Deliverable: D 4.1

Organisation: TCD

FP7 – 285229 – Collaborative Project

Knowledge-based energy management for public buildings through holistic information modelling and 3D visualization

KnoholEM

Deliverable 4.1: Web based application - Smart building simulator, visual monitoring tool and user-in-the-loop on demonstration object.

Authors	Kris McGlinn (TCD) and Anton Gerdelan (TCD)
Reviewers	Kelvin Jones (BRE)
Delivery due date	M23
Actual submission date	M24
Status	Public

1.1 Executive Summary

The delivery report DR4.1 - “Web based application - Smart building simulator, visual monitoring tool and user-in-the-loop on demonstration object.” describes the work performed in work package 4 and refers to the task “Assemble Simulation, Integrate Visualisation and Integrate “User-in-the-Loop”. This task is to ensure that the validation objects building geometry and aesthetics, appropriate devices and user interactions, are incorporated successfully into the simulation platform to facilitate simulation of use cases and scenarios. This task will also ensure that devices and data are visualised in a manner which best supports real time intelligent monitoring of the building and its energy usage. Finally, this task will ensure that interactions are supported between users and the building through graphical tasks.
Development to date has been driven by requirements produced by Task 2.1 “Analysis of building use” which has conducted requirements gathering on the demonstration objects and resulted in the development of a set of use cases and scenarios for testing purposes, which are referred to here where appropriate. The purpose of these use cases is to show user interaction support with the developed ontology through querying and enhancing the building specific ontology and visualisation of the data for monitoring. The visualisation of building elements and data are captured at a high level in the generic ontology through the concept “Visualisation”. Here descriptions of how the different building elements are to be represented visually are described as well as how data is to be represented, for example the current state of a sensor or historic data in graph form. How data is represented has a relationship to the role of the person querying the data. This section details the work completed on the web based interface with respect to two main users in relation to the above objectives:

- Facility Managers
- Building Occupants

2 Contents

2.1 Introduction

This report details the progress towards developing the web-based interface for visualising devices and data in a manner which best supports real time intelligent monitoring of the building and its energy usage and that the interface supports interactions between users and the building through graphical tasks. The interface communicates directly with the ontology developed in DR1.1 and DR2.2 and also with data stores created during the data mining in DR2.2. It is developed to meet requirements resulting from use cases and scenarios developed in DR2.1.

2.2 Use Cases

2.2.1 Introduction

This section presents the initial use case development for the SmartBuildVis tool. These use cases have been developed to meet the requirements of the different scenarios and use cases for reducing energy consumption in the five building demo objects, as identified in DR2.1. The SmartBuildVis tool is to address requirements of two building user types: Facility Managers and building occupants.

2.2.2 Facility Manager

The most important task the FM must be able to accomplish is to monitor the energy consumption of the building to determine whether it is within expected energy consumption thresholds. The first use case addresses this requirement.

2.2.2.1 Building Energy Performance Monitoring

Building energy performance monitoring [1] enables an FM to select areas (or zones) in a building, as well as building entities, and assign those entities (based upon historical behaviour) values within which they are expected to perform. For example, that a zone which encompasses a room should not exceed a certain amount of energy consumption over a certain period of time. Visual tools will inform the FM when values fall outside the expected performance metrics. The SmartBuildVis tool must therefore enable the FM to select zones and monitor sensors within those zones so that they may assign it an appropriate performance metric. A zone may cover a room or a floor, and contain multiple sensors and sensor types (e.g. temperature, humidity, energy meter) to monitor the energy consumption of the zone and devices within it, as well as the state of the zone (e.g. its temperature). In order to define the appropriate performance metric, historical data can be visualised, or recommendations can be made by the RT controller (as described in DR3.x).

In Figure 1 we see five main use cases that need to be met by the tool. The first is to view and select a particular zone through the interface. Within this use case, a number of sub tasks are possible, but not defined here, which are related to selecting the particular types of sensors within that zone which are to be monitored. The capability to then select a time period within which data for each of those sensors within the zone is returned is required, so that historical data can be viewed. This includes input given by the users themselves (see section xxx). Finally, the visual tools should support the FM monitoring not only historical and current performance, but also predicted performance, as determined by the simulations conducted in DR5xx.

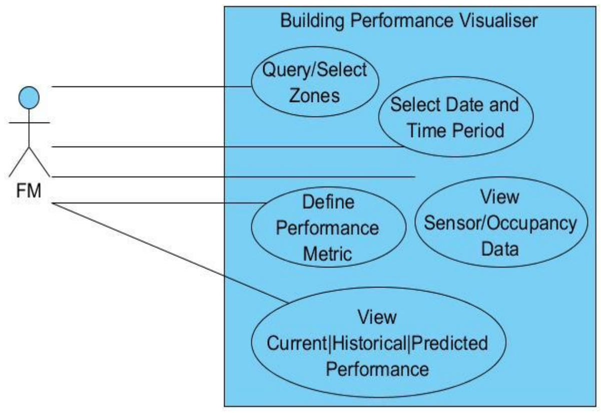

Building Performance Visualiser
FM
Query/Select Zones
Select Date and Time Period
Define Performance Metric
View Sensor/Occupancy Data
View Current|Historical|Predicted Performance

Figure 1 Building Energy Performance Monitoring Use Case

2.2.2.2 Building Control Interface

The building control interface is required to support the FM to both manually control the building systems through the interface and to create rules for the RT controller described in DR3x. The FM can therefore act upon the information he is being presented with from the building performance monitoring tool through this interface. The interface has two main functions, firstly, it must query the building ontology to determine the different applications which can be configured and secondly, it must present the FM with usable interfaces for accomplishing this configuration.

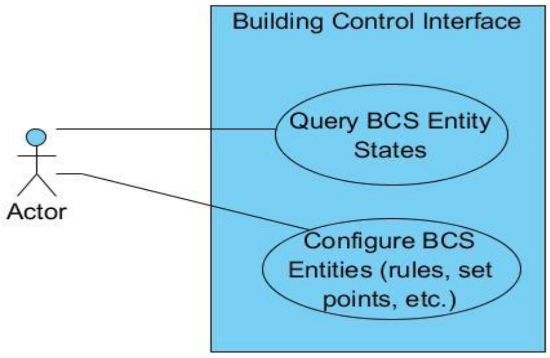

Building Control Interface
Actor
Query BCS Entity States
Configure BCS Entities (rules, set points, etc.)

Figure 2 Building Control Interface Use Case

2.2.2.3 Zone and Path Modeller

The final FM use case details functionality required in order to add and remove zones. For this, the interface must support the visualisation of zones, their creation, their selection and their deletion. In addition, in order to support the occupant modelling to supplement knowledge about individual occupant behaviours within the building, as detailed in DR2.1 and DR2.2, the interface must also support the creation of paths in a similar manner.

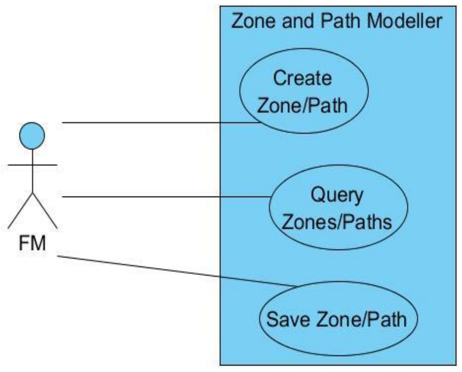


Figure 3 Zone and Path Modeller Use Case

2.2.3 Building Occupants

The building occupants can provide valuable information regarding their behaviour to the knowledge base [2]. To enable occupants to provide such data, novel interfaces are required which support them to quickly enter in data regarding their activities in the building, and how they move through the building. This use case identifies the main functionality which must be supported by such an interface. These are the capability to select the date for entering in their activities, the capability to query previous activities, and finally the ability to enter in and save current activities.

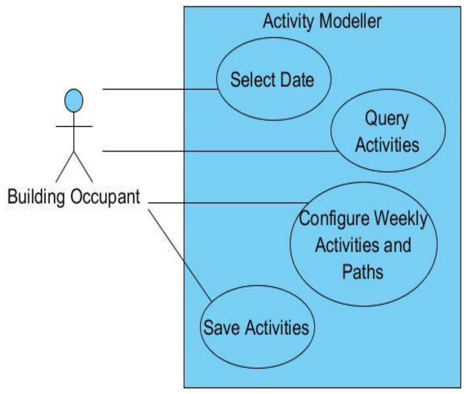


Figure 4 Activity Modeller Use Case

2.3 Web Based Visualisation and Configuration Interface

2.3.1 Introduction

This section describes the web-based interface which supports the use cases described in the previous section. It highlights the different functionalities supported, the chosen technologies and the rational for choosing them. The communication between the client and the server (e.g. for querying historical data, or device/zone state) is not addressed in this section and can be found in section 2.4.

2.3.2 Building Performance Visualisation

Building energy certification standards require that modern buildings meet strict energy consumption targets. However, facility managers do not currently have the necessary tools to monitor building performance [1]. Methods are therefore required which allow FM's to configure how a particular zone or entity within a building should be performing. Making use of historical, current and predicted energy consumption, an intuitive visualisation interface has been developed which makes use of a traffic light system for displaying when that zone or entity is within its performance metrics. A red light indicates when it is performing below its performance metric, an orange light, when it is performing within it (i.e adequately), and finally a green light when it is performing better than expected (bottom left of Figure 5). This simple visualisation technique builds upon previous work [1] and can give a quick indication to the FM about the performance of the selected zone or entity. This visual feature is implemented using jQuery [3], which is a flexible javascript library for creating many event driven visual widgets.

2.3.2.1 Sensor Data and Occupancy Data Visualisation

In order to make more detailed comparisons between historical data, current data and predicted data, the interface provides functionality to select a zone and display sensor data for each sensor in that zone. This can then be overlaid with predicted energy consumption for that zone. Figure 5 (bottom centre) shows the chart with examples of data from three sensor types deployed in Office 02 in the forum building (see deliverable 2.1 for more information on this building). Also displayed here is the occupancy data for one user (who is assigned a unique i.d.) provided by the Activity Modeller interface, discussed in more detail in section 2.3.5. The chart is implemented using a jQuery library called HighCharts [4], which supports multiple graph visualisations using json [5] data objects.

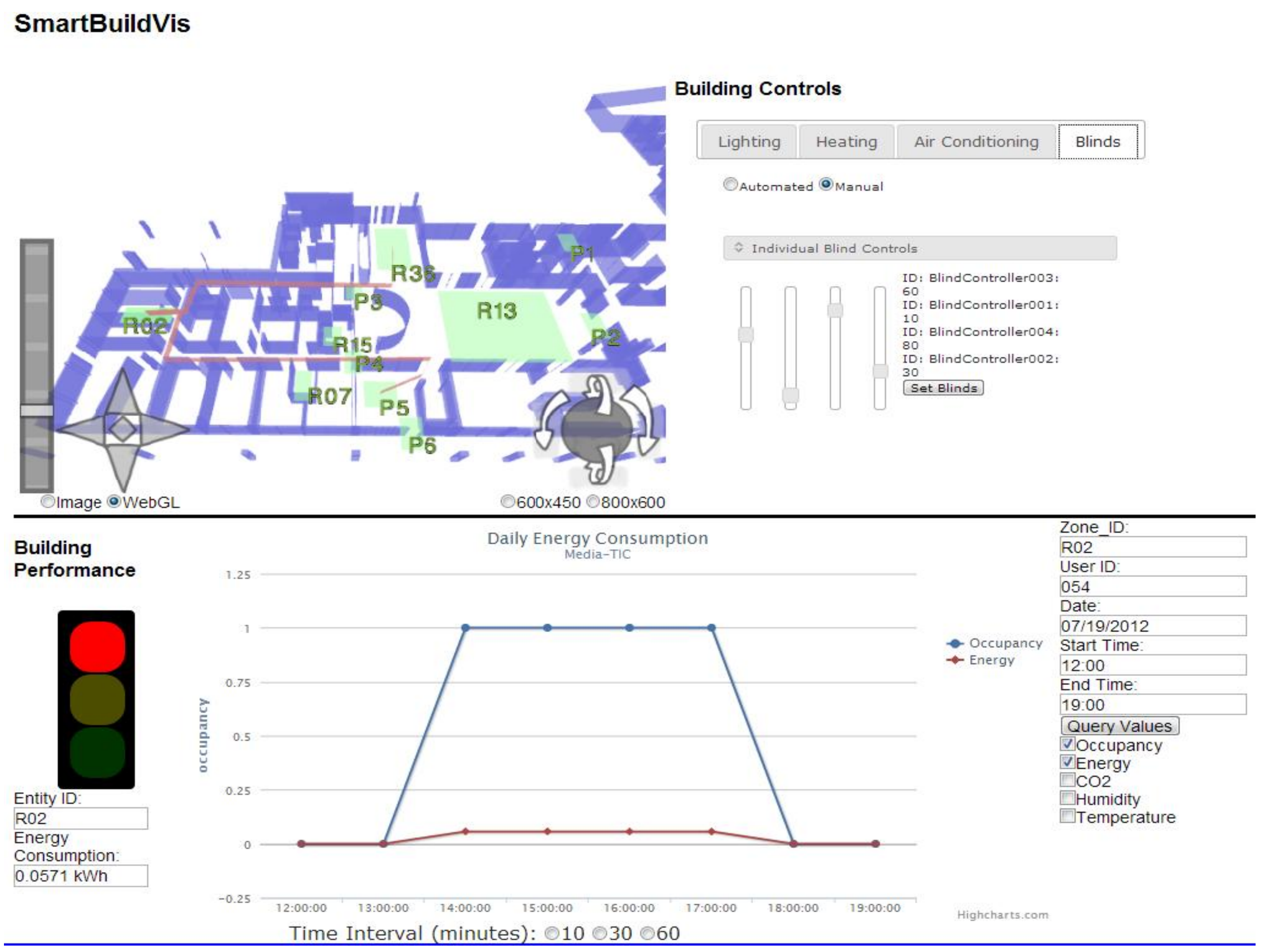


Image [✓ WebGL]

600x450 800x600

Automated [✓ Manual]

ID: BlindController003: 60
ID: BlindController001: 10
ID: BlindController004: 80
ID: BlindController002: 30

Entity ID: R02
Energy Consumption: 0.0571 kWh

Zone_ID: R02
User ID: 054
Date: 07/19/2012
Start Time: 12:00
End Time: 19:00

[✓ Occupancy]
[✓ Energy]
CO2
Humidity
Temperature

Time Interval (minutes): 10 30 60

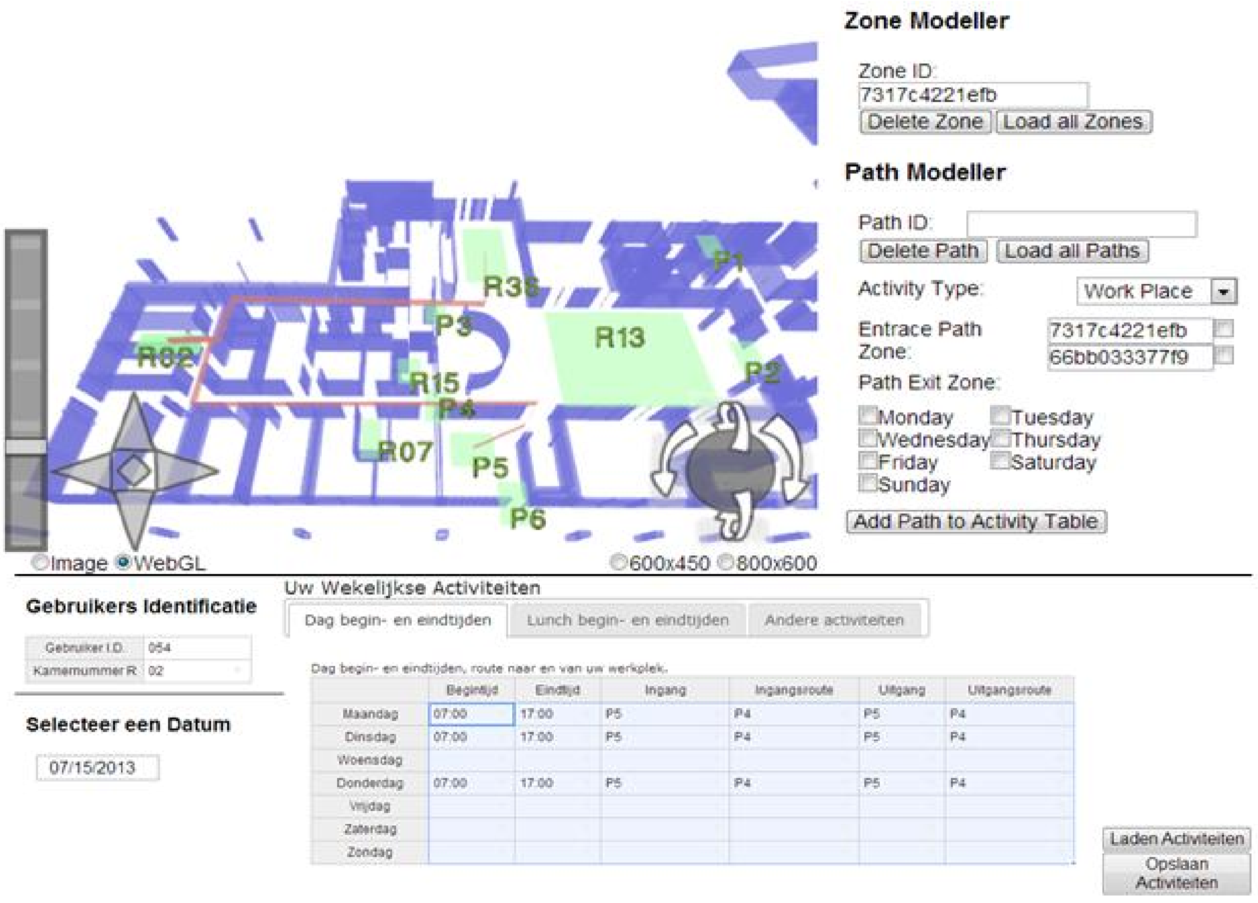
Figure 5 Building Energy Performance Monitoring and Building Control Interface

2.3.3 Building Controls Configuration

Existing Building Control and Building Management systems (BMS/BCS) provide controls for configuring HVAC, lighting and other systems, for example, the state of blinds, windows etc. (as discussed in DR3.3). The SmartBuildVis interface supports these types of configurations through the development of a javascript library of visual jQuery widgets to provide functionality to configure different values of the building state (as stored in the ontology). For example, to configure the set point of a heating system, or to configure the percentage that blinds are opened or closed (see top right Figure 5). This library aims to support all the existing functionality of the BMS and BCS of the demonstration objects, whilst being extendable to support new functionality as required. The interface will also support the additional configuration of rules to control the buildings behaviour, for example, to adjust heating settings dependant on occupancy. As yet this functionality has not been implemented.

2.3.4 Zone and Path Modelling

The Zone modeller supports the FM in creating zones within the WebGL interface. When created, each zone is assigned a unique i.d. and a symbolic value, for example R02. Zones can be related to activities which take place in the building, for example, office work. Also, a zone can be assigned to an area containing sensors which records data relevant to the monitoring of energy consumption, for example presence sensors or energy meters. By then selecting the zone, this data can be displayed in the chart described in section 2.3.2.1. The Path modeller supports the FM creating specific paths around the building which individual occupants may take to get to and from activity zones, and where they pass zones which contain sensors to detect their presence. For example, an office zone may be connected to a zone containing a presence sensor, which in turn is connected to an exit/entrance. When a user leaves their office, they must apss tis presence sensor, and thus, this gives an indication of their likely destination. This modelling of paths is therefore required to support the functionality of the Activity Modeller, described in the next section.



Zone Modeller

Zone ID: 7317c4221efb

Path Modeller

Path ID:

Activity Type: Work Place

Entrace Path Zone: 7317c4221efb

66bb033377f9

Path Exit Zone:

☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday ☐ Sunday

Gebruikers Identificatie

Gebruiker I.D.	054
Kamernummer R	02

Selecteer een Datum

07/15/2013

Uw Wekelijkse Activiteiten

Dag begin- en eindtijden | Lunch begin- en eindtijden | Andere activiteiten

Dag begin- en eindtijden, route naar en van uw werkplek.

	Begintijd	Eindtijd	Ingang	Ingangsroute	Uitgang	Uitgangsroute
Maandag	07:00	17:00	P5	P4	P5	P4
Dinsdag	07:00	17:00	P5	P4	P5	P4
Woensdag						
Donderdag	07:00	17:00	P5	P4	P5	P4
Vrijdag						
Zaterdag						
Zondag						

Figure 6 Zone and Path Modeller + Activity Modeller

2.3.5 Activity Modeller

The activity modeller tool enables building occupants to supplement existing knowledge about user behaviour with data on their occupancy in the building and their movement patterns around the building. This data is entered in for a one week period and this process is to be repeated at the end of each month in order to build up a more accurate picture of their behaviour over time. To use the interface, each user is assigned a unique id. They then select the start and end times they spend in their office, as well as the different paths they take to and from the entrances and exits of the buildings (Figure 6). This is repeated also for their lunch breaks, meetings, as well as other types of breaks (like going to the toilet, smoking, etc.). The activity data is entered into a table, which is created using a jQuery library called handsontables [6]. This provides a flexible method for handling data in table format and is based upon json objects. To view the different locations around the building, the WebGL 3D representation of the building is presented to them, with specific zones marked out (as defined by the FM). It should be noted that an English and Dutch version are currently supported, and work on a Spanish version is underway.

2.4 Communication between Interface and Databases

2.4.1 Introduction

This section describes the communication between the SmartBuildVis tool and the underlying databases.

2.4.2 Database Communication

On the server side two types of database are currently supported. The first is the FUSEKI SPARQL server and the second is an SQL server.

2.4.2.1 FUSEKI

Each building object has its own FUSEKI server storing the building specific ontology for that building. These store all the state data for each building, which is updated every 10 minutes (see DR3x). Using AJAX and SPARQL, SmartBuildVis maintains a connection to the ontology. On opening SmartBuildVis, it begins a dialogue with the ontology and aligns this to the update rate of the RT controller (DR3.x). The FM can also make SPARQL updates, to change the state of the ontology (i.e. the set points of heaters, etc.). The dimensions and properties of zones and paths are also stored in the ontology, along with activities. Although, depending on its impact on performance of the RT controller, activities could be stored elsewhere, for example, in an SQL database. The number, types and id's of the different sensors and devices in the building are also queried from the ontology to support the configuration of building controls and actuators and the querying of historical data from an SQL database. This later process is described in the next section.

2.4.2.2 SQL

The SQL database stores all the historical data from the sensors in a building. Once again, each building has its own dedicated SQL database and server. SmartBuildVis maintains, client side, the number, ids, and placement of all sensors as queried from the ontology. Using these id's, a connection is created using AJAX and php. This code returns json objects, which can be used client side by the jQuery libraries to visualise the required data.

2.4.2.3 Sequence Diagram Examples

Figure 7 demonstrates SmartBuildVis's communication with both FUSEKI and the SQL database. SmartBuildVis must communicate with the ontology to access state data on the building. For example; the current temperature, humidity, or metered energy value from a range of deployed sensors. This state data is updated every ten minutes. In Figure 7 we see the FM begins by loading zones from the ontology to be displayed in the WebGL interface. They then select a zone of interest and configure a time period. They indicate they wish to visualise data on both metered energy readings and occupancy data. This is then returned in a format which can be displayed using highcharts (json objects).

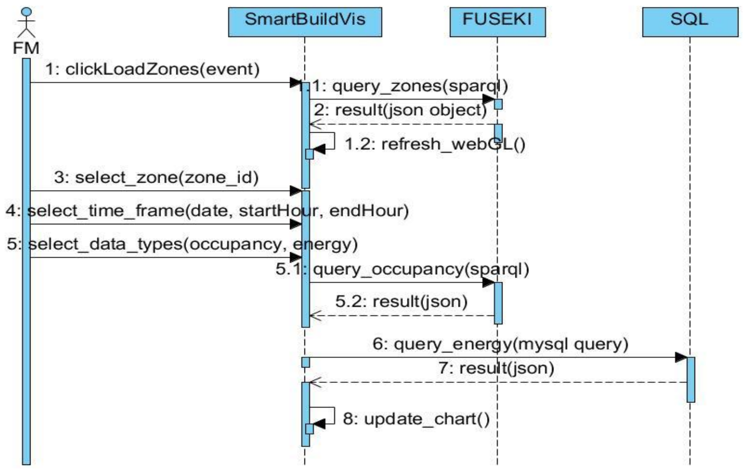


Figure 7 Communication between SmartBuildVis, FUSEKI and the SQL database.

Figure 8 shows the communication when an occupant wishes to view their past activities and also update their current activities in the ontology. Here no communication goes on between SmartBuildVis and the SQL database, as all the occupancy data is currently stored in the ontology. The same process of updating the ontology which is used for saving current activities is used when updating the state of control systems in the ontology, by the FM.

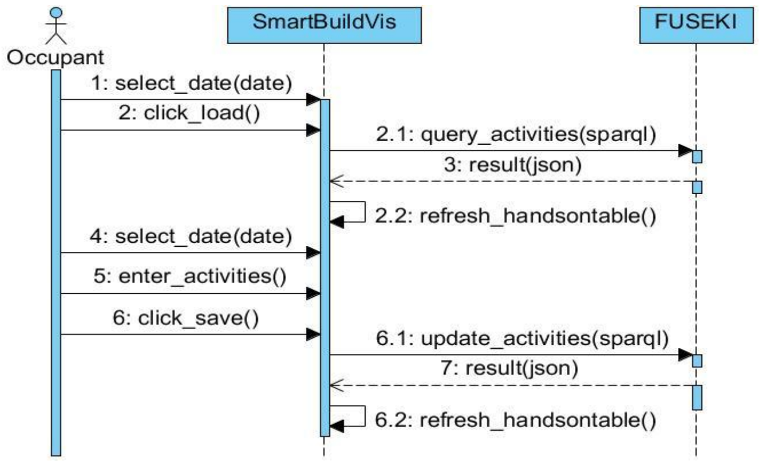


Figure 8 Communication between SmartBuildVis, FUSEKI and the SQL database.

2.5 Building Visualisation

2.5.1 Introduction

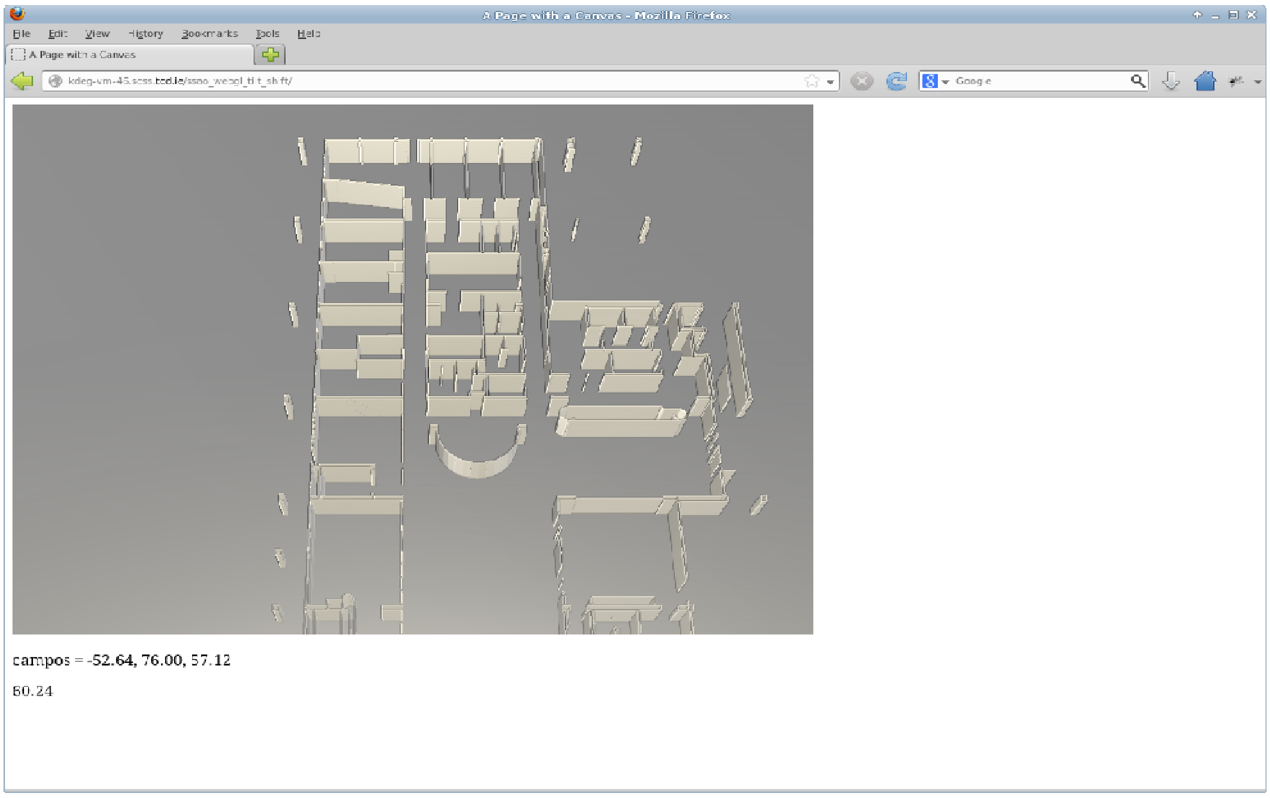
We have explored several different visualisation techniques for rendering the structural CAD models, and for communication with the user *via* on-screen GUI (graphical user interface) controls and highlighting of pertinent spatial information. WebGL is a web-interface to 3d graphics which directly uses a computer's specialised computer graphics hardware. WebGL has an almost identical interface and language to OpenGL ES (embedded systems graphics library). These two features mean that we are significantly more advanced than the current state of the art of CAD rendering, both in performance, and in porting to all desktops, independent to the operating system, and to mobile devices. The most advanced contemporary project of note is the ViziCities project [7], which visualises the greater London area and displays a range of statistics visually. We have implemented some of the same advanced rendering techniques in this project to similar effect.

2.5.2 Post-Processing Techniques

We have prepared a number of optional WebGL-driven post-processing filters for the building visualisation tools. The filters leverage the ability of WebGL to render the 3d scene in a number of different passes; when each pass is rendered it is stored in a 2d image. Subsequent passes may operate on the image from the previous pass, allowing a 2d image-processing filter or further 3d rendering technique to be applied. The following techniques with applicability to improving building rendering have been made available:

- Screen space ambient occlusion (SSAO)
- Miniaturisation filter and area-of-focus Gaussian blur
- Wall-edge outline-drawing using Laplacian of a Gaussian filter

Screen space ambient occlusion (Figure 9) is an advanced rendering technique developed in the video games industry to reduce ambient light in areas of the 2d screen that can be approximated to occlude 3d geometry. It is more computationally efficient than calculating actual occlusion in 3d, and produces a soft shadowing which improves a user's perception of 3-dimensional space represented in a scene.


A Page with a Canvas - Mozilla Firefox
File Edit View History Bookmarks Tools Help
A Page with a Canvas
kdeg-vm-45.scss.tcd.ie/ssao_webgl_tilt_shift/
Google

```
campos = -52.64, 76.00, 57.12
60.24
```

Figure 9 The SSAO technique analyses every pixel of the rendered scene and determines if an area should be in a corner or convex area. This region is very subtly darkened to improve spatial perception of the scene. We can see that areas between walls, for example, are visibly darker, rather than a solid colour which would be difficult to perceive spatially. A gentle Gaussian blur is also applied to the top and bottom regions of the display to provide an illusion of focus in the centre.

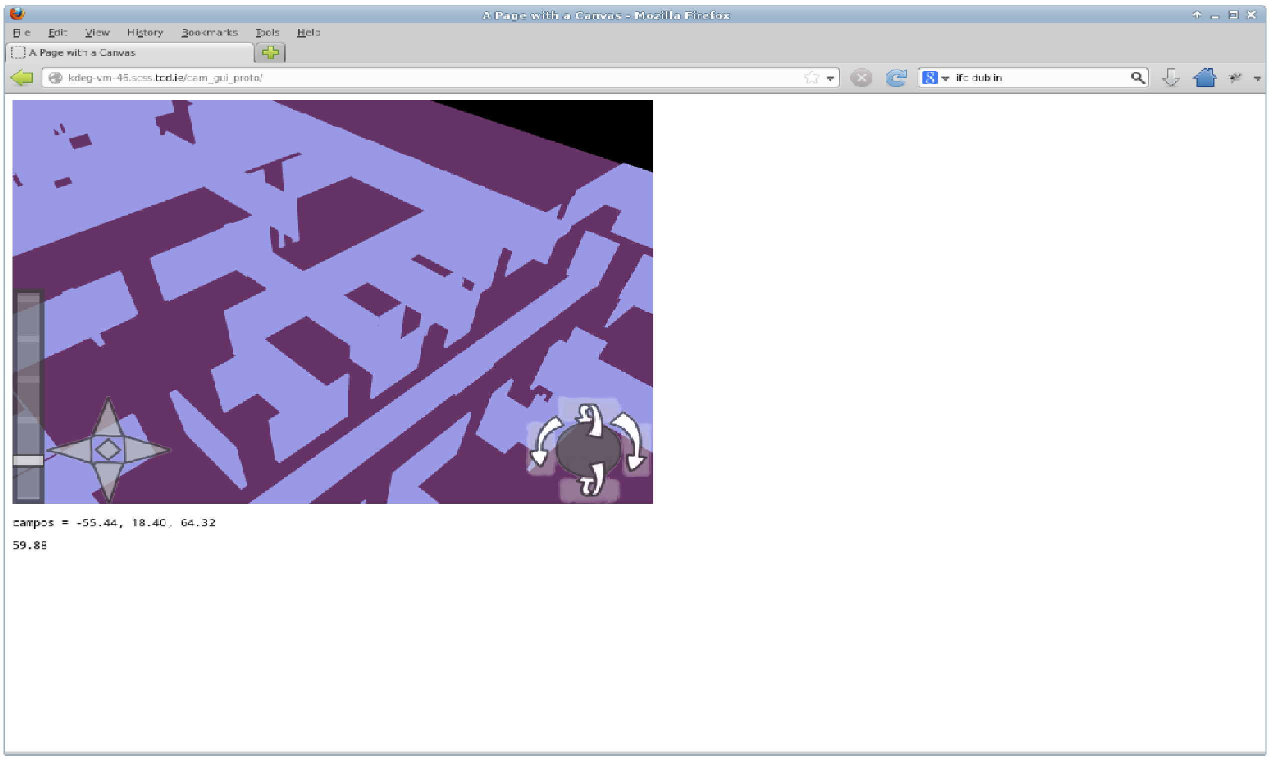
A popular modern visualisation technique in games with an over-hanging view of a scene, and similar experimental building visualisations, is the "tilt-shift" filter which applies a Gaussian blur kernel across opposing portions of the viewport. This produces the appearance of an "in-focus" region in the centre of the user's view, and because of the similarity to physical camera focus, it can also give the illusion of the view being zoomed-in to focus on a miniature model.

3d visualisations constructed from raw geometry do not have an inherent ability to differentiate between surfaces such as overlapping walls - they will appear to blend together into a larger surface. Our basic rendering techniques used Blinn-Phong [8] and diminished lighting to apply approximated scene lighting which can go some way to aiding this perceptual problem, but it would be additionally helpful in a CAD application to also draw black outlines around the extents of surfaces, as you would see in the equivalent 2d plan. One way to do this is as an image filter. We take a second, Gaussian-blurred image of the scene, and apply a Laplacian filter on it to detect the edges. The resulting image is a black-and- white image of the edges. We blur this again, and combine it with the original image to draw outlines on the edges of the walls. This technique is one of several outline-drawing options that we have available. The SSAO algorithm, discussed above, is also essentially an outline-detection algorithm, and can also be used to find more edges more accurately in wall-overlap situations by comparing stored surface normals (facing directions of the wall at each pixel) and the depth buffer (distance from the camera of the wall at each pixel). We then also have a more effective fragment-shader technique applicable to extrusions of 2d CAD models, which is discussed later in this section.

2.5.3 User Communication Techniques

The building visualisation tool is an interactive visualisation, which means both communicating pertinent information to users, as well as allowing users to steer the visualisation itself. Two areas of recent work are:

- Development of mobile device-friendly on-screen controls; graphical user interfaces (GUIs)
- Prototypes of new energy-consumption visualisation paradigms


A Page with a Canvas - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
A Page with a Canvas
kdeg-vm-45.scss.tcd.ie/cam_gui_proto/
ifc dublin

```
campos = -55.44, 18.40, 64.32
59.88
```

Figure 10 A prototype of our on-screen GUI controls which are designed to suit touch-screen displays as well as computer mouse interaction. Controls move the camera up and down (bottom-left), on a horizontal plane (left), and change the orientation (bottom-right).

With a longer-term view to deployment on a range of mobile and touch-screen devices, which are applicable to facility managers and building occupants moving through the buildings, we have built a prototype user interface with large on-screen controls - see Figure 10, above. The task of this GUI is to give the user intuitive controls to navigate the camera through the 3d scene, as well as change the orientation and height of the view. We have written a JavaScript maths library with the appropriate linear algebra functions and encoded Hamilton's unit quaternions (complex number theory) to allow free orientation in 3d.

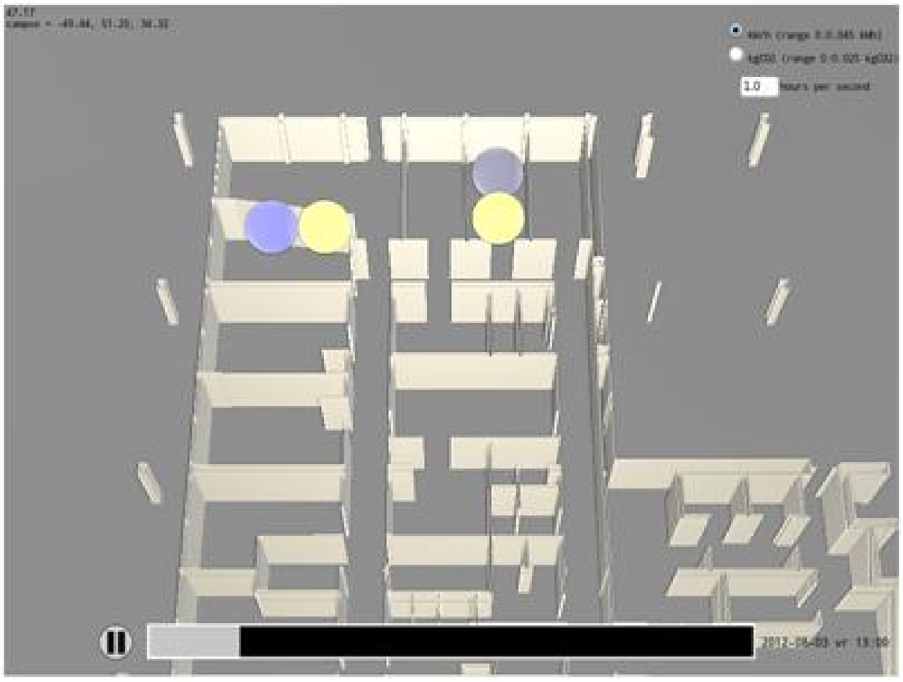


Figure 11 A demo plays back energy consumption data collected in 2 rooms of the Forum building by the PlugWise system. The blue spheres display data collected from computer devices, and the yellow spheres indicate the electric room lights. As consumption increases the colour intensity of the spheres increases (becomes brighter), which can give the facility manager an overview of consumption patterns over the course of the week.

We developed a demo to visualise per-device CO_2 output and energy consumption, as pictured in Figure 11, above. Using a feed of Plugwise-collected data from devices in 2 rooms, we visualise a chunk of collected data in real-time, where the time step compression (speed of playback) can be set by the user via a form field. Indicators are placed in the rooms, which are coloured in greater intensity as CO_2 output, or energy consumption (depending on the mode selected) increases. A time slider GUI element, similar to a video playback slider, allows the user to select a certain time in the playback, or pause playback at an interesting event. This type of visualisation of a time-series of data may help to show a building occupant or facility manager if there are patterns in spatial regions of a building that are consuming a large amount of power or energy over the course of a working week. For further analysis we would expect the user to highlight a region of interest where they can then gather more precise per-room statistical information, such as the 2d chart feedback tool-set.

2.5.4 Importing Building CAD Geometry

Some building CAD models have become available as 3d geometry, but generally most extant public buildings in Europe can provide a 2d digital CAD model. In this project we need to support visualisation of both types of building model. WebGL does not natively render architectural CAD models, which are stored in a variety of file-formats. There is an export-import process that is necessary before we can render a building's geometry at all. Most CAD software will export to some sort of intermediate XML-like format which we can readily interpret, and re-assemble into a format used by modern graphics hardware; triangles created from sets of vertices, known as vertex buffers.

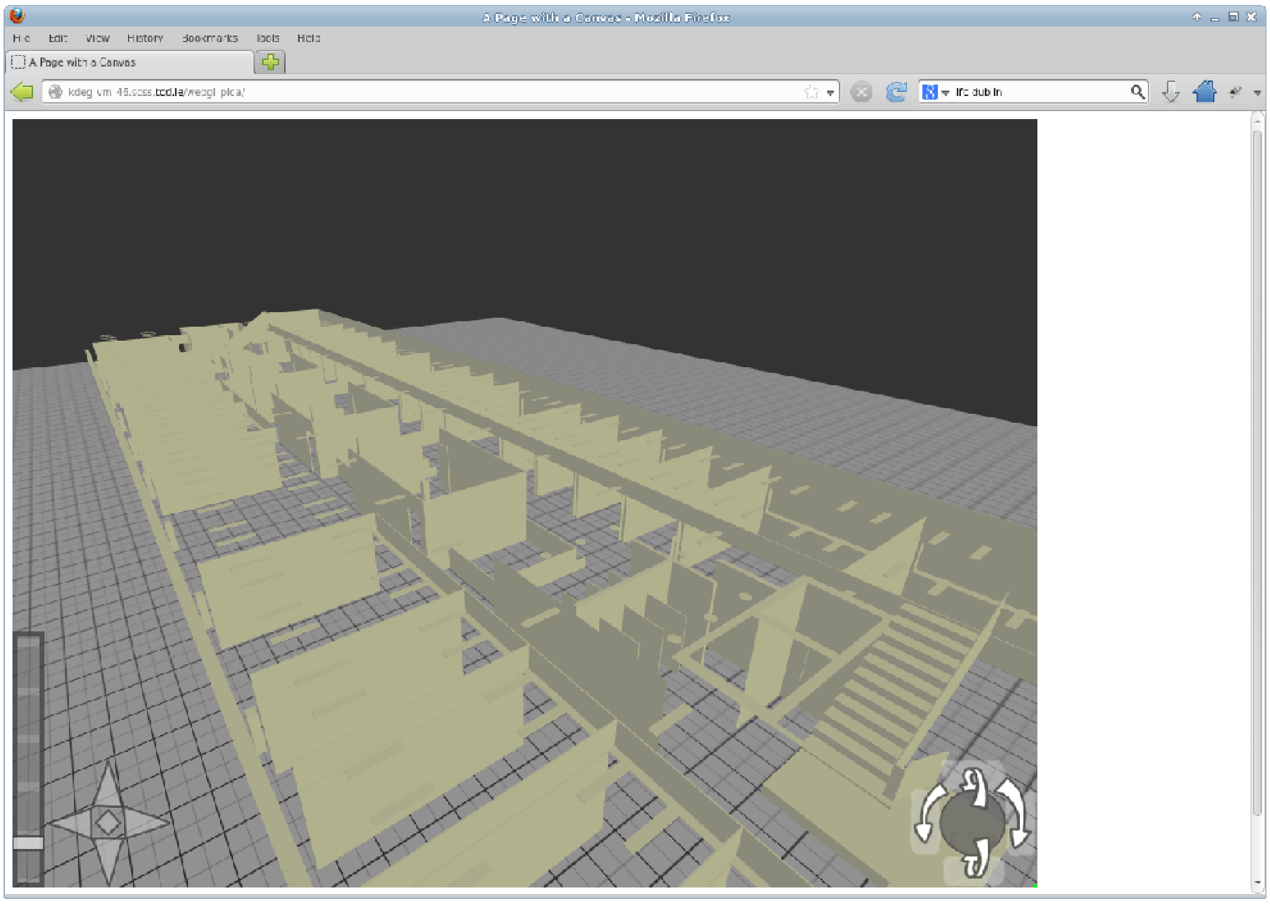

A Page with a Canvas - Mozilla Firefox
File Edit View History Bookmarks Tools Help
A Page with a Canvas
kdeg-vm-46.scss.tcd.ie/webgl-pica/

Figure 12 A 3d CAD model available for a floor of the PICA building has been exported into Collada DAE format.

We built an off-line converter to load this and store it into a JSON (JavaScript Object Notation) format which is trivial to import into our web-programme and store in WebGL vertex buffers. 3d CAD models may require some manual re-assembly in order to either group or separate regions or zones of the building to be more suitable for rendering - see Figure 12, above. Modern graphics processing units (GPUs) are very efficient at rendering large amounts of geometry in parallel, but can only read from a single vertex buffer at a time. Generally we can render entire floors of public buildings that are stored in a single vertex buffer, but if we wish to render different elements with different colours conveying some semantic information, such as windows and doors, then it becomes practical to group all of the doors into one vertex buffer, and all of the windows into another. It is possible to automate most of this process, but there are typically omissions, and human error in CAD models; doors stored as walls, for example, that require some hand editing.

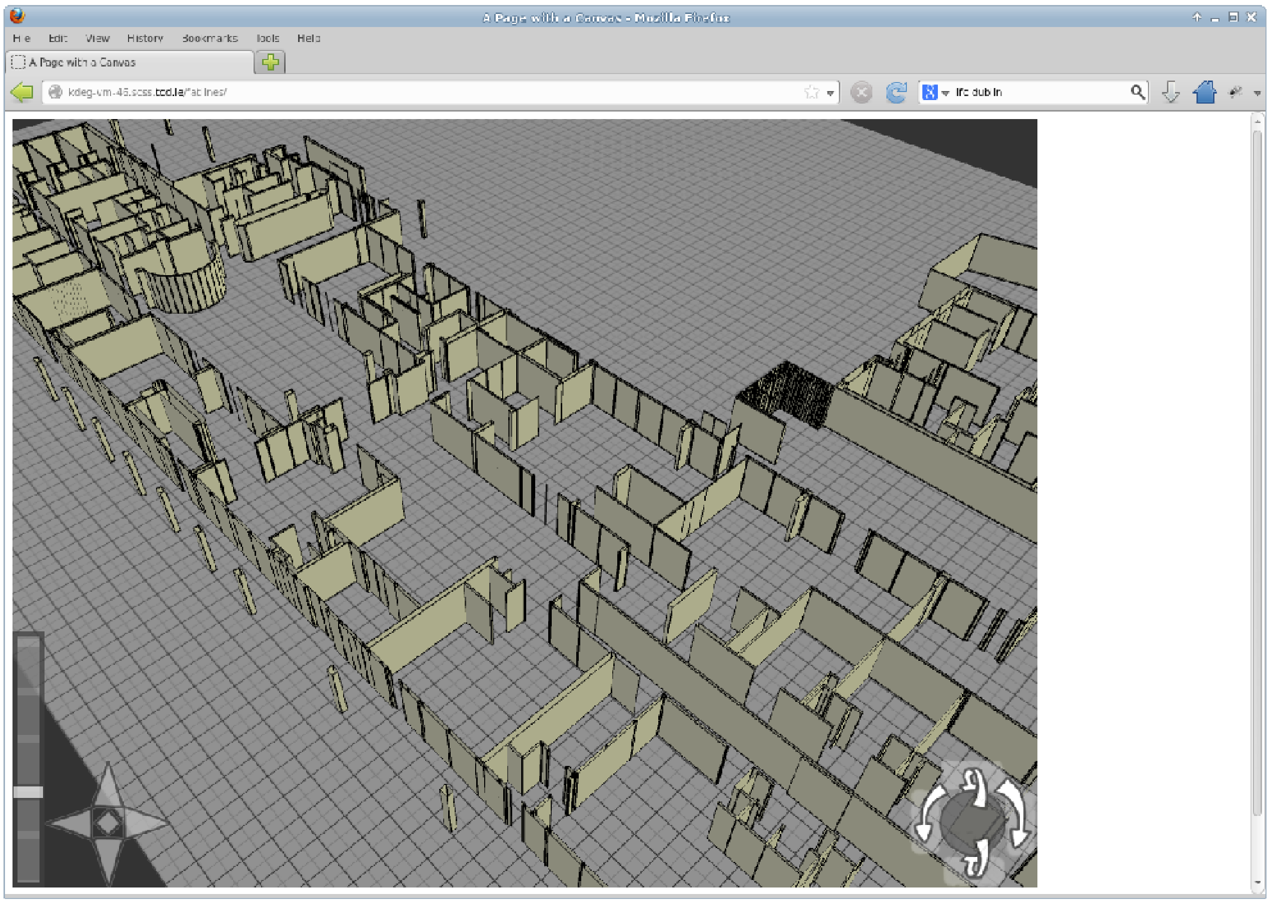

A Page with a Canvas - Mozilla Firefox
File Edit View History Bookmarks Tools Help
A Page with a Canvas
kdeg-vm-46.scss.tcd.ie/~at lines/

Figure 13 A 2d CAD model of the Forum building has been exported to XML and imported into our web-programme. An algorithm generates 4 triangles for every wall section (2 front, 2 back), and the vertices at the corners of these are stored in a WebGL vertex buffer, allowing for very efficient rendering. A shader programme draws outlines on the edges of the walls.

2d CAD models must be exported and imported in the same way as the 3d models, except that we must apply an extrusion to generate 3d walls from the 2d walls. To do this we generate 2 triangles per-wall, and 2 more to allow the user to see the reverse side whilst using the WebGL ability to "cull" back-faces from the rendering pipeline. This means that only one side of each wall is drawn, but we need to create a reversed wall in GPU memory to allow the other side to be seen when the camera moves around.

Without specific knowledge of the heights of individual walls, we can create a convincing spatial model by extruding all walls to a 3 meter height. Once again, storing all walls for a particular floor into the same vertex buffer allows us the exploit the parallel-processing power of the modern commodity GPU. Knowing the uniform height of the walls of this type of building allows us to create a very effective outline-drawing system. Every drawing operation in modern computer graphics is associated with a computer programme that runs on the GPU hardware called a "shader". The shader has two components; a "vertex shader", which positions the vertices of each wall in the correct place, in respect to the camera's perspective, and a "fragment shader", which determines the colour of every pixel-sized area of surface on the geometry. In WebGL, shaders are written in GLSL (OpenGL Shader Language) ES 2.0, which is the same format as those used on mobile devices - which makes porting to those a much simpler task. We inform the shaders of the width and height of the walls, which allows the fragment shader to colour a fixed proportion of the edges as a gradient of dark colours, which produces a pleasing outline.

2.6 Future Work

The report focuses on the forum building, but through the rapid approach to integrating new building models into the WebGL interface, new buildings can quickly be imported, and building models for the Media-tic and PICA building have also undergone this process (for more on these building objects see DR2.1). Current evaluations are also taking place on the usability of the Activity Modeller for Dutch users in the forum. These are not discussed in this report. These evaluations will be extended next to the media-tic building, and then the other building demo objects.

Future work will look at the continued development of the SmartBuildVis tool, which includes:

- Continued usability evaluations for building occupants and facility managers.
- Porting the solution to the Android nexus tablet device to support proactive FMs on the move.
- Further integration with the RT controller to enable writing of rules through the interface.

All the current demos discussed in this report are accessible on the TCD server at this address: http://phaedrus.scss.tcd.ie/buildviz/demos

2.7 References

[1] K. McGlinn, E. Corry, E. O. Neill, M. Keane, D. Lewis, D. O'Sullivan, and E. O'Neill, "Monitoring Smart Building Performance Using Simulation and Visualisation," in *Ubicomp 2010 Workshop: Ubiquitous Computing for Sustainable Energy (UCSE 2010)*, 2010, no. March 2007, pp. 1–8.

[2] K. Mcglinn, A. Gerdelan, K. Jones, and D. Lewis, "A Flexible And Extensible Web-Based Visualisation And Configuration Interface For Holistic Energy Management Of Buildings," in *International Conference on Applied Energy ICAE 2013, Jul 1-4, 2013, Pretoria, South Africa.*, 2013, pp. 1–10.

[3] "jQuery," *Website*, 2013. [Online]. Available: http://jquery.com/. [Accessed: 30-Jul-2013].

[4] "Highcharts - Interactive JavaScript charts for your webpage." [Online]. Available: http://www.highcharts.com/. [Accessed: 22-Jan-2013].

[5] "JSON." [Online]. Available: http://www.json.org/. [Accessed: 30-Jul-2013].

[6] "Handsontable - jQuery grid editor. Excel-like grid editing with HTML & JavaScript." [Online]. Available: http://handsontable.com/. [Accessed: 22-Jan-2013].

[7] "ViziCities ." [Online]. Available: http://rawkes.com/articles/vizicities-dev-diary-1. [Accessed: 31-Jul-2013].

[8] J. F. Blinn, "Models OF Light Reflection For Computer Synthesised Pictures," in *4th annual conference on computer graphics and interactive techniques*, 1977, pp. 192–198.