated by the user can be used by CmapTools to search the web and retrieve results related to the concept map.

Kapler et al. [30] developed a novel visualization technique named “GeoTime” with the goal of displaying and tracking events, objects and activities within a combined temporal and geospatial display. The system capabilities include descriptive events and relationships, association analysis, event aggregation methods and geo-located linked charting. The study highlighted two useful use cases for GeoTime as follows. It can be used as a visual history tool for the ‘information consumer’ to get an instant view of activity at any time/space coordinate. The ‘information producer’ can also use it as a criminal analysis tool, for example, a police officer can use it as an interactive log of events gathered during the course of long-term investigations. The system was evaluated with four subject matter experts and the results showed that the following information could be easily extracted from the system: significant events happened in a certain area in a specific time, people involved, history of involved people, people connections, activity hot spots, and similar events occurrence.

Heer et al. [32] built a software framework for creating dynamic visualizations of both structured and unstructured data named “prefuse”. The motivation behind prefuse is to create an abstract information visualization framework that fits a wide range of data applications. Prefuse provides a set of building blocks for constructing customized visualizations. It includes a multitude of layout algorithms, navigation and interaction techniques, integrated search, and more. Applications with prefuse interactive visualizations maintained real-time interaction and animation rates with thousands of on-screen items and over a million data elements. To evaluate prefuse, the study run an observatory evaluation on eight participants of varying background and expertise while they are using the prefuse API. Results showed that all participants succeeded to build visualization with prefuse.

Keim [31] proposed a classification of information visualization and visual data mining techniques based on the data type, visualization technique and the interaction and distortion technique. Keim classified data types into the following categories: one-dimensional like in the case of temporal data, two-dimensional like x-y plots, multi-dimensional like relational database tables, textual data and hypertext, hierarchies and graphs, and finally algorithms and software. In addition to standard graphing techniques like bar charts, line graphs, etc., Keim suggested more advanced graph types suitable to complex data type as follows. Geometrically transformed display technique aims at finding suitable transformations of multidimensional data sets. Iconic display technique maps values of a multi-dimensional data item to the features of an icon. Dense pixel technique maps each dimension to a colored pixel and group the pixels belonging to each dimension together. Stacked display technique is used to present data partitioned hierarchically. Keim also introduced several interaction and distortion techniques like dynamic projections, interactive zooming, interactive distortion, and interactive linking and brushing.

Plaisant [33] presented a summary of current information visualization evaluation practices and a review of challenges in this field. As for the current information visualization evaluate practices, Plaisant identified four common thematic areas of evaluation: (1) Controlled experiments comparing design elements, (2) Usability evaluation of a tool. (3) Controlled experiments comparing two or more tools, (4) Case studies of tools in realistic settings. Matching tools with users, tasks and real problems is an important challenge for information visualization. Additionally, characteristics of information visualization makes it hard to evaluate, and thus user testing needs improvement. Another challenge to study is making visualization tools accessible to diverse users irrespective of their backgrounds. Yi et al. [34] studied the role of interaction in information visualization. His study has two main contributions. Firstly, he calls for the importance of interaction in information visualization research and highlight its associated complexity. Secondly, Yi proposed seven general categories of interaction techniques widely used in infovis as follows: select, explore, reconfigure, encode, abstract/elaborate, filter, and connect. The seven categories are ordered according to the typical user progressive thinking while interacting with an interactive information visualization application. While the categories look holistic, they are not collectively exhaustive. Some complex technique could be hard to classify into one of the seven proposed categories.

Gleicher et al [35] proposed a general taxonomy of visual designs for comparison that works with all application domains and data types. The taxonomy works by classifying the design space based on three general categories, based on how the relationships between the related parts of different objects are encoded. The three categories are juxtaposition (showing different objects separately), superposition (overlaying objects in the same space) and explicit encoding of relationships. The benefit of this taxonomy is that it is independent on the design or data type. It generally works for all designs although some drawbacks exist. The study also surveyed a multitude of representative systems from the information visualization literature related to comparison. The survey goal is to show the diversity of application domains and data types. The survey also showed that all designs are built from the basic building blocks of juxtaposition, superposition, and explicit encodings.

## 2.5. Technologies for 3D modeling of buildings

### *2.5.1. BIM, IFC and BIM server*

The US National Building Information Model Standard Project Committee defined BIM as: "a digital representation of physical and functional characteristics of a facility". Beyond geometry, BIM also provides additional information like the spatial relationships, light analysis, geographic information, and quantities and properties of building components. Autodesk characterized BIM with three core characteristics [42]:

(1) They create and operate on digital databases for collaboration.
(2) They manage change throughout those databases so that a change to any part of the database is coordinated in all other parts.
(3) They capture and preserve information for reuse by additional industry-specific applications.

BIM models usage has become popular in 2002 when Autodesk started contributing to it and published a white paper about it. BIM models are made up of smart models that when updated maintain the state of the BIM file up to date throughout the design no matter who is working with it. BIM has important applications in construction management, facility operation, land administration, and more.

BIM server is an open-source software used to start a server that serve BIM files. BIM server works with IFC data as IFC is the most widely used and most mature open standard for BIM models. Typically, IFC data are uploaded to the server, usually in the form of IFC files. The software analyses the data in the files and places them in an underlying database. BIM server has many features for working with BIM files. Essentially, it enables a project to be divided into subprojects. Each part of the model can then have its own project with its own users and authorizations. All underlying subprojects are always neatly converged into the master project.

### *2.5.2. Revit*

Revit [38] is a software by Autodesk for building information modeling (BIM). Beyond modeling, Revit allows users to annotate the model with 2D drafting elements, and access building information from the building model's database. Revit is equipped with tools to help planning and tracking different building's lifecycle stages. Revit can be used as a very powerful collaboration tool between different parties in the project. Each party in the project development lifecycle approach the project from a different perspective. Revit contains a big number of pre-made solid objects including walls, windows, furniture and many more. As for rendering, Revit comes with a basic rendering engine to make a more realistic image of what is otherwise a very diagrammatic model. Cloud-based rendering is also supported by Revit via an experimental plug-in named Project Neon. Could-based rendering allows users to render their models through their Autodesk accounts instead of locally through their own computers. Additionally Revit can be linked to 3ds Max to benefit from the advanced rendering capabilities that 3DS Max provides.

### *2.5.3. 3ds Max*

3ds Max [41] is a professional 3D computer graphics program for making 3D models, animations, games, videos, and images. 3ds Max provides several modeling techniques that make it possible to model any kind of 3D object. Those modeling techniques are polygon models, NURBS and surface tools. Moreover, the user can edit the model on the sub-object level, which allows editing object's polygons, faces, edges, and vertices. Additional modeling tools exists to further help the modeling process. This includes Boolean operations to merge object, subdivision, extrude, lath, and many more. As for rendering, 3ds Max comes with two rendering engines: scanline and ART Renderer. Each renderer uses a specific algorithm for image generation and incurs different measures for output quality and rendering time. It is also possible to import and use a third party rendering engine in 3dx max. There is a multitude of advanced third party renderers commonly used with 3ds Max including mental ray, RenderMan, V-Ray, and more. Additionally, 3ds Max is equipped with a multitude of powerful specialized tools including: MaxScript, character studio, scene explorer, animations, skinning, Inverse kinematics (IK), and many more.

### *2.5.4. three.js*

Three.js is an open-source cross-browser powerful library for creating 3D graphics for the web. It uses JavaScript as its programming language and renders its output to a WebGL canvas. It also supports fallback to an HTML5 canvas. Three.js provides good set of features that are usually enough to develop most of the 3D simulations on the web. High-level libraries such as Three.js make it possible to author complex 3D computer simulations running in a web browser while saving the effort required to program on the low-level computer graphics and shaders level. Three.js allows creating all the basic geometric objects like plane, cube, sphere, torus, 3D text and more. It supports two camera types: perspective and orthographic and many controllers: trackball, FPS, path and more. It also supports the common light types like ambient, direction, point and spot lights plus shadows. For advanced usages, three.js provides access to full OpenGL Shading Language (GLSL) capabilities like lens flare, depth pass and extensive post-processing library. Today three.js library is widely used for a wide variety of applications, visualization, simulations and games on the web. The community for three.js is large and it is continuously being developed with new features. Recently, an online visual editor [58] has be developed for three.js to allow creating and manipulating 3D scenes visually. The three.js editor allows creating primitive 3D object, materials, lights, and cameras. The editor exports the created scene in JSON format, which can then be imported to three.js easily.

### *2.5.5. Unity*

Unity is a professional cross-platform engine for developing interactive and 3D application. It is mainly used for developing high quality video games for PC, consoles, and mobile devices, but it is also suitable for creating any sort of 3D simulations, visualizations, etc. Unity targets the following graphics APIs: Direct3D and Vulkan on Windows and Xbox 360, OpenGL on Mac, Linux, and Windows, OpenGL ES on Android and iOS, and other proprietary APIs on video game consoles. One of the greatest advantages of using Unity is its ability to target multiple platforms. Supported platforms include Windows, Windows Phone 8, Xbox, Android, iOS, macOS, Apple TV, BlackBerry, Linux, Nintendo 3DS line, PlayStation 4, PlayStation Vita, Unity Web Player, Wii, Wii U, and Nintendo Switch. Today unity is one of the most widely used engine for games production. Statistics show that in Q3 2016 alone, five billion downloads for unity developed games has been made [59]. Unity is geared with a full list of features for physics, animation, optimization, audio, scripting, and many more [60].

# 3. DESIGN AND IMPLEMENTATION

This chapter describes all the details of the design and implementation of the two smart building visualization systems. This includes the system requirements, implementation process, system design, justification of the design decisions, system architecture, and the implemented visualizations UI elements. The two implemented applications are similar in functionality as both visualize sensor data using various techniques in a 3D virtual environment. The next sub-sections explain in detail the system design and implementation details.

## 3.1. Design and implementation process

We followed the waterfall model in the design and implementation of both projects independently. This progressive process provided a systematic pipeline for our system, which guaranteed quality and minimized the time needed to fix bugs. The process consists of the flowing phases (Figure 6):

1. **Requirements definition:** In this process, we define and establish the system requirements based on the system service, constraints and goals (as discussed in section 3.1).
2. **System design:** The system design process establishes the system architecture by partitioning the system requirements into functional software sub systems and finding the relationship among them.
3. **Implementation and testing of a new feature:** During this process, software implementation of a system feature (one type of visualization) takes place. Newly implemented feature is tested thoroughly to make sure it is functioning correctly without any bugs.
4. **Integration and testing:** After one feature is developed and tested, it is integrated into the current system state. Next, the whole system is tested to make sure the newly added feature did not cause any system failures. This guarantees the software is functioning well up to the current state.
5. **Maintenance:** This process starts once the system is ready for use. At this process, the system is tested as a fully functional application. Should there be any remaining bugs, features edits, etc., the software is maintained according to the required modification.

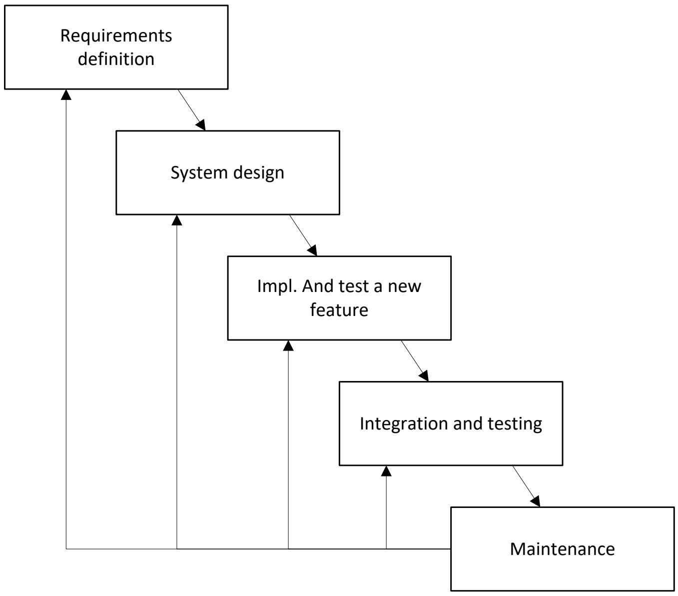


Figure 6. Design and implementation process

### 3.2. Requirements

The system aims to provide user friendly, easy to use, visualizations for sensor data. The following requirements were set for the system:

- **Scalability:** Since the building model can be large, the system must be able to scales up when new sensor nodes are added without degrading performance. To meet this requirement, the visualization application must be able to load large amount of sensor data without problems.
- **System performance:** The system must run smoothly on an average computer. Our system uses particle systems which could be computationally expensive, to simulate fire and ice. To meet this requirement, the system must perform well in worst case, which is when a particle system simulation is running plus a point visualization.
- **Maintainability:** The system should easily support adding new features, like new visualization types. Additionally, the system must support changing the sensors data source with ease. For example if the sensors’ cloud server, the data structure, protocol, etc. changed, this should not require a lot of code rewrite to get the application to run.

### 3.3. Design

We explain in this section the design details of the two implemented projects. We start by explaining the common features of the two projects then highlight the differences between them before elaborating of the design of each project. Both of the two projects serve as systems for visualizing sensor data captured from a sensor network deployed in a smart building. The visualizations are implemented in a 3D virtual environment that is accurately designed to match the same dimensions of the smart building it represents. Both projects were designed to run in a web browser. The 3D user interface manipulation is similar for the two projects as well. We implemented the view manipulation as an orbit camera in which users can manipulate the 3D UI using the mouse by clicking and dragging to change the viewing angle of the virtual environment so that they can view a sensor visualization at a desired area of the model. In addition, the model can be zoomed in and out using the mouse scroll wheel. While both projects provide the same fundamental functionality, they have significant differences in many aspects as highlighted in Table 1.

Table 1. Details of the two approaches dubbed as project A and B

| | **Project A** | **Project B** |
|---|---|---|
| **IDE** | Three.js | Unity |
| **data format** | JSON | XML |
| **programming language** | JavaScript | C# |
| **visualization types** | point, wall | point, floor |
| **data source** | python webserver | SOAP web service |
| **model type** | BIM | Mesh |
| **building type** | smart apartment | 4 floored hospice |
| **sensor data** | temperature, humidity, air quality, air pressure | temperature |

The literature review revealed that the choice of colors plays an important role in making data visualization easier to understand [22]. As an example, the work highlighted earlier by Fan et al. [20] used colors to partition different parts of a map to give a meaning for each map partition. Literature also suggests that chosen colors have linear separation and of different color categories. Based on that, we used colors for our temperature visualization to make it easy for users to visually distinguish between temperature values above and below zero. We used red to represent temperature values above zero and blue to represent values below zero. Additionally, we kept the user interface as simple as possible as [29] suggests that users tend to perform better with simpler user interfaces. The use of textual information for data visualization has been widely seen in the literature [23, 24, 28, 30]. We used textual information to display the values of the temperature visualizations. Yet we allowed the users to toggle it on/off.

Next, we look at the specifics of each project. Starting by project A, a sensor network was installed in a research smart apartment located inside VTT research center in Oulu, Finland. The apartment consists of three rooms, a kitchen, a bathroom, a sauna, and a balcony as shown in Figure 7. The project's goal is to create a system for visualizing the sensors data in the smart apartment in a web interface. The model for the smart apartment already existed and served to us as a BIM model via a BIM server. For the sake of our project, we did not need anything specific from the BIM data except for the geometry data. We were provided with the BIM server details and we were able to establish a connection with the server and load the BIM model with ease. For project A, we used three.js to program the application. For project A, we categorized the visualizations into two groups: point visualizations, where the visualizations are laid at the locations of the sensors, while showing the value and/or a visual representing the sensor value at that point. The second visualization group is wall visualizations, where the visualization is displayed as a 2D banner positioned on a room wall inside the apartment. The design allows users to switch between wall sensors with ease using a control panel on the right (more on that in the UI section). Users can also mix point and wall visualizations at the same time.

Figure 7. Project A: 3D model of VTT's smart apartment

In Project B, a sensor network is installed in a big, multistory hospice building for elderly care located in the city of Oulu, Finland. The building consists of four floors, each of which contains a multitude of rooms and passages. The first task was to create the 3D mesh model of the hospice building. We were provided with architectural floor plan drawings of the building. We used 3ds max to model and the building and add the materials to it. We chose to use 3ds Max over other 3D modeling software for three main reasons: it is a powerful professional 3D computer graphics program, we have experience using it, and as students, we get a free 3 years license to use the software with all its features enabled. For project B we decided to use Unity as the development environment. We would import the 3D hospice model to Unity and start coding the application logic. Since the hospice building has four floors, we constrained the visualizations to only one floor at a time to keep the view neat and clear. The user can switch between the floors using an intuitive vertical slider that has four snap-able steps matching the total number of building floors. When the slider is at the bottom most state, only the ground floor is visible. Upon moving the slider up, the corresponding floor will appear. This gives the user a natural feel of visualizing the floors of the building. Figure 8 shows the four floors of the building corresponding to each state of the floors slider. The floors slider is shown on the top right of the screen.

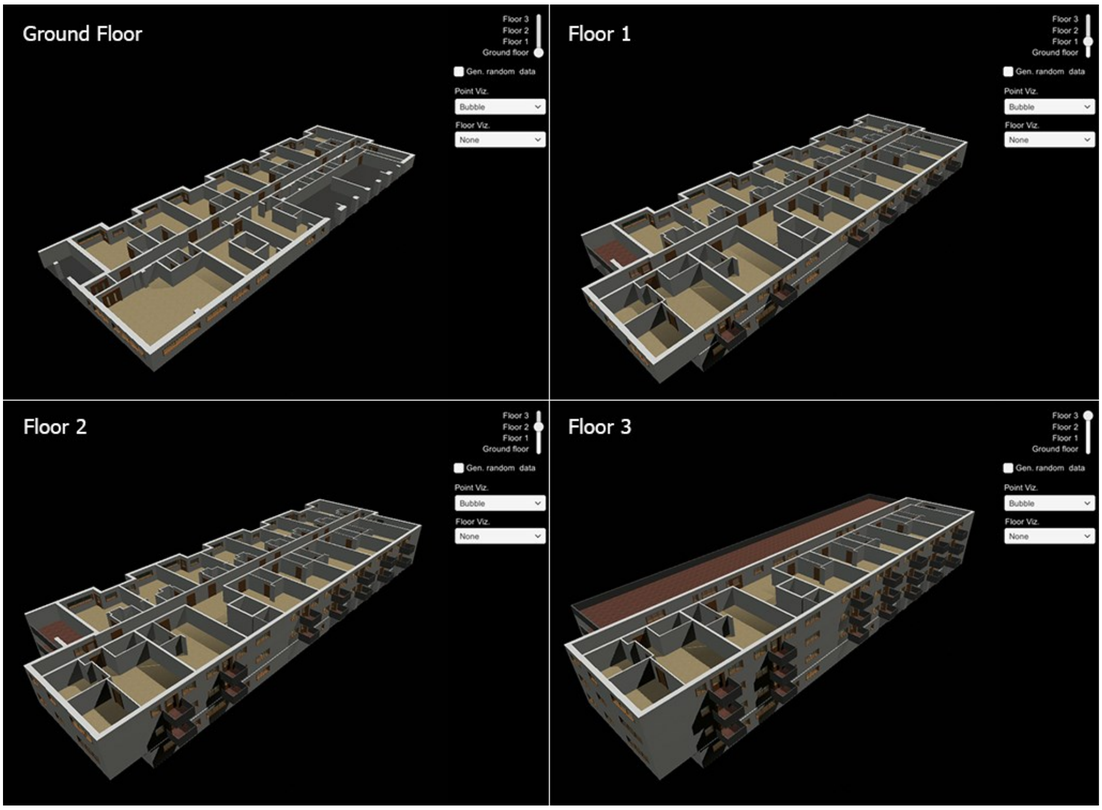


Figure 8. Project B: the four floors of the hospice building

In project B, we categorized the temperature visualizations into two groups: point visualization, where each sensor is represented with a matching visualization element in the 3D environment. Typically, the point visualization displays the value of the temperature at the sensor location point in a user-friendly manner. The second category is floor visualization, where for each sensor we visualize the value of the sensor as an effect on an area around the sensor location on the floor. Typically, showing a simulation of the area of effect of the sensor. We used red and blue colors to allow the user to distinguish between the temperature values below and above zero with ease (more on design justification in next section). For each visualization category, we created several visualization options. The user can change the visualization of each group. He can also mix one visualization from each group together. However, it is not possible to mix two visualizations from the same category, as in this case the two visualizations will overlap and thus obstructing the temperature visualization altogether. The implementation sections presents detailed information about the look of the implemented visualizations.

### 3.4. System architecture

We show in this section the system architecture for the two projects. Based on the system requirements, we planned and constructed the system architecture. The system architecture includes the main building components of the system in the real environment, the cloud and in the client visualization application. For project A, Figure 9 shows the various components of the system architecture.

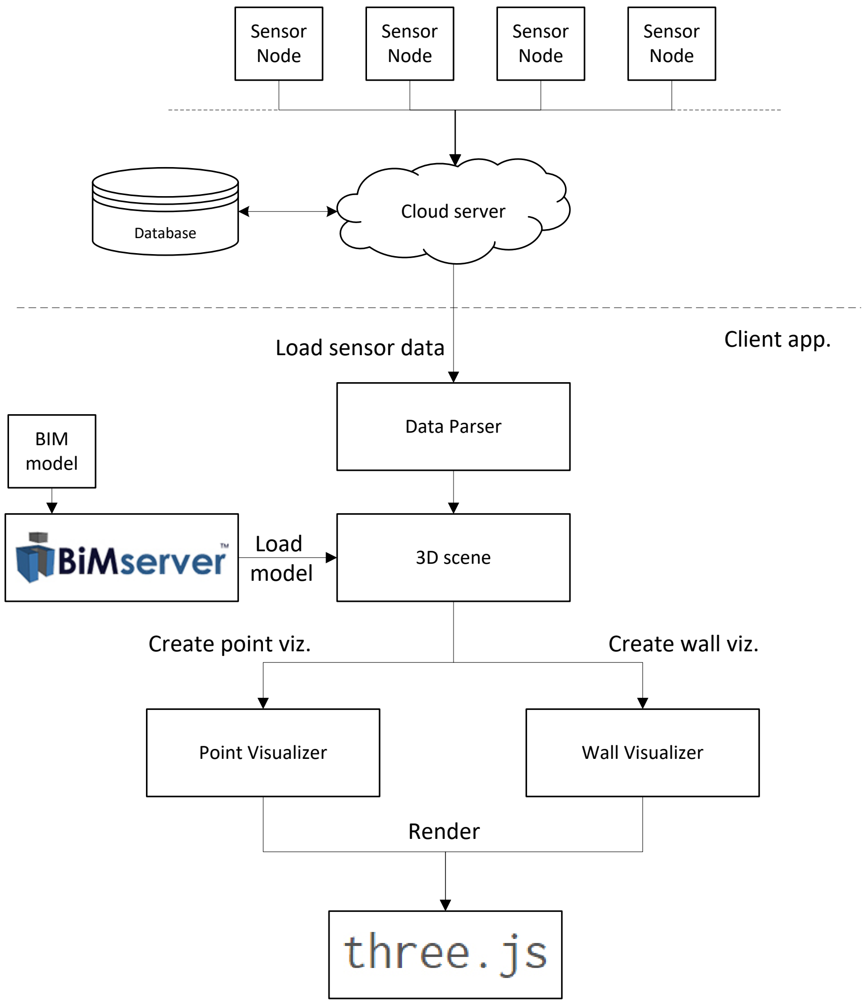


Figure 9. Project A: system architecture

Project A's system architecture comprises of following components:

- **Sensor Node:** 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. The sensors were developed by VTT and provided to us by VTT and iProtoxi.
- **Cloud server:** The cloud server was deployed by VTT on ThingSpeak IoT platform. The server acts as the centralized cloud that connects between the real environments and the visualization application. The server receives the sensor data from the sensor nodes and persists them to the database. The server exposes an API for the visualization application to use to request the sensors' data in JSON format. The visualization application requests the sensors's 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.
- **Database:** this the data storage for the sensors's data collected by the sensor nodes network. The database was developed by VTT. 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.
- **BIM Model:** this is the 3D model of the smart apartments in BIM format. The model is served as a BIM file. BIM format was explained in the previous section. BIM files are served from BIM servers.
- **BIM server:** BIM server serves BIM models in the form of BIM files. The client visualization application connects to the BIM server to request the smart apartment BIM model.
- **Point visualizer:** Once the sensors' data are loaded, it is fed to the point visualizer to create the point visualizations. Point visualizations show the value of the environmental parameters at the location of the sensor. For project A, we have implemented five point visualization types. Each point visualization type has its unique design, size, and color. One of them has slight animation as well. More details on each type and its look will be provided in the implementation section.
- **Wall visualizer:** This component works is a similar manner to the point visualizer, but instead of creating point visualizations, it creates visualizations as banners laid on the building's internal walls. A wall visualization can also be a 3D model, we have implemented one of the wall visualizations as a 3D thermometer to be laid on a wall. Each wall banner shows the values of the sensor data in a certain color, font, size, and design. We have implemented five wall visualization types in project A. More details on each type and how it look will be provided in the implementation section.
- **three.js**: three.js is an open-source JavaScript library used to create computer graphics simulation running in a WebGL canvas in a web browser. It contains a wide set of features and rendering capabilities suitable for creating 3D graphics for the web. We programmed project A using JavaScript programming language.

For project B, we followed a similar system architecture to what has been designed for project A. The main difference is that project B did not request the model from a BIM server. Instead, the model was loaded from a mesh model file. The system architecture of project B is shown in Figure 10.

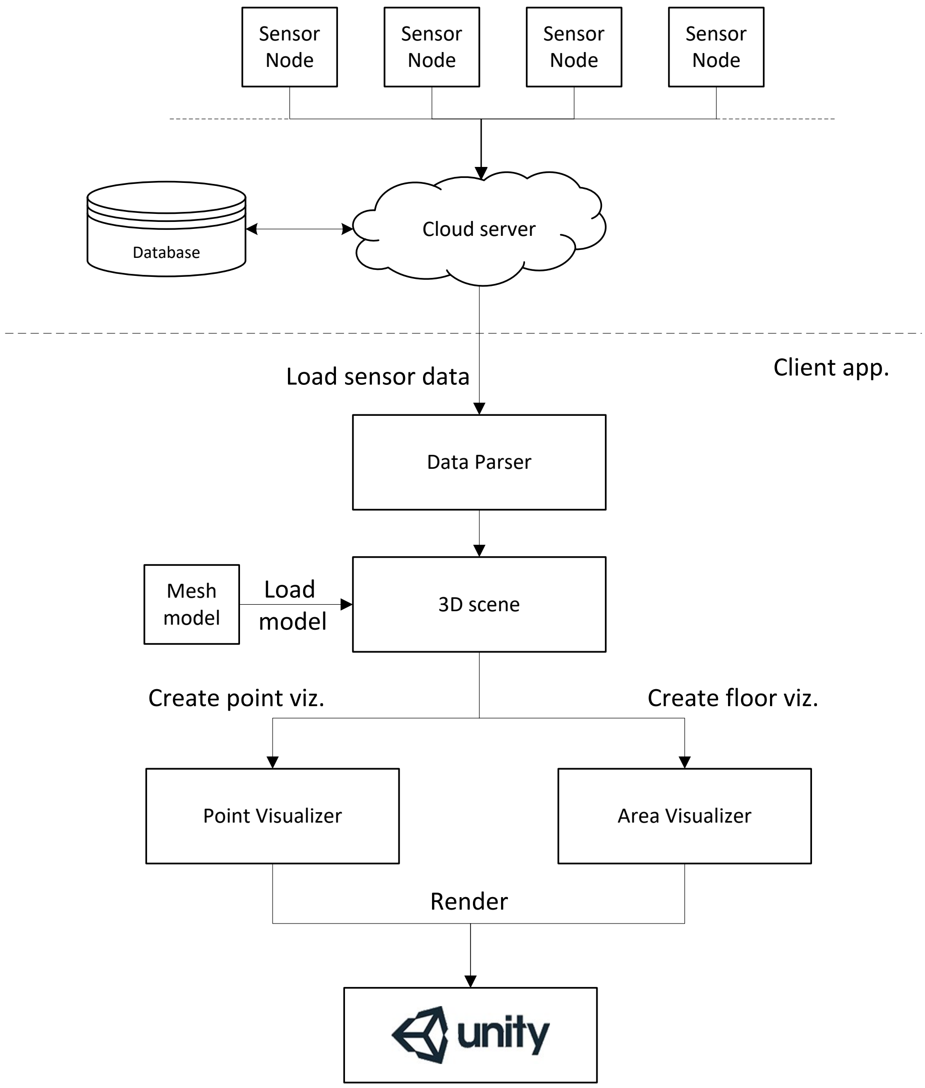


Figure 10. Project B: system architecture

In the following sub-section, we explain in detail each component in the system architecture:

- **Sensor Node:** Each sensor node is a physical sensor deployed at a certain location within the hospice building. A sensor node is responsible for capturing the temperature value within its location and submitting it to the cloud server. The sensor nodes were deployed at the building construction stage.
- **Cloud server:** This is the centralized server that connects between the real environments and the visualization application. The server is developed by Fidelix [57]. The server receives the temperature readings from the sensor nodes and persists them to the database. The server exposes a SOAP-based web service running over HTTP for the visualization application to use to request temperature values. The visualization application requests the temperature values 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. 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 sensors

visualizations. SOAP protocol uses XML based data format to serialize and transmit the data. XML is a widely used data transfer format on the world wide web (WWW), for storing, serializing and transferring data around the internet.

- **Database:** this is the data storage for the temperature values collected by the network of sensor nodes. 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.
- **3D scene:** this component is where the 3D environment is created. 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.
- **Point visualizer:** Once the data are loaded, it is fed to the point visualizer to create the point visualizations. Point visualizations show the value of the temperature at the location of the sensor in a certain design and color. We have implemented four point visualization types: bubble, sphere, thermometer, and icons. More details on each type and it look will be provided in the next section.
- **Floor visualizer:** This component works in a similar manner to the point visualizer, but instead of creating point visualizations, it creates floor visualizations. A floor visualization shows the color or effect of the temperature value on the floor instead of showing the temperature textual value. We have implemented three floor visualization types: solid colors, heat map, and an experimental fire and snow visualization. More details on each type and it look will be provided in the next section.
- **Unity:** unity is a cross-platform game engine providing a rich set of features and rendering capabilities. We used unity to build and render project B. Unity allows publishing the project to multiple platforms with ease. We used C# programming language to code the application.

### 3.5. Visualizations

We present in this section the implemented visualizations in both projects. In Project A, we implemented five point visualizations, five wall visualizations together with one table view and one visualization based on area coloring. Project B implements four point visualizations and three floor visualizations including a fire and snow simulation. All point visualizations in both projects are made always facing the camera, which make them easy for the user to view from any angle. In project B was included an option to generate random data for rooms that does not have real data coming from the server. This makes it easy to view the visualizations even when there is no data available at the server. Figure 11 shows the view of the smart apartment when only the wall visualizations are active. Figure 12 shows the apartment when both the wall and point visualizations are active together.


Figure 11. Project A: wall visualizations


Figure 12. Project A: wall and point visualization

The five implemented point visualization are labeled: sprite, sprite 2, bars, arrows, and bubbles. Figure 13 shows an up-close look at the various point visualizations implemented for project A. The sprite and sprite 2 visualizations (Figure 13 a and b) show the value of sensor data as a yellow text that is always facing the camera. The difference between them is that Sprite visualization shows one sensor value at a time (temperature, humidity or Co2). On the controls panel, the Sprite visualization has a radio button group that the user can use to switch the visualized sensor data type. Sprite 2 visualization on the other hand shows all the data for one sensor stacked vertically in one visualization. Additionally, it shows indicative icons beside the values. Bar visualization (Figure 13c) shows the sensor values as bars where the height and color of the bar change according to the sensor value. Each bar represents one sensor data. The arrow visualization (Figure 13 d) is a special visualization for visualizing pressure. It shows an animated arrow plus the pressure value in blue above the arrow. Finally, the bubble visualization (Figure 13 e) shows each value of the sensor data in a bubble containing the sensor data value. The color of the bubble changes according to the sensor data type and value. The bubble has a slight vertical animation that gives it a vivid look.

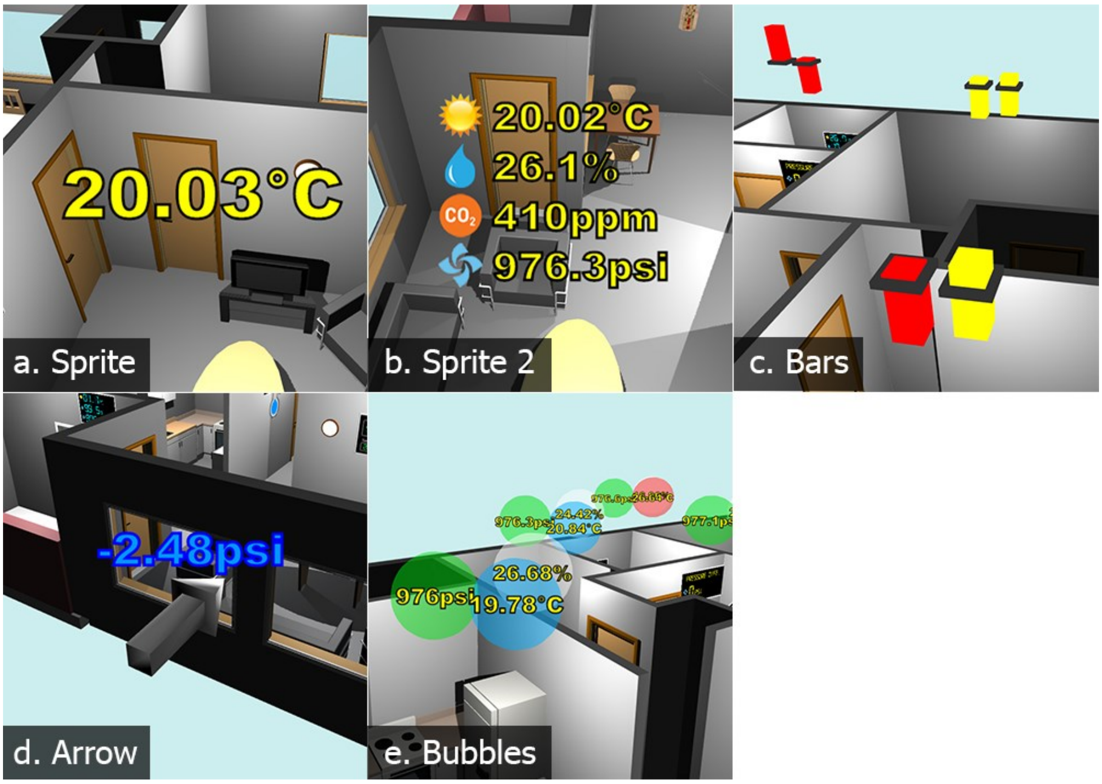


Figure 13. Project A: point visualizations

In addition to the wall and point visualizations, a table view shows all the sensors and their data in a raw table view as shows in Figure 14.

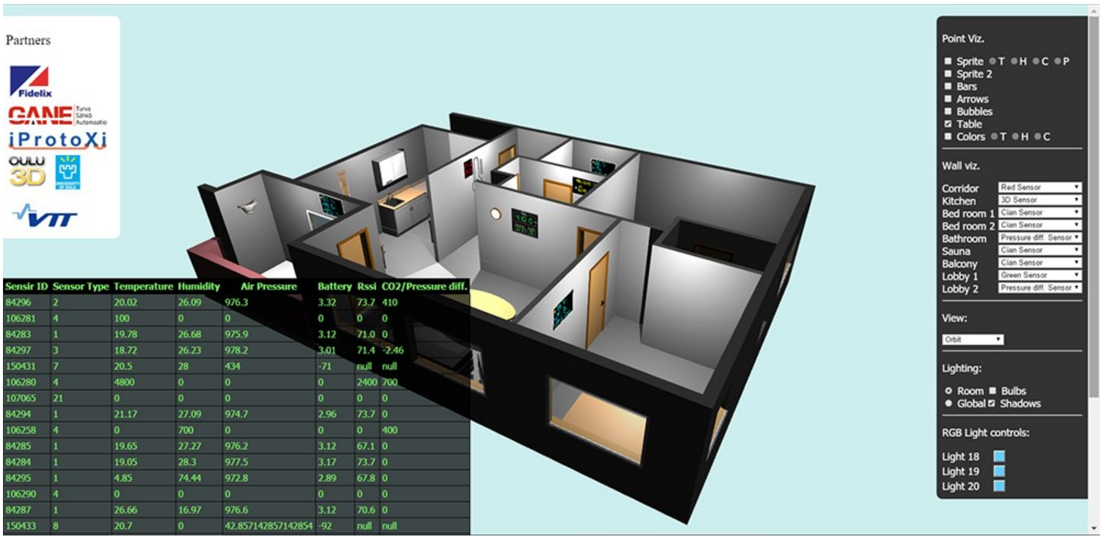


Figure 14. Project A: table view

We have implemented some additional features in project A that are not available in project B. As for camera controls we have added preset camera position and rotation values in a drop down list that allows the user to switch the view to certain position and angle so that he can move directly to a certain room. In addition, we have implemented a first person view so that the user can move inside the apartment using the standard first person shooter (FPS) game controls (arrows or A, S, D, W keys to move + mouse to look around). As for lighting, we created two lighting modes: global and room. The global lighting is composed of two three.js directional lights positioned so that they light the entire scene. The result is a basic strong light that lights the apartment equally. The room lighting mode is a more advanced light that provides a soft lighting in each room. For room lighting, one point light was created for each room. The light radius and attenuation of the point lights were set so that the light affects the area of the area it is lighting. There is also a checkbox to toggle the visibility of the room's light source (bulb) on and off. Room lighting casts soft shadows. We added also a checkbox to toggle the room lightings' shadows on and off. The scene looks much better with shadows but switching off shadows could come in handy when working on a computer with low rendering capabilities. The last additional feature in project A is an experimental light controls that allows the user to change the color of three RGB color-lights installed in the real apartment. The user would be able to click a color picker and choose a color and this color would be reflects to the real light in real-time. This feature is still experimental and additional work is required to finalize and test it. Lastly, project A apartment model contains some furniture models with materials applied while project B doesn't.

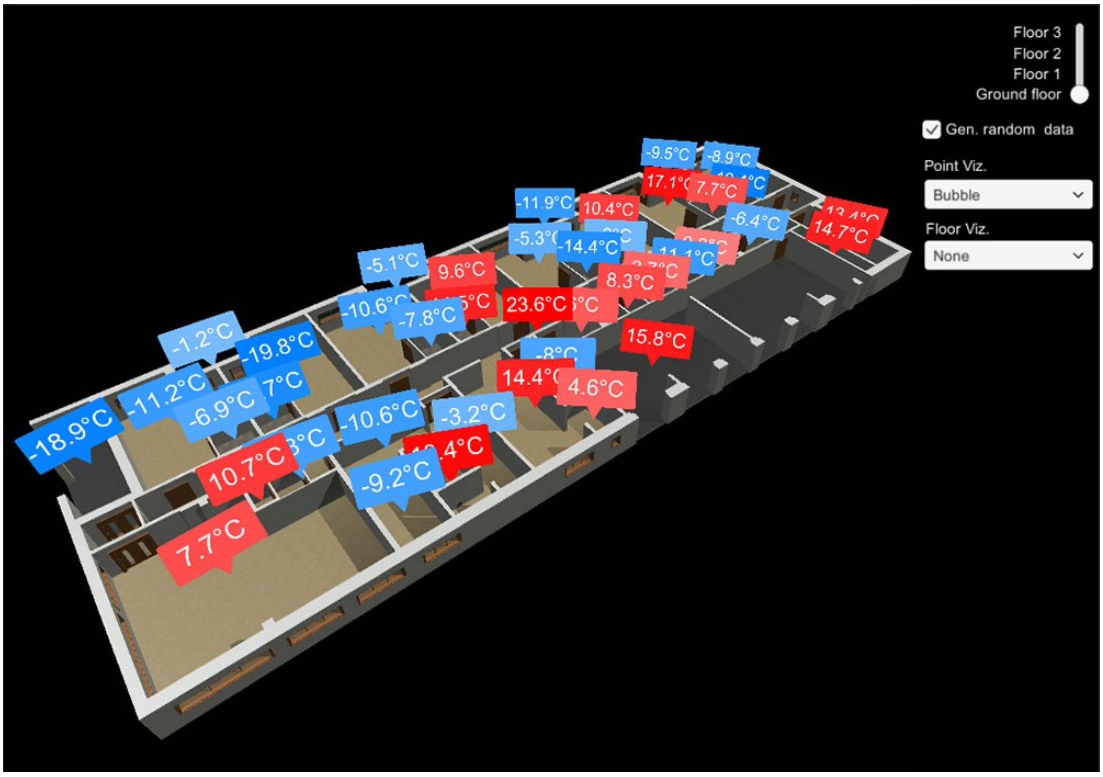


Figure 15. Project B: bubbles point visualization

Next, we present the implemented visualizations in project B. We implemented four point visualizations and three floor visualizations in project B. Project B hospice model container 4 floors. For the sake of simplicity, we will present our implemented visualization in the ground floor (Figures 15 and 16). Yet, toggling the model to another floor will show the currently selected visualizations applied for new floor. As highlighted before, point and floor visualizations can be mixed together. Figure 15 shows a view from the ground floor of the hospice building with a point visualization only. Figure 16 shows the ground floor when a point visualization plus a floor visualization are toggled on at the same time.

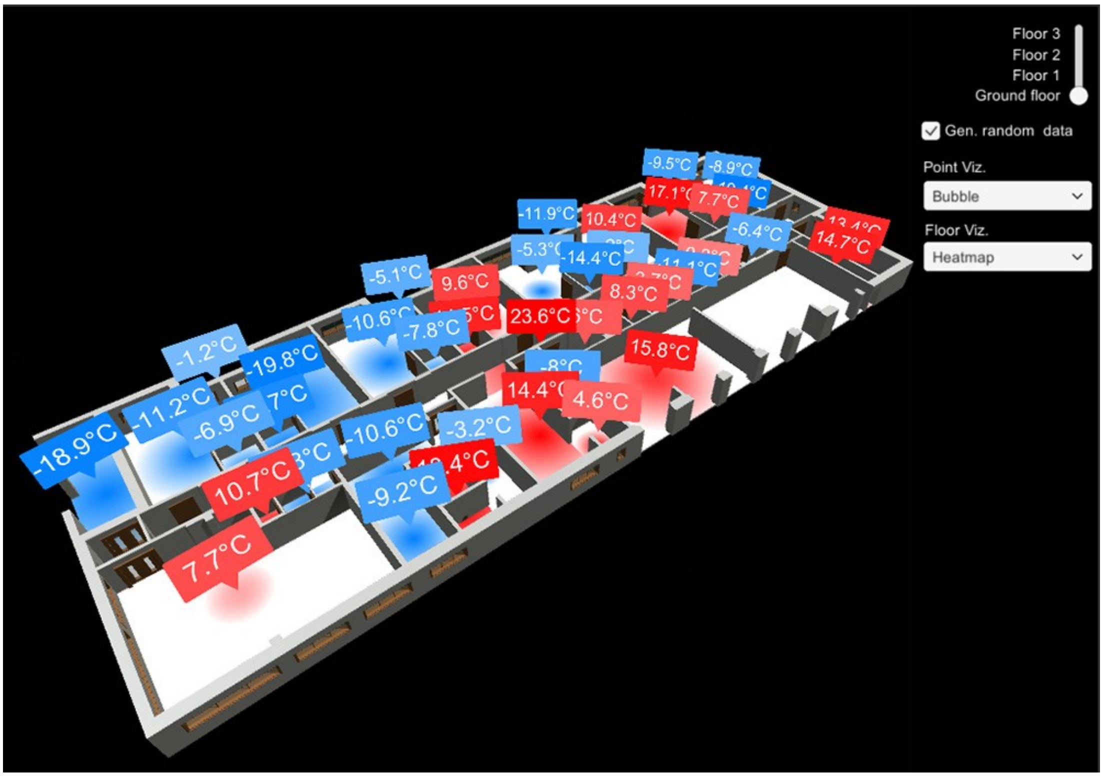


Figure 16. Project B: bubbles point visualization with heat map floor visualization

The intensity of the visualizations color and the radius of the heap map is proportional to temperature value as shown in Figures 15 and 16. Also it worth noting that the heat map visualization looks good with almost all point visualizations as it acts as a continuation of the visualization on the floor. In Figure 17, the four implemented point visualizations are displayed.

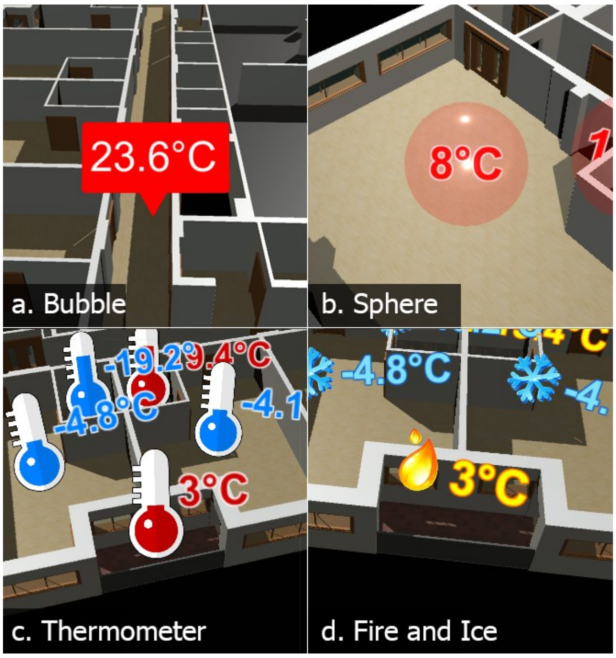


Figure 17. Project B: point visualizations

The first visualization is labeled "Bubble" (Figure 17a). It shows the temperature value in a message bubble. The color of the bubble's background is a shade of red or blue depending on the temperature value. The higher the temperature value, the more saturated the color. Figure 18 shows all the possible color shades corresponding to different temperature values. If the temperature value is bigger than 20 or less than -20 the color is completely saturated red or blue.


Figure 18. Bubbles point visualization color shades

The second point visualization is labeled "Sphere" (Figure 17b). It shows the temperature as a number inside a translucent specular sphere. The color of the text as well as the sphere is either red, if the temperature is above zero or blue if temperature is below zero. The text also has a thin white outline that makes it clearer to read. The third point visualization is labeled "Thermometer" (Figure 17c). It shows a graphical icon of a thermometer where the height of the inner bar reflects the temperature value. The color of the thermometer is either blue if the temperature value is below zero or red if the temperature is above zero. The temperature value is also shown beside the thermometer. The user interface has a checkbox to toggle the temperature value beside the thermometer on and off. We added the ability to switch off the text because when there are many visualizations in the scene, the text values interfere together making it hard to read individual sensor values. The last point visualization implemented in project 2 is "Fire and Ice" (Figure 17d). It shows the value of the temperature in either a yellow color with orange outline if the temperature is above zero or cyan with blue outline if the temperature value is below zero. Additionally, either a fire or a snowflake icon is shown next to the temperature value depending on the temperature value. The colors of the text were chosen to match the colors of the fire and snowflake icons. This point visualization looks good either alone or in combination with the fire and snow floor visualization. Next, we look at the three implemented floor visualizations in project B (Figure 19).

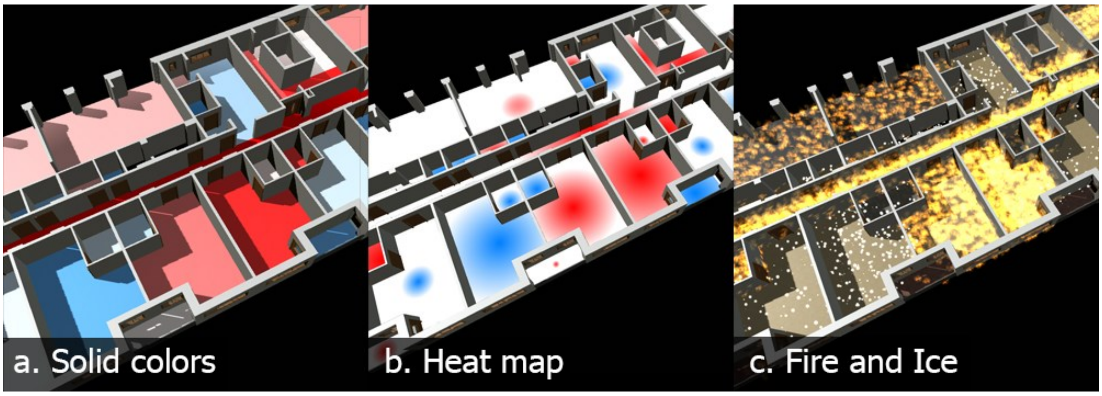


Figure 19. Project B: floor visualizations

The first visualization is labeled "solid colors" (Figure 19.a). It shows the floor of each area (typically a room) in either red or blue shade. The color intensity is proportional to the temperature value so the more reddish the color, the more hot the temperature. The more bluish, the more cold. If the temperature value is close to zero, the floor's color would be close to white. Figure 20 shows the possible color shades of red and blue versus the temperature values for project B's solid colors floor visualization.


Figure 20. Solid colors visualization color shades

The heat map visualization shown in Figure 19.b works in a similar way. It shows a circle with fading edge on the floor. The center of the circle represents the point where the sensor is located. The color of the circle can be either red if the temperature value is above zero or blue if it is below zero. The radius of the visualization circle is proportional to the temperature value. The higher the temperature, the bigger the radius. To implement the heap map visualization we created custom pixel and fragment shaders that can be fed with the sensors' locations and temperature values and generate a material with the discussed heat map properties. Additionally, we split the floor into separate meshes so that each room has its own floor piece. This is required technically so that the heat map shader material only affects the floor part it is meant to affect and do not extend to adjacent floors. The heat map visualization looks good in combination with any of the aforementioned point visualizations. The last implemented floor visualization is the fire and ice (Figure 19.c). In this visualization, fire or snow falling simulations are simulated in each floor area depending on the temperature value. For temperature values above zero, a fire particle system is started. For values below zero a snow falling system is started. The intensity of the fire and snow particles is proportional with the temperature value. So the higher the temperature, the more strong is the fire. The lower the temperature, the higher the snow precipitation rate. This visualization looks good in combination with the fire and ice point visualization.

### 3.6. User interface

We present in this section the user interface (UI) of the two implemented projects. In both projects, the user interface is implemented as a panel docked to the right of the screen. We start by demonstrating the user interface for project A. The user interface of project A is shown in Figures 11, 12 and 14. To the left of the screen is a white panel containing the list of the project partners. The partners for project A are Fidelix, Gane, iProtoXi, Oulu3D, and VTT research center. At the bottom of the screen is another white panel containing a descriptive message about the project and its funding source. To the right of the screen is the main UI panel, which contains all the controls to toggle the visualizations on and off. The panel is divided into five sections stacked vertically. The first section contains the point visualizations. Each visualization has a checkbox next to it where the user can toggle it on and off. Depending on the visualization type, additional controls may exist, for example, Sprite point viz. has a radio button group allowing the user to choose which sensor data to show (temperature, humidity, co2 or pressure). The second section is for wall visualizations. Wall visualizations cannot be switched off. For each room in the apartment that is a drop down list showing the list of the wall visualizations. The user can change the wall visualization for a specific room using the drop down list for that room. The third section in the panel contains a drop down list containing the implemented preset camera controls plus the first person mode. The fourth section contains lighting controls including a radio button group to choose between the global and room lighting modes, a checkbox to toggle the room bubbles on and off, and a checkbox to toggle shadows on and off. The last section contains the experimental RGB light controls. This section contains three light labels and color pickers that the user can use to control the colors of three real lights deployed in the real apartment as discussed in the implementation section.

The user interface for project B is simpler than project A's as shown in figures 15 and 16. It contains only the controls panel to the right where the user can choose the visualizations and the floor. The first section of the panel contains a vertical slider that the user can use to change the view to a certain floor of the building. We implemented the slider in a vertical direction with four snap-able steps so that switching floors mimics the visual change of the model floors, which give a natural feedback to the user about changing floor and hence improving the system usability. When the user change to a different floor the current visualizations are removed from the old floor and recreated to the newly chosen floor. After the floor selection section, a check box exists to allow generating random data for all the floor areas.

Current active visualizations will be updated with the randomly generated data. This is useful to be able to experiment with the application visualizations even if the sensor data is limited or not available. Next comes the two visualization sections. Each visualization type (point and floor) is represented as a drop down list that the user can click to check the available visualizations and choose one of them. Depending on the chose visualization, additional options can appear in the panel to change some aspects of the visualization. The thermometer point visualization has an additional option to toggle the visibility of the temperature text. The sphere point visualization has an additional option to toggle the sphere visibility on and off.

# 4. COMPARATIVE PERFORMANCE EVALUATION

This chapter reports the comparison of the two approaches with respect to rendering performance, model size, request time, and development effort.

## 4.1. Rendering performance

To compare the performance of both applications we ran a small experiment in which we started each project and tried all visualizations and controls. We monitored the frames per second (fps) count while doing this experiment. We ran both projects on the same computer having the following specifications: Intel Core i7 6700HQ CPU, 16GB RAM, NVIDA GeForce GTX 960M, memory 4MB GDDR5 GPU. Results showed that project A kept a steady frame rate of 59-60 fps in worst case when all visualizations were on at the same time plus shadows. The frame rate only dropped shortly to about 55 when moving the view very quickly while all the visualizations are switched on. Yet it is just a momentarily drop that did not cause any noticeable result on the user experience. Project B also kept a steady frame rate of 59 all the time in worst case when the fire and snow particles were on plus a floor visualization. The frame rate dropped shortly when switching between floors or changing the point or floor visualization type. This is expected as when switching floors or changing visualizations, the application removes the old visualizations and creates new ones. Yet the frame rate drop is very short and acceptable as it happens only during changing a setting. Therefore, we conclude that both applications run smoothly on a web browser on an average modern computer. Table 2 summarizes the performance of both projects.

Table 2. Performance of the two projects in terms of fps

| | **average fps** | **shortly dropped fps (reason)** |
|---|---|---|
| **Project A** | 59-60 | 55 (very fast navigation) |
| **Project B** | 59 | 50-55 (change visualization or floor) |

## 4.2. Model format and size

Each of the two projects has a different model format. Project A utilized a BIM model loaded from a BIM server. BIM models cover more than just geometry. It also covers spatial relationships, light analysis, geographic information, and quantities and properties of building components. For the sake of creating a 3D visualization application, we did not really need the extra information that the BIM standard provides. Additionally the BIM file size is bigger than a regular mesh model due to the extra information it holds. Also bigger file size means additional time to request and load the BIM model from the server. For those reasons, we extracted a mesh model out of the BIM file used for Project A and removed the extra information that we do not need. The BIM model is 26.7MB while the exported FBX is just 10.3MB. This gave us a 61.4% save on the request size. If the exported FBX is compressed it can even greatly reduce in size to around 390KB. To load the model to the three.js scene, we converted it to the JSON format that is supported by three.js. Project B utilized a mesh model created using 3D max. The mesh model is exported as FBX file format, which works well with unity. Project B model is a four-floor hospice building, which has much bigger geometric data compared to Project A's model. The file size of projects B model is 3.74MB before compression and 975KB after compression. Table 3 summarizes relevant details of the models used in the two projects.

Table 3. Relevant details of the models used in the two projects

| | file size (MB) | file size compressed | file format | model description |
|---|---|---|---|---|
| **Project A** | 3.72 | 974 KB | JSON | small apartment |
| **Project B** | 10.3 | 390 KB | FBX | large 4-floor hospice |

### 4.3. Request format and load time

To measure the time taken to request the sensor data from the server, we monitored the period since a request is issued until the result is returned successfully. To request the sensor data from the server in project A, we issue a GET request to the sensor webserver. We pass an authentication key as a parameter with the request. The result comes in JSON format and contains the most recent sensors data for all the sensors in the smart apartment. The returned result contains 100 records. We found from monitoring the request time that it takes between 0.1 to ~0.3 seconds. In average, it took 0.15 seconds. The size of the requested data is 17.8 KB. For project B, we issue a POST request to a SOAP web service to load the sensors data. We need to issue one request per each sensor node. The request takes as parameter the sensor's unique ID. The request took between 0.03 to 0.05 seconds to return a result. In average, it takes 0.04 seconds. While this is very small compared to project A's request time, the request only returns data for a single sensor. The response has a size of 0.42 KB. To estimate the request speed we divide the request time by the response size. This returns a load speed of 8.42 milliseconds/KB for project A's requests and 95.2 milliseconds/KB for project B. The results of the tests are summarized in the table below.

Table 4. Request and response time of the two approaches

| | request time | response size (KB) | speed (ms/KB) |
|---|---|---|---|
| **Project A** | 0.15 s | 17.8 KB | 8.42 |
| **Project B** | 0.04 s | 0.42 KB | 95.2 |

### 4.4. Development SDK

We discuss and compare in this section the development environments for both projects. Project A was developed using Three.js JavaScript library while project B was developed using Unity IDE. When comparing the two SDKs, the comparison is not balanced. Unity is a full-fledged commercial software with rich set of high-end tools and features tailored for creating 3D applications and game. On the other hand, three.js is an open-source library for creating 3D graphics using JavaScript for the web. Three.js provides a good set of features that are enough most of the time to create 3D graphics simulations yet Unity is much more powerful. We start by taking a closer look at Unity IDE. Unity is a complete game engine for create 2D and 3D games and application. It contains a rich visual editor allowing manipulating the 3D scene visually with ease. Moreover, the visual editor serves as a level designer. Unity also provides tools for networking/multiuser support, advanced particle system, physics system, post effects, and more. Furthermore, it has an easy to use user interface and utilizes an intuitive component system. Due to its powerful capabilities, unity has a productive workflow. Unity provides three scripting languages for programming: C#, JavaScript and boo. Boo programming language has been deprecated since Unity version 5 and is no longer supported. The majority of unity community uses C# with a ratio of 80.4% compared to 18.9% of JavaScript users. Unity provides two license options: commercial and free. The free license provides all the engine features but the builds have a unity watermark splash screen. In addition, the free license come with a light editor theme while the commercial editor comes with a dark theme. Unity can build to all common platforms including windows, mac, iphone, android, xbox, and more. Some aspects to consider as weaknesses for using unity is that it is not the best choice if the required task is a to create simple animated 3D object. In order to program that, Unity must be downloaded and set up. Visual Studio IDE and .NET framework are mandatory for scripting on Unity. Also in terms of communication between the application and the web page containing it, it would be cumbersome to achieve this communication using unity whereas three.js would naturally have access to the DOM, as well as any JavaScript snippets on the web page.

Next, we discuss three.js, the library used by project A. Three.js is a JavaScript WebGL library that runs in the browser and integrates well into existing website code. Three.js handles common graphics feature like shadows, rendering, post effects, vertex processing, shaders, etc. It is a lightweight library that takes care of most of the complex low-level graphics and shaders programming. Being open-source, three.js is completely free to use and customize. In terms of weaknesses, being open-source, the library is rapidly changing and as a result, it can be difficult to rely on it to build an enterprise level project using it until the codebase matures enough to a consistent stable build. It is also very challenging to build something big using three.js due to the absence of a visual editor and the limited features of JavaScript. Table 5 summarizes the relevant properties and differences between three.js and Unity IDE.

Table 5. Comparison of Three.js and Unity

| | **three.js (project A)** | **Unity (project B)** |
|---|---|---|
| **programming language** | JavaScript | C#, JavaScript |
| **visual editor** | no | yes |
| **features and tools** | basic | rich |
| **build target** | web browser | all common platforms |
| **license** | open-source | free and commercial |

# 5. DISCUSSION

The purpose of this study is the compare the two built projects and highlight the advantages of disadvantages of the different tools and methods used to build them. In this chapter, we discuss in more detail some points based on the analysis. We also give our recommendation based on our experience of building the two projects. We conclude the chapter by shedding some light on how to develop this work further.

## 5.1. BIM models

As shown before, BIM models contain additional information beyond the geometric data. Those additional features are useful for collaboration and other purposes. However, those features come at the cost of bigger file size and additional time to hit the BIM server and load the model. This means, for small-scale projects that do not require additional information beyond the geometry information, it would be redundant to use BIM model. We believe that for a small smart apartment, using BIM incurs unnecessary additional load time. Consequently, we extracted a mesh model out of the provided BIM and used it to save bandwidth and to simplify our workflow.

## 5.2. SOAP web service

Project B utilized SOAP web service to load the sensors data. SOAP format is language and platform independent which makes it easy to use on any platform. There are additional advantages for using SOAP web services, yet the SOAP XML data format presents a major disadvantage that makes it considerably slow to transfer data compared to its counterparts. While XML is a human friendly data format that is easy to read and edit even with a simple text editor, it is verbose due to the text-based delimiters it uses. Even worse, SOAP standard injects extra XML tags in the SOAP response, which bloats the response unnecessarily. Considering the response of the web service call to load a sensor's data in project B. The response look like the following (Appendix 2):

```
<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://fidelix.fi/webVision/">
  <getPointDataResult>23.673</getPointDataResult>
  </getPointDataResponse>
 </soap:Body>
</soap:Envelope>
```

As seen from the code block above, the XML response is unnecessarily bloated due to the XML verbosity and the SOAP format tags although the only required data is the sensor's temperature value (23.673). If JSON is utilized, the same result could look like the following:

```
{“data”: 23.673}
```

Thus, an important improvement to project B is to utilize JSON format instead of SOAP XML. JSON is a common data transfer format similar to XML but it is much more compact as it uses curly brackets to delimit stored values instead of text nodes like in XML. This makes JSON a lightweight format that is faster to parse and transmit. Since the sensor network consists of a large number of nodes and the sensors data is sent periodically, we believe that utilizing JSON instead of XML would greatly improve the overall system performance.

### 5.3. Unity vs Three.js

We used three.js and JavaScript to build project A, and unity and C# programming language to build project B. Based on our experience using both SDKs, we can say that unity was more productive and easier to use. In particular, the powerful visual editor that unity provides 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 nevertheless is not strongly typed nor object-oriented language. This makes writing maintainable code with it is a bit challenging. Additionally, unity provides a rich framework of component-based features that are easy to integrate with existing code. As a result, we conclude that to create any non-trivial 3D simulation, unity would provide a more productive pipeline compared to Three.js.

# 6. CONCLUSION

The vision of smart buildings is rapidly progressing. Investment in smart building solutions is growing across different regions and business segments. Consequently, creating an easy-to-use 3D user interface to assist non-technical users to take full command of their smart apartment facilities is of a major concern. We presented in this thesis two approaches for building a visualization application that visualizes sensors data in smart buildings. The underlying principles and the building blocks of the two applications have been discussed. We analyzed and discussed the advantages and disadvantages of several of the approaches we used in our implementation. The applications were designed to run smoothly in a modern WebGL-enabled web-browser. We kept the controls and visualizations simple so that all user categories can use the application easily. This work contributes towards increasing the usability of smart buildings monitoring and configuration and enhancing the user experience for IoT services generally.

We highlighted in this thesis a small subset of the smart home vision, which is sensors visualization. However, the vision of smart apartments extends largely beyond that. In a smart home, everything is connected and controlled. This includes lighting, heating, ventilation, air conditioning, and security. In addition, home appliances like washing machine, refrigerator, oven, etc. are also connected using wired and wireless communication technologies. That said, to extend this work, additional types of smart sensors need to be installed and subsequently visualizations need to be developed to visualize the sensors data in the 3D visualization application. Moreover, the projects we developed work in one direction, it takes the sensors data and visualize it. The other direction is equally important for smart apartments. Allowing the client application to control the real apartment to adjust lighting, temperature, and more is an interesting subject for future work. Furthermore, there is a big space for experimenting with new techniques for sensor data visualization. Porting this work to mobile devices would also be of significant importance. It would allow people to monitor and configure the smart environment anytime anywhere. Other ideas open for future research include the use of voice commands, camera gestures, visualizations in virtual reality (VR) and augmented reality (AR).

# 7. REFERENCES

[1] De Cristofaro, E., & Di Pietro, R. (2013). Adversaries and countermeasures in privacy-enhanced urban sensing systems. IEEE Systems Journal, 7(2), 311-322.

[2] Dutta, P., Aoki, P. M., Kumar, N., Mainwaring, A., Myers, C., Willett, W., & Woodruff, A. (2009, November). Common sense: participatory urban sensing using a network of handheld air quality monitors. In Proceedings of the 7th ACM conference on embedded networked sensor systems (pp. 349-350). ACM.

[3] Abdelzaher, T., Anokwa, Y., Boda, P., Burke, J., Estrin, D., Guibas, L., ... & Reich, J. (2007). Mobiscopes for human spaces. IEEE pervasive computing, 6(2).

[4] Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., & Peterson, R. A. (2006, August). People-centric urban sensing. In Proceedings of the 2nd annual international workshop on Wireless internet (p. 18). ACM.

[5] Voogt, J. A., & Oke, T. R. (2003). Thermal remote sensing of urban climates. Remote sensing of environment, 86(3), 370-384.

[6] Resch, B., Mittlboeck, M., Lipson, S., Welsh, M., Bers, J., Britter, R., & Ratti, C. (2009, June). Urban sensing revisited–Common scents: Towards standardised geo-sensor networks for public health monitoring in the city. In Proceedings of the 11th International Conference on Computers in Urban Planning and Urban Management-CUPUM2009, Hong Kong (pp. 16-18).

[7] Lane, N. D., Eisenman, S. B., Musolesi, M., Miluzzo, E., & Campbell, A. T. (2008, February). Urban sensing systems: opportunistic or participatory?. In Proceedings of the 9th workshop on Mobile computing systems and applications (pp. 11-16). ACM.

[8] Ono, T., Ishizuka, H., Ito, K., Ishida, Y., Miyazaki, S., Mihirogi, O., & Tobe, Y. (2007, November). Uscan: Towards fine-grained urban sensing. In International Workshop on Real Field Identification (RFId2007).

[9] Farias, C., Soares, H., Pirmez, L., Delicato, F., Santos, I., Carmo, L. F., ... & Dohler, M. (2014). A control and decision system for smart buildings using wireless sensor and actuator networks. Transactions on Emerging Telecommunications Technologies, 25(1), 120-135.

[10] Meyer, S., & Rakotonirainy, A. (2003, January). A survey of research on context-aware homes. In Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21 (pp. 159-168). Australian Computer Society, Inc..

[11] Robles, R. J., & Kim, T. H. (2010). Review: context aware tools for smart home development. International Journal of Smart Home, 4(1).

[12] Kleissl, J., & Agarwal, Y. (2010, June). Cyber-physical energy systems: Focus on smart buildings. In Proceedings of the 47th Design Automation Conference (pp. 749-754). ACM.

[13] Morvaj, B., Lugaric, L., & Krajcar, S. (2011, July). Demonstrating smart buildings and smart grid features in a smart energy city. In Energetics (IYCE), Proceedings of the 2011 3rd International Youth Conference on (pp. 1-8). IEEE.

[14] Wang, Z., Wang, L., Dounis, A. I., & Yang, R. (2012). Multi-agent control system with information fusion based comfort model for smart buildings. Applied Energy, 99, 247-254.

[15] Martins, J. F., Oliveira-Lima, J. A., Delgado-Gomes, V., Lopes, R., Silva, D., Vieira, S., & Lima, C. (2012, October). Smart homes and smart buildings. In Electronics Conference (BEC), 2012 13th Biennial Baltic (pp. 27-38). IEEE.

[16] Främling, K., Oliver, I., Honkola, J., & Nyman, J. (2009, October). Smart spaces for ubiquitously smart buildings. In Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009. UBICOMM'09. Third International Conference on (pp. 295-300). IEEE.

[17] Stefanov, D. H., Bien, Z., & Bang, W. C. (2004). The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives. IEEE transactions on neural systems and rehabilitation engineering, 12(2), 228-250.

[18] Lu, J., Sookoor, T., Srinivasan, V., Gao, G., Holben, B., Stankovic, J., ... & Whitehouse, K. (2010, November). The smart thermostat: using occupancy sensors to save energy in homes. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (pp. 211-224). ACM.

[19] Friendly, M. (2008). A brief history of data visualization. In Handbook of data visualization (pp. 15-56). Springer Berlin Heidelberg.

[20] Fan, F., & Biagioni, E. S. (2004, January). An approach to data visualization and interpretation for sensor networks. In System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on (pp. 9-pp). IEEE.

[21] Su, J. M., & Huang, C. F. (2014). An easy-to-use 3D visualization system for planning context-aware applications in smart buildings. Computer Standards & Interfaces, 36(2), 312-326.

[22] Healey, C. G. (1996, October). Choosing effective colours for data visualization. In Visualization'96. Proceedings. (pp. 263-270). IEEE.

[23] Buschmann, C., Pfisterer, D., Fischer, S., Fekete, S. P., & Kröller, A. (2005). Spyglass: a wireless sensor network visualizer. Acm Sigbed Review, 2(1), 1-6.

[24] Hu, Y., Li, D., He, X., Sun, T., & Han, Y. (2009, November). The implementation of wireless sensor network visualization platform based on wetland monitoring. In Intelligent Networks and Intelligent Systems, 2009. ICINIS'09. Second International Conference on (pp. 224-227). IEEE.

[25] Boers, N. M., Chodos, D., Huang, J., Gburzynski, P., Nikolaidis, I., & Stroulia, E. (2009, April). The Smart Condo: Visualizing independent living environments in a virtual world. In Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on (pp. 1-8). IEEE.

[26] Saito, T., Miyamura, H. N., Yamamoto, M., Saito, H., Hoshiya, Y., & Kaseda, T. (2005, October). Two-tone pseudo coloring: Compact visualization for one-dimensional data. In Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on (pp. 173-180). IEEE.

[27] Schneiderman, S. C. J. M. B. (1999). Information Visualization: Using Vision to Think.

[28] Cañas, A. J., Carff, R., Hill, G., Carvalho, M., Arguedas, M., Eskridge, T. C., ... & Carvajal, R. (2005). Concept maps: Integrating knowledge and information visualization. In Knowledge and information visualization (pp. 205-219). Springer Berlin Heidelberg.

[29] Chen, C., & Yu, Y. (2000). Empirical studies of information visualization: a meta-analysis. International Journal of Human-Computer Studies, 53(5), 851-866.

[30] Kapler, T., & Wright, W. (2005). Geotime information visualization. Information Visualization, 4(2), 136-146.

[31] Keim, D. A. (2002). Information visualization and visual data mining. IEEE transactions on Visualization and Computer Graphics, 8(1), 1-8.

[32] Heer, J., Card, S. K., & Landay, J. A. (2005, April). Prefuse: a toolkit for interactive information visualization. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 421-430). ACM.

[33] Plaisant, C. (2004, May). The challenge of information visualization evaluation. In Proceedings of the working conference on Advanced visual interfaces (pp. 109-116). ACM.

[34] Yi, J. S., ah Kang, Y., & Stasko, J. (2007). Toward a deeper understanding of the role of interaction in information visualization. IEEE transactions on visualization and computer graphics, 13(6), 1224-1231.

[35] Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C. D., & Roberts, J. C. (2011). Visual comparison for information visualization. Information Visualization, 10(4), 289-309.

[36] Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. International Journal of Communication Systems, 25(9), 1101.

[37] Eastman, C. M., Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2011). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons.

[38] Autodesk Revit [Computer software]. Retrieved April 6, 2017, from http://www.autodesk.com/products/revit-family/overview

[39] Unity Technology. Unity (Version 5). Retrieved April 6, 2017, from https://unity3d.com

[40] Cabello, R. (n.d.). Three.js r84. Retrieved April 06, 2017, from https://threejs.org/

[41] Autodesk 3DS Max (Version 2017) [Computer software]. Retrieved April 6, 2017, from http://www.autodesk.com/products/3ds-max/overview

[42] Autodesk, B. I. S. (2002). Building information modelling. Autodesk Inc. White Paper, San Rafael, CA.

[43] Documentation, Unity scripting languages and you – Unity Blog. (n.d.). Retrieved April 06, 2017, from https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/

[44] Suresh, P., Daniel, J. V., Parthasarathy, V., & Aswathy, R. H. (2014, November). A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In Science Engineering and Management Research (ICSEMR), 2014 International Conference on (pp. 1-8). IEEE.

[45] Risden, K., Czerwinski, M. P., Munzner, T., & Cook, D. B. (2000). An initial examination of ease of use for 2D and 3D information visualizations of web content. International Journal of Human-Computer Studies, 53(5), 695-714.

[46] Robertson, G. G., Card, S. K., & Mackinlay, J. D. (1993). Information visualization using 3D interactive animation. Communications of the ACM, 36(4), 57-71.

[47] Ware, C., & Franck, G. (1994, October). Viewing a graph in a virtual reality display is three times as good as a 2D diagram. In Visual Languages, 1994. Proceedings., IEEE Symposium on (pp. 182-183). IEEE.

[48] Marcus, A., Feng, L., & Maletic, J. I. (2003, June). 3D representations for software visualization. In Proceedings of the 2003 ACM symposium on Software visualization (pp. 27-ff). ACM.

[49] Kjellin, A., Pettersson, L. W., Seipel, S., & Lind, M. (2010). Evaluating 2D and 3D visualizations of spatiotemporal information. ACM Transactions on Applied Perception (TAP), 7(3), 19.

[50] McMahan, A. (2003). Immersion, engagement and presence. The video game theory reader, 67, 86.

[51] Gauntlett, D. (n.d.). VISUALISATION. Art Lab. University of Westminster. Retrieved April 06, 2017, from http://www.artlab.org.uk/visualisation.htm

[52] Quigley, A. (n.d.). Information Visualisation: Proceedings. Los Alamitos Calif.: IEEE Computer Society, 2002. University of St Andrews. Retrieved April 06, 2017, from https://aquigley.host.cs.st-andrews.ac.uk/downloads/tc/TradingConsequencesInfoVisMay17.pdf

[53] Harrower, M., & Brewer, C. A. (2003). ColorBrewer. org: an online tool for selecting colour schemes for maps. The Cartographic Journal, 40(1), 27-37.

[54] PointLight. (n.d.). Retrieved April 06, 2017, from https://threejs.org/docs/api/lights/PointLight.html

[55] DirectionalLight. (n.d.). Retrieved April 06, 2017, from https://threejs.org/docs/api/lights/DirectionalLight.html

[56] ThingsSpeak. (n.d.). Retrieved April 06, 2017, from https://thingspeak.com/

[57] Älykäs ratkaisu energiansäästöön ja hyvään sisäilmaan. (n.d.). Retrieved April 06, 2017, from https://www.fidelix.fi/

[58] Three.js / editor. (n.d.). Retrieved April 07, 2017, from https://threejs.org/editor/

[59] U. (n.d.). Fast Facts. Retrieved April 08, 2017, from https://unity3d.com/public-relations

[60] Unity - Engine Features. (n.d.). Retrieved April 08, 2017, from https://unity3d.com/unity/engine-features

[61] Tracy, P. (2016, July 25). What is a smart building and how can it benefit you? Retrieved April 11, 2017, from http://www.rcrwireless.com/20160725/business/smart-building-tag31-tag99

[62] Haynes, G. (2017, December 11). Beyond Smart Cities. Retrieved April 11, 2017, from https://www.linkedin.com/pulse/beyond-smart-cities-george-haynes-5948850577994760192

# 8. APPENDICES

Appendix 1. Project A data request result

Appendix 2. Project B data request result

Appendix 1. Project A data request result


```
{
    "channel": {
        "id": 8,
        "name": "Sensor data",
        "field1": "SensorId",
        "field2": "SensorType",
        "field3": "Value1",
        "field4": "Value2",
        "field5": "Value3",
        "field6": "Value5",
        "field7": "Value5",
        "field8": "Value6",
        "created_at": "2016-08-23T07:01:24Z",
        "updated_at": "2017-03-21T11:37:06Z",
        "last_entry_id": 3246796
    },
    "feeds": [
        {
            "created_at": "2017-03-21T11:35:12Z",
            "entry_id": 3246697,
            "field1": "84283",
            "field2": "1",
            "field3": "19.89",
            "field4": "18.76",
            "field5": "994.2",
            "field6": "3.12",
            "field7": "71.0",
            "field8": "0"
        },
        {
            "created_at": "2017-03-21T11:35:13Z",
            "entry_id": 3246698,
            "field1": "106258",
            "field2": "4",
            "field3": "0",
            "field4": "600",
            "field5": "0",
            "field6": "1000",
            "field7": "0",
            "field8": "400"
        },
        …
    ]
}
```

Appendix 2. Project B data request result


```
<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://fidelix.fi/webVision/">
    <getPointDataResult>23.673</getPointDataResult>
    </getPointDataResponse>
  </soap:Body>
</soap:Envelope>
```