Development of a Web-Browser Based Interface for 3D Data—A Case Study of a Plug-in Free Approach for Visualizing Energy Modelling Results

Jochen Wendel, Syed Monjur Murshed, Akila Sriramulu and Alexandru Nichersu

Abstract This research explores the usage of freely available open-source resources for the deployment of a plug-in free web-application interface for 3D geospatial data to visualize energy related modelling and simulation results. Such plug-in free web mapping applications will be essential for future cartographic web applications as forthcoming web browsers will no longer support the usage and installation of those plug-ins used in the past. As a proof of concept, a 3D city model of the city of Karlsruhe in Germany consisting of over 87,000 buildings is used as a case study. This data set was compiled using OpenStreetMap data and outputs from energy simulation models. The CityGML format is used for data storage of this multi-domain data set. In order to ensure independence from browser plug-ins, HTML5 and freely available JavaScript libraries are used for the creation of this application. Multiple analytical cartographic and geospatial functions such as cartographic classification, attribute selection, descriptive statistics, spatial buffer analysis and the retrieval of modelling results from a PostgreSQL and PostGIS data infrastructure are implemented in this interface. This paper further discusses some case studies, future enhancement opportunities of the proposed interface and experiences gathered during the interface development process that would help other cartographers and GIScientists in developing future native 3D web mapping applications.

Keywords 3D web mapping · Geovisualization · WebGL · Web mapping API · Energy analysis

J. Wendel (✉) · S.M. Murshed · A. Nichersu
European Institute for Energy Research (EIFER), Karlsruhe, Germany
e-mail: wendel@eifer.org

A. Sriramulu
Karlsruhe University of Applied Sciences, Karlsruhe, Germany

G. Gartner et al. (eds.), *Progress in Cartography*, Publications of the International Cartographic Association (ICA), DOI 10.1007/978-3-319-19602-2_12

1 Introduction

The visualization of urban energy modelling and simulation results is a crucial part in energy research as they act as the main communication tool among scientists and decisions makers. Results from energy modelling and simulations are usually directly linked to spatial objects such as buildings or city furniture. 2D visualisation techniques have been implemented since many years but with the recent emergence of detailed 3D urban data models and their advantages in energy modelling, the visualization in 3D plays a significant role. In some cases, the results are usually not aggregated by the building object but at a finer scale such as building surfaces. This requires a higher grade of visualization and therefore 3D becomes a necessity (Nouvel et al. 2014). Furthermore, the presentation of energy simulation results in 3D enables the decision makes and users to better explore and comprehend outcomes from multiple perspectives (Bahu et al. 2014).

Recently, numerous 3D web Application Programming Interfaces (API) ranging from GoogleEarth[1] to Unity's Game Engine[2] have evolved, but most of the web-mapping services that visualize 3D data are either based on browser plug-ins and require the users to install one or more programs locally on their computing devices. The installation of those plug-ins is often cumbersome and raises compatibility issues with different web browsers. Furthermore, browser plug-ins, especially those that do graphically represent information such as Adobe Flash or Java are prone to security issues (Barth et al. 2010; Soltani et al. 2010). In the future, new releases of major web browsers from Apple, Google, Mozilla and Microsoft will no longer support any plug-ins. This is particularly important to cartographers and GIScientists as many of those 2D and 3D web mapping applications still rely on plug-ins. In addition to those functional limitations, most proprietary web browser plug-ins do not provide a wide range of custom functionalities specific to the application area of urban energy analysis (Wendel and Nicerhsu 2015).

This research focuses on the development of a plug-in free 3D web mapping application for the display of different energy related simulation and modelling results at the urban scale (Fig. 1). The focus is thereby on the usage of free and open software libraries that can be natively run in any web browser. Given the complex multidisciplinary nature of energy related modelling data sets commonly differ in spatial and temporal resolution, data structure or data storage format that requires an intensive data integration workload. A major obstacle in this process is the current proprietary nature of major commercial software packages (Wendel and Nichersu 2015). The lack of interoperability among software packages makes it difficult to share and further process research results. Therefore, all data sets can be stored in an open source data infrastructure that is based on a PostgreSQL database and PostGIS, for spatial capabilities (Simons and Nichersu 2014). A major requirement of this research is the connectively to a PostgreSQL databases and the usage of open-source Geographic Information Systems (GIS) functionality with PostGIS directly from the interface.

[1] https://www.google.com/earth/.

[2] https://unity3d.com/.

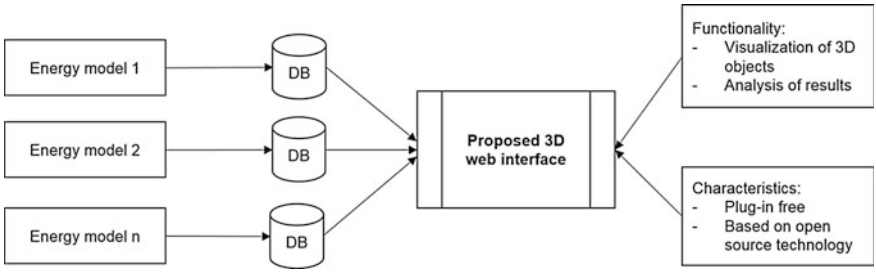

Fig. 1 Proposed 3D web interface for visualization and analysis of different energy related simulation and modelling results

In addition to the interface development, this research further focuses on the lessons learned from this experiment as well as gives an outlook in the field of open-source 3D web mapping interfaces.

In the following section, the evolution of different web-mapping technologies for the visualization of 3D data are summarized. Sections 3 and 4 describes the data and technologies used to develop the web-mapping platform. The methodology and the description of the interface are illustrated in Sects. 5 and 6. Finally, a discussion and future direction of such a plug-in free 3D web visualization are explained in the Sect. 7.

2 The Evolution of Related Research

The visualization of spatial 3D information has experienced multiple popularity peaks during the last two decades. These peaks were tightly coupled with major breakthrough in technological enhancements (Peterson 2015). In establishing visualization technologies, 2D data visualization developments were always on the forefront before this trend was adopted to the visualization of 3D cartographic displays such as the development of HTML and VRLM. Figure 2 highlights major breakthroughs in 2D and 3D web mapping applications and technologies ranging from early GIS desktop systems (Goodchild 1991; Dix et al. 2004) and web mapping applications such as Xerox PARC MapViewer (1993), GRASSLinks (1995), MapQuest (1996) and TIGER Map Server (1997), that mostly provided static displays, to web technologies such as Web Feature Service (WFS), Web Map Service (WMS) and Web Coverage Service (WCS) that made the distribution of geospatial data on the web easier and thus helped to increase web-based 2D mapping applications drastically. These web mapping services provide static image tiles of maps or XML (Extensible Markup Language), using Open Geospatial Consortium (OGC) web-service standards, formatted feature details and coverage data in GML (Geography Markup Language) format, respectively. While Fig. 2 does not show all milestones on 2D and 3D visualization technologies it clearly highlights the trend to web technologies that can visualized both, 2D and 3D.

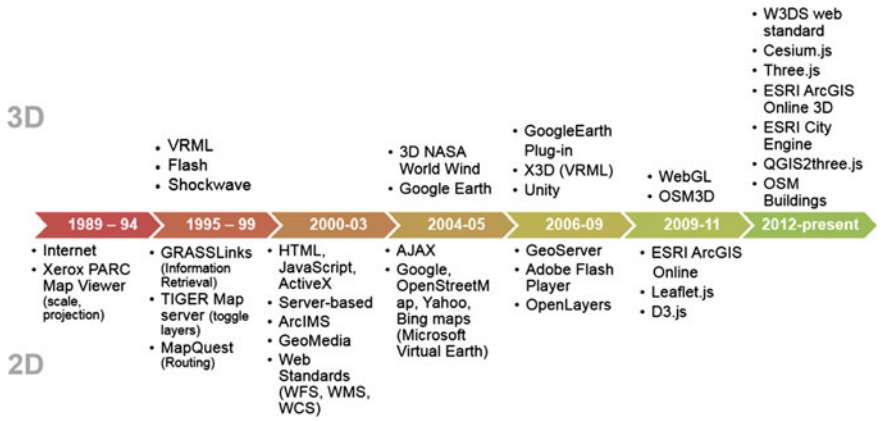

Fig. 2 The evolution of 2D and 3D web mapping milestones

On the visualisation front, all the above mentioned services were limited to displaying static geographic data, with exception to browser plug-ins such as Flash, Shockwave and VRML. After 2005 the adoption of these plug-ins for geospatial data (e.g. Google Earth plug-in) and X3D, the successor of VRML was observed. However, due to the lack of native graphics processing (GPU) support, VRML X3D was unable to handle larger amount of data (Ming 2008). Since 2010, with the emergence of OpenStreetMap (OSM), several 3D research projects such as OSM3D Globe[3] (University of Heidelberg), OSM Buildings,[4] etc.[5] have been initiated at experimental stages which demonstrate the popularity of 3D city models. With the advancement in graphics technology these plug-ins have evolved to display 3D objects or models that were exported from specialised 3D Desktop software (Fig. 2). Recently the adoption of Unity's powerful gaming engine towards the usage of 3D geospatial data in web mapping applications have been demonstrated (Raghothama and Meijer 2015; Jenny et al. 2015). Although Unity provides a preformat way to visualize large quantity of data, it is still reliant on web browser plugins in its current version.

With the support of webGL in all major web browsers, many new libraries were developed in the last few years. For example, the open source library project Cesium.js, based on webGL and not reliant on any browser plug-in gained

[3] http://osm-3d.org/home.en.htm.

[4] http://osmbuildings.org/.

[5] http://wiki.openstreetmap.org/wiki/3D_Development.

increased popularity among cartographers and web developers. For example, the Sunshine project made use of an early adoption of cesium.js for cartographically displaying urban services and energy consumption at the urban scale (Giovannini et al. 2014).

3 Data Sets for 3D Visualization

The generation of 3D city models can be costly and time extensive process (Thiele et al. 2013; Döllner et al. 2006; Huber et al. 2003). Currently, many major cities such as New York, Berlin or Lyon started to provide detailed 3D models of their cities which could be used by any cartographer for visualization and analysis purposes. In addition, OSM provides building data as building footprints and often additional height information, mainly for public buildings. However, the availability highly varies among regions. Out of 5.5 million mapped buildings in Germany, only 5 % have the height information tagged in OSM data, only 7 % of the buildings have the number of floors and 2 % have roof type information specified (Goetz and Zipf 2012).

In this research, the city of Karlsruhe in Germany is considered as a case study area. Karlsruhe is a medium sized city of 300,000 inhabitants located near the Franco-German border in the southwestern part of Germany. The city does not provide any public access to its 3D spatial data. Although building footprints were available for the whole city, the height information was missing or unusable for the objective of this experiment. Instead, the height information of each building was estimated by the number of floors of each building obtained from areal imagery. In total 86,000 buildings were generated at a Level of Detail 1 (LoD1). For each floor, a standard height value in meters, depending on the building type, is used to estimate the total height of a building that provided higher accuracy levels of 92.4 % within a 95 % confidence interval for the whole data set (Saed and Wendel 2015).

Due to interoperability requirements with energy simulation models, the data set is stored in a PostgreSQL database using the CityGML standard.[6] A benefit of using the CityGML standard over other geospatial data formats is that semantic information of each surface or element of a building or object can be stored. This would be particularly interesting to cartographers when designing multi-scale cartography data bases for the definition of cartographic visualization rules across multiple scales (Brewer and Buttenfield 2007).

[6]CityGML is an open data model standard developed by the OGC (Open Geospatial Consortium). See: http://www.opengeospatial.org/standards/citygml.

Table 1 Resources used in interface development

Technologies	
WebGL	3D web standard
HTML	Functional control of web page
JavaScript	Control of elements and data conversion
Python	Database and server connection
Structured Query Language (SQL)	Data base queries
Data exchange format	
GeoJSON	Data transfer to WebGL
Software libraries/APIs	
EXT JS	Interactive web applications
Three.js	3D visualization
D3.js	Conversion of geographic coordinates
D3-ThreeD.js	Link Three.js with D3.js

4 Technological Requirements

The process of designing and implementing a 3D web mapping application that natively runs in any web browser requires the exploitation of different technologies, software and programming APIs. Table 1 summarizes the resources, technologies, software libraries and data formats utilized in this study.

4.1 Technologies

The implementation of the 3D web interface and visualisation requires different technologies and software such as WebGL, HTML, JavaScript, Python and SQL, in a client and server side configuration. HTML and CSS are used for the functional aspect of the front-end of the interface, while other components that interact with data and provide functionality of the web pages, are controlled using scripting languages, namely JavaScript and Python. JavaScript is used to control all the elements in the interface and to manipulate the data which is in the JSON format that constitutes the client-side processing in this application. In contrast, Python and related adapter such as psycopg[7] are used to access functions in PostgreSQL through scripts, to query the database, and to run energy simulation scripts. Regarding the display of 3D data, the WebGL standard is used. It enables the implementation of 3D graphics directly in the web browser without the use of any plug-ins.

[7] http://initd.org/psycopg/.

4.2 Data Exchange Formats

The GeoJSON format is applied for encoding a variety of geographic data structures and for interchanging geospatial data that is based on JSON (Butler et al. 2008). JSON enables easy access and manipulation of objects not only with JavaScript, but also SQL and Python. In addition, it is also advantageous to use this format since it is possible to convert GeoJSON to and from many other spatial data formats such as KML, GML and ESRI shapefiles. For example, using GeoJSON, ViziCities9, an open source project inspired by the massive online multi-player game SimCity, aims to visualize the entire world in 3D.

4.3 Software Libraries/APIs

Several software libraries are utilised to develop the visualisation interface. Ext JS is used for designing the user interaction of the web mapping interface. It is a pure JavaScript application framework for building interactive web applications and is developed by Sencha.Inc.[8] It leverages HTML5 features to provide pre-defined web page elements such as button, text field panels and layouts that can be customised and used in building cross platform web applications. Some advantages of using this API are its Model-View-Controller (MVC) architecture, theming and styling using CSS and SASS, extensive documentation and support (Suderaraman 2013).

Three.js handles the entire 3D visualisation part in this interface. It is a relatively new open-source JavaScript library (released in 2010) that uses the WebGL capability of a browser to create, display and animate 3D objects with a very low level of complexity (Doob 2010). Apart from native geometries such as line, sphere or cube, this library also supports visualization of 3D models exported in JSON format from other software such as Blender, 3D Max and hence supports interaction with GeoJSON. Although Three.js is employed mainly for interactive games development, it has been recently demonstrated to be useful in geographical data representation and web mapping applications (Sandvik 2013).

D3.js is a JavaScript library for manipulating documents based on data using HTML, SVG and CSS and has been used in many web mapping applications (Ledermann and Gartner 2015; Bostock 2011). It combines powerful visualization components and a data-driven approach to Document Object Model (DOM) manipulation. D3.js has a module called *Geo* that can convert the geographic coordinates between 12 major projection systems. It is used in this research for converting geographic coordinates to screen coordinated.

[8] http://www.sencha.com/products/extjs.

d3-ThreeD.js a small library of functions that aims to link Three.js with D3.js (Sutherland 2012). The ability to transform SVG path from D3.js to Three.js object is offered by one of the functions provided by this library. It is used in facilitating the extrusion of the building footprints within the interface.

5 Interface Development and 3D Data Handling

This section describes the methodology of the interface development process as well as the handling of 3D data and their visualization challenges. The interface development process can be further divided into design of interface architecture, interface workflow and interface visualisation. Furthermore, a detailed description of the geometric transformations from 2D to 3D and resulting challenges are described in this section.

5.1 Interface Architecture

A typical client–server (e.g., PostgreSQL database) is used for the architecture of the interface (Bell 2005). Using WebGL, all visualization and rendering processing are performed on the client side while data storage and analytical capabilities are executed on the server side. While the complexity of WebGL is higher than propriety plug-in based environments such as Microsoft Silverlight or Adobe Flash, support and performance of 3d spatial data is more efficient, interactive and responsive (Resch et al. 2014).

5.2 Interface Workflow

In order to visualize and analyse data from the PostgreSQL database, several steps are required. Figure 3 highlights the detailed flow of process between the interface and the database. For data input (Fig. 3 (1)) multiple Python scripts are used depending on the data type. All data Processing (Fig. 3 (2)) is done on the server side. When the request from the interface is sent to the server with the required connection parameters, the data is processed based on the query and sent back to the interface in the GeoJSON format. The spatial coordinates and attributes are retrieved by using the ExtJs functions and subsequently visualised using functions from the Three.js library.

Data processing and visualization is handled on the server and client side. The connection to spatial data sever is established by a Python script using the 'psycopg' adapter and then an SQL query is executed. The SQL query performs the

necessary manipulation of the data through PostGIS extension functions. For instance, *ST_AsGeoJSON* converts the resulting rows from the PostgreSQL to GeoJSON file and *ST_Transform* function converts the spatial coordinates to WGS84 (defined by EPSG:4326) adopted in the interface (Fig. 3 (3)).

The data infrastructures presented in Fig. 3 is particularly of interested when developing an interface to visualize multi-domain datasets. The PostgreSQL database with the PostGIS functionality enables the merger of results from energy model outputs and spatial data. Furthermore, by strictly using a Python environment each independent energy model can be extended and visualized further by directly using PostGIS functions in the PostgreSQL database.

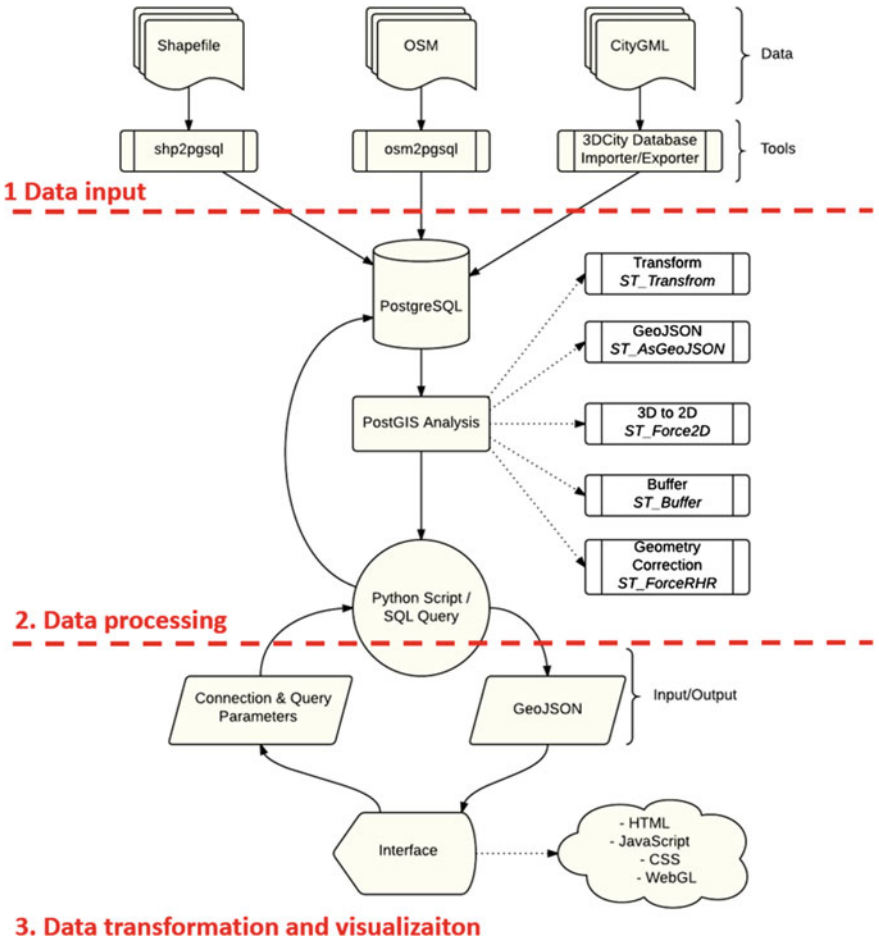

Fig. 3 Detailed interface workflow illustrating data input, data processing, data transformation and visualisation steps

5.3 3D Data Handling and Visualization Challenges

The transformation and handling of spatial 3D data is a major challenge in this research since no out of the box programming libraries that could handle the technical requirements to natively display urban energy data, such as Cesium.js, were available at the start of this experiment. Therefore, coordinate conversion and extrusion had to be handled by custom implementation. The majority of objects in the Karlsruhe 3D city model that are displayed in the interface are buildings, natural features and roads. While the former two can be stored as polygons (Polygon or MultiPolygon), roads are usually stored as lines (LineString or MultiLineString) in a GeoJSON file. Hence, the basic flow of data from PostgreSQL to Three.js for both buildings and roads is common but the handling of these two data types for visualisation is different from each other. The flow of data conversion undergone by a polygon feature footprint is as shown in Fig. 4.

Once retrieved from the database, the coordinates are used to generate an intermediate SVG path using the D3.js library function. This path is then converted to a 3D object. In the scope of Three.js, each object added to the 3D scene is called a *mesh*. In order to be rendered on screen, a mesh requires two parameters namely, THREE. Material and THREE.Geometry, whereas the latter is provided by the building footprints. The *transformSVGpath* function from the d3-threeD.js library converts the path generated by d3.geo object to a Three.js 2D path (THREE.Shape) object. An arbitrary extrusion value is applied to each of the meshes to make them three-dimensional. No extrusion value is applied for the objects representing natural features so that they appear flat and an even lower value of 1 or 2 is applied for water bodies so the water-type polygons overlapping the natural features polygons appear above. An arbitrary extrusion value of 10 (determined through testing) multiplied by the number of floors is applied to the building meshes to give them the required 3D perspective. All the extrusion values are following the Three.js coordinate system.

Several challenges have been encountered when displaying spatial 3D objects natively in a web browser. For example, in the process of converting building objects from a GeoJSON file to SVG using the D3.js library, it was found that the building footprints were not displayed as expected due to their polygon winding order (Grünbaum and Shephard 1990; Neumann and Winter 2001) In order to allow efficient data handling on the client side, the order of coordinates must be reversed directly in the PostgreSQL database before the conversion into a GeoJSON object takes place. Another challenge appeared while displaying line features. For example, in the database road features are stored as LineStrings in the GeoJSON file where each vertex of the line segment is a pair of latitude and longitude coordinates. If converted to SVG, the ends of the line segments are connected resulting in a polygon. In trying to overcome this limitation, the coordinates are directly converted to screen coordinates using *d3.geo* object and passed to the Three.js function to create a *THREE.Line* object. However, this line object, if extruded, will result in a wall rather than a road (Fig. 5, left). Therefore, a rectangular shape was chosen (Fig. 5, right) and extruded along the line object created to build a road. A detailed discussion about line extrusions for WebGL web mapping applications can be found in McNamara et al. (2000) and Deslauriers (2015).

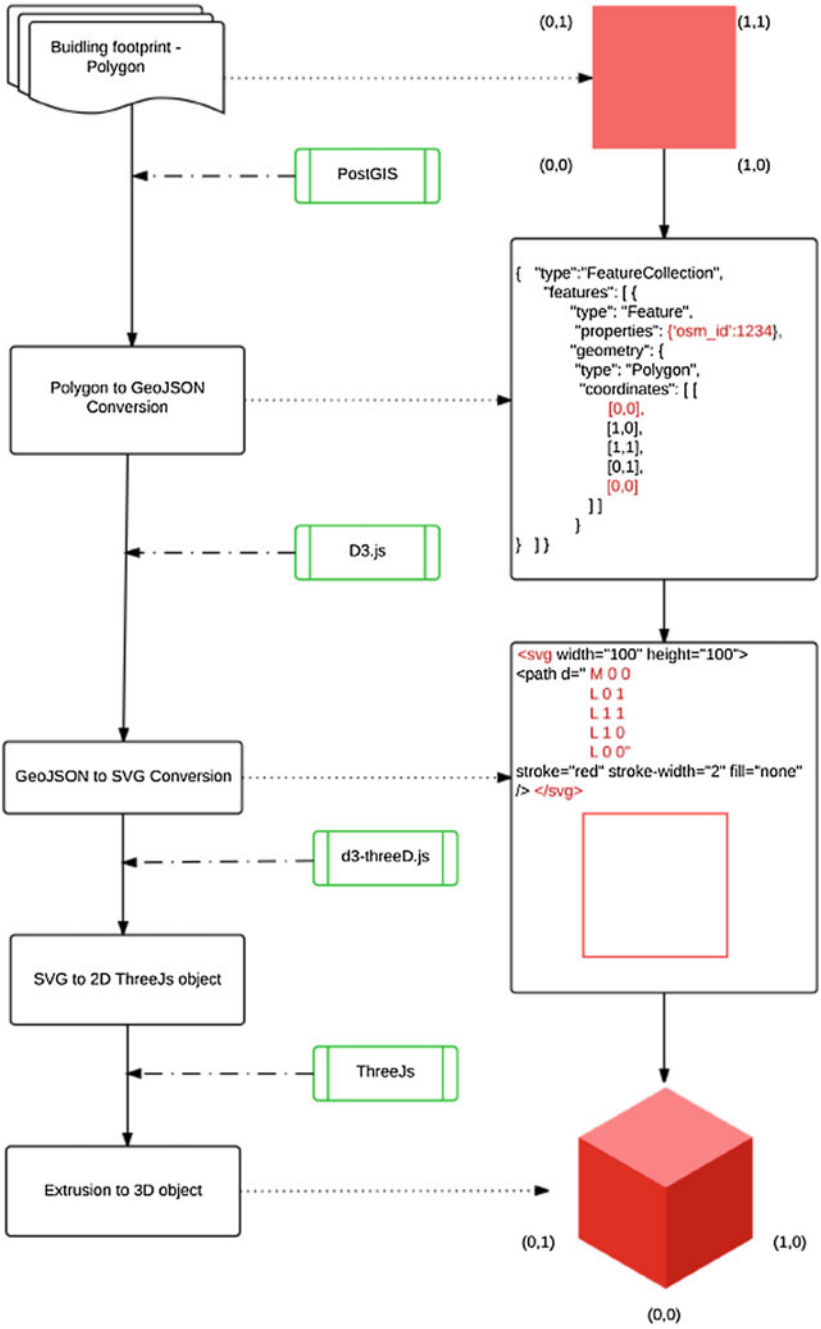

Fig. 4 Dataflow and processes required to extrude 3D features natively in the web browser

Fig. 5 Extrusion of road features: direct extrusion (*left*), shape extrusion (*right*)

5.4 Interface Visualization

The main focus of this research lies in the usage and implementation of open-source libraries for a plug-in free 3D browser based visualization. Therefore, the user interface that has been implemented in the research, has not been gone through user testing or design adaptation phases. However, common Human-Commuter Interaction (HCI) guidelines, for example layout by Dix et al. (2004) were considered in the design process. The main components of the interface layout are:

1. Viewer window
2. Navigation controls
3. Toolbar
4. Action panel
5. Information windows

Furthermore, the interface is divided into front-and back-end. The front-end includes elements such as buttons, windows and message boxes, and the appearance of the interface, whereas the back-end includes the general organization of the files containing the code so that the application remains readable and expandable. Figure 6 shows the buildings and natural features layers directly derived from the PostgreSQL database displayed in the interface while the action panel shows classification options of the data.

6 Interface Functionality and Case Studies

The proposed interface not only allows cartographic visualisation of the 3D features, but also supports the exploration of the results from energy simulation with several functionalities such as layer navigation, feature selection, information retrieval, visibility control, data classification and spatial buffer functions. These functionalities are made accessible through a toolbar available at the top of the browser window (Fig. 6). In this way, scientific cartographic visualisation such as well-defined symbols, harmonious use of colour through ColorBrewer.org (Harrower and Brewer 2003) of spatial data derived from energy modelling is ensured. Moreover, several case studies consisting of the outcomes of energy modelling and simulations are integrated into the interface as a proof of concept.

Fig. 6 A 3D web interface layout consisting of Viewer window (*1*), Navigation controls (*2*), Toolbar (*3*), Action panel (*4*), and Information windows (*5*)

6.1 Retrieval of Information

The user can select and retrieve basic information and energy model outcomes of particular features such as buildings from the database via the interface. When a client request is executed data, in form of a database table with the requested information is loaded from the PostgreSQL server on to the interface and the information is saved in the browser's cache memory. The geometry details of the loaded features get stored in the *THREE.Mesh* object of the Three.js library and the attributes are programmed to be stored as an *Ext.data.JsonStore* object which acts as a local database at the client-side. On selecting or deselecting a building, the *osm_id* of the building is accessed by id of the mesh and added to or subtracted respectively from an array. This id serves to access the type of a selected feature, retrieve corresponding entry in the Ext.data.JsonStore previously constructed and display them in grid format provided by ExtJs (Fig. 7). The Ids stored in the array can be later used for further analysis such as creating charts or buffer.

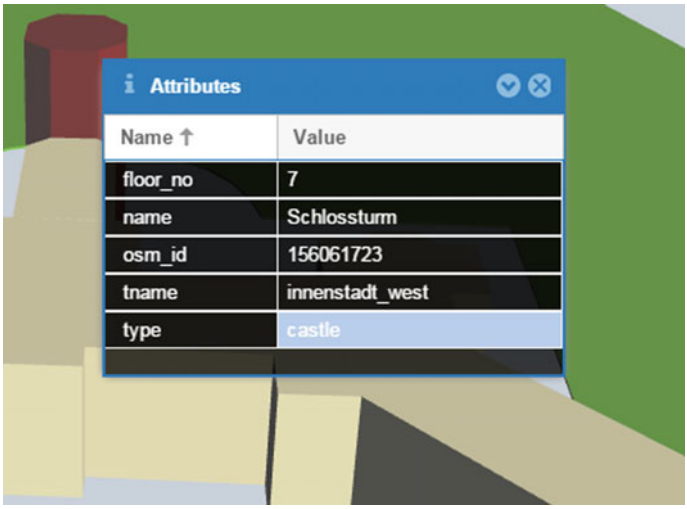

Fig. 7 Retrieval of sample information displaying building attributes

6.2 *Cartographic and GIS Functionality*

Several visualization and spatial functions have been implemented as a proof of concept. As an initial proof of concept for visualization functionalities, the classification of building attributes are demonstrated by the 'Equal-interval' method has been implemented (Slocum et al. 2009). A corresponding legend displays the harmonious colour code while the corresponding class intervals are based on building classification (Fig. 6). Color schemas were derived from the recommendation of ColorBrewer[9] and the symbols were drawn on the screen using the *Ext.draw.Sprite* object from the Ext Js library. This powerful library allows several other built-in cartographic representations of 3D objects.

As a proof of concept for spatial functionalities, buffer analysis has been implemented to identify buildings or any other feature, within a certain search radius of another feature or to find the extent of impact of any phenomenon around points or lines or polygons. 3D buffers enable further explorative analysis, such as noise emission mapping and emergency route planning (Zipf 2010), costs and risks of urban networks (e.g. district heating networks), dynamic air flow analysis (Autodesk Ecotect Analysis 2010), spread or visibility analysis, fire exposure or explosion and are commonly represented as a polyline feature with extruded z-values. In energy research, 3D buffers can be exploited such as to identify the coverage of street lights in order to optimally plan their setting. The implementation of buffers in this research are created by using PostGIS functions and the results are saved back to the PostgreSQL database (Fig. 8).

[9] http://colorbrewer2.org/.

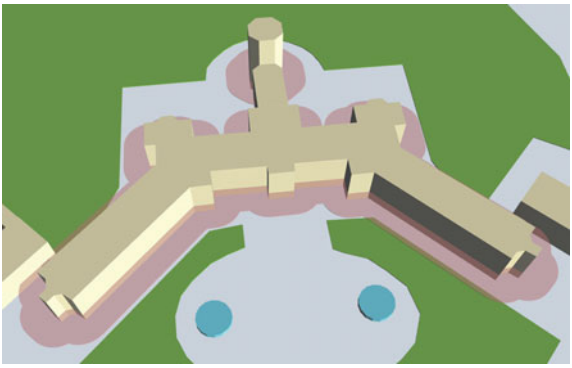

Fig. 8 Line buffer function showing a 10 m 2D circular buffer around a building

6.3 Case Studies

Several energy simulation and modelling results are incorporated into the interface platform in order to test the capability of the interface in terms of technical feasibility, visualization and explorative analysis of results. The outcomes of these energy models are generally stored in a different database and the interface was able to successfully connect to them. In the first case study, the modelling of vertical solar radiation potential of LOD1 and LOD2 buildings facades resulted in a set of values as attributes of a building and building surfaces (Wieland et al. 2015). This included six fields which are comprised of beam and diffuse solar radiation on building surfaces with clear and overcast sky conditions in kilowatt hour (kWh) for each month of a year. Since the values are distributed over certain time intervals line charts are well suited to visualise the distribution and are hence demonstrated in this application (Fig. 9). Charts can be saved and exported into multiple formats as well. Further case studies are tested for example, by incorporating the socio-demographic energy behaviour research on energy consumption for the city of Karlsruhe (Saed and Wendel 2015) and an energy balance model to calculate heat losses from LoD1 and LoD2 buildings (Simons and Nichersu 2014).

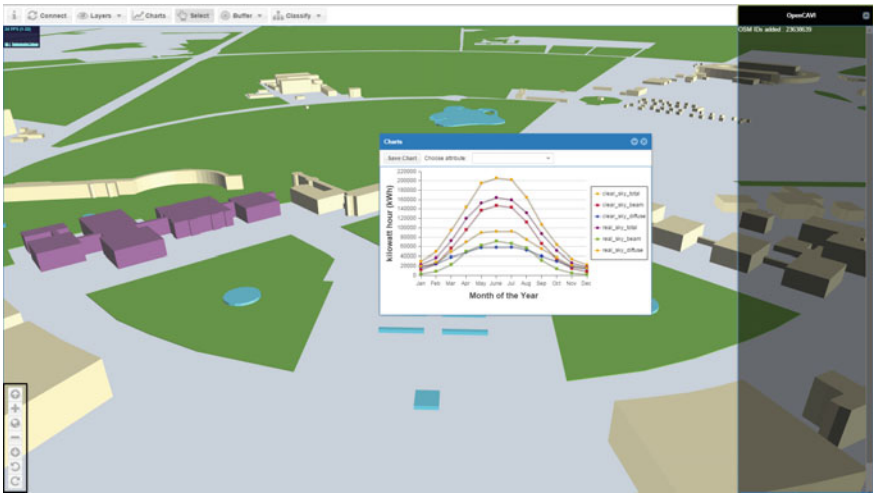

Fig. 9 Visualization of energy modelling results in charts and colour codes

6.4 *Performance of the Interface*

The interface represents the usage of various software technologies, large number of 3D objects, cartographic functionalities, and data from several energy model. Moreover, powerful computer systems (e.g., client, server, and network) have been exploited to load and to display the 3D objects in real time. Therefore, the performance of the interface and its capabilities need to be tested to evaluate its performance.

Generally, three types of the testing strategies such as functional testing, non-functional testing and acceptance testing are conducted for a web application (Liang 2010). The functionalities listed in the earlier section are forms of functional testing. The non-functional testing includes the usage of memory space in the browser, the time required to load the data on the interface and the frames per second (FPS) value of the renderer measure the performance of this interface (Table 2). This test is particularly of interest when developing an interface for displaying buildings as it sets the threshold of how many 3D objects can be

Table 2 Comparison of the performance of the interface in standard browsers

Browser	With 1874 building parts		With 14,000 building parts	
	Memory (MB)	FPS	Memory (MB)	FPS
Chrome	~200	35–45	~400	7
Firefox	~240	57–60	_~940	3
Safari	~273	25–40	~1.73	9–15
Internet explorer	~350	24–38	~960	6–10

Benchmark was run a regular laptop, running Windows 7 64bit (Core I5-4310U, 8 GB DDR3 Ram, Samsung PM851 256 GB SSD, Intel HD 4400 graphics)

displayed simultaneously in the interface. Furthermore, the acceptance or benchmarking tests defines target level scales for visualization of 3D as well as 2D data. A comparative assessment of the performance of the interface with varying number of building parts are performed. The memory and FPS required by some standard browsers are displayed in Table 2. It is important to note that the FPS are monitored by the Three.js library's stats object when one or more datasets of buildings are loaded on to the interface.

Googles' Chrome browser provides the best support for visualisation with higher FPS and lower memory because of its compatibility with the WebGL, compared to the Firefox and Safari browsers. Internet Explorer browser indicated difficulty to run the interface because of its limited compatibility with WebGL. However, it indicates better FPS after implementing the *Detector.js* object which is provided by the Three.js library. This object identifies whether the web browser is able to run WebGL and in case of its absence, *THREE.CanvasRenderer* is used, however, results are then displayed in lower graphics quality. The performance assessment helps to identify the best suitable browser and the limited capability of certain browsers, ascertains the optimal number of buildings or parts to be displayed.

7 Discussion of Results and Future Direction

The process of developing the 3D web mapping interface led to the identification and explanation of different resources, technologies, data exchange formats, libraries, and APIs. Moreover, during different implementation stages, the potential, limitations and opportunities in the field of current 3D web mapping technologies are explored systematically. From these experiences, cartographers and GIScientists could gain a comprehensive overview of current web mapping technologies as well as gain lessons to implement similar application.

Along with the visualization of basic 3D data, several functionalities, such as buffer analysis, and classification of data are integrated into the interface. The suitability of the interface has been demonstrated by integrating several energy research related case studies such as 3D solar potential assessment and visualization of the energy consumption per building. Finally, a comparative assessment on the performance of the interface has been tested in several web browsers. The interface would allow users and energy policy makers not only to visualize different data and modelling results but also enable to use the interface as a tool for analytical and explorative purposes. As the developed open-source data infrastructure of the interface allows flexibility for extensions, other models and simulation such as results from urban simulation or environmental modelling could be shown in the interface.

It is observed that the developed interface for the LOD1 building model, though functional and adequate in certain application areas, demonstrated that the depiction of the real-world on a computer screen is a tedious process and leaves opportunities for further research and enhancement. For instance, improvements in computing

power such as computer graphics to handle the implementation of textures as well as standardization (OGC and W3C) to obtain building height information are especially subject to further development. While new implementations in 3D web mapping such as cesium.js has seen an increase in performance for visualizing 3D objects in a higher resolution than LoD2, WebGL is still reliant on the performance of the client side computer system. To overcome this major bottleneck some 3D web mapping services that do visualized high detailed and textured spatial data, such as CyperCity3D[10] are based on streaming services. Thereby, the rendering of all 3D objects is done on the server side and content is streamed to the client. However, the interactive remote visualization of 3D graphic data in a web browser remains a challenging issue due to latency (data volume) and the adoption of heterogeneous networks (Lavoué et al. 2013).

Moreover, the lack of standardization in data models and data-exchange was a major obstacle in this development process. For example, the user must know how data is structured if the data source is unknown. It was observed during the retrieval of data for the chart (Fig. 9) that the SQL query contained the exact field names of calculated energy modelling related output attributes. This will not be an issue if a standardized naming convention such as conventions from the INSPIRE initiative or the CityGML Application Domain Extension (ADE) Energy is followed for the attributes of a building. Furthermore, data formats and tools have to be constantly modified to suit changing technology. For example, O3D[11] is a free API for generating 3D graphics built basically as a plug-in for desktop and web browsers. Recently, this API was re-built to suit the growing interest in WebGL. It is the same case with X3D[12] which is an open standard for 3D graphics. X3D files can be incorporated into web pages by Applets which is a type of plug-in. Due to WebGL, a X3DOM[13] JavaScript library was developed to include 3D graphics without plug-in. These changes are however reasonable considering that most of the APIs were built predominantly for gaming or CAD applications and not necessarily for geographic data.

Open standards do exist for 2D geographical data services such as WFS and WMS where the request for information is returned in formats such as GML, SVG and images, respectively. However, such standards for 3D geodata (e.g., W3DS, Web 3D service) are still in a draft versions and need to be further defined. W3DS will however, be the standard portrayal service for 3D geodata such as landscape models, city models, textured building models, vegetation objects, and street furniture (Open Geospatial Consortium 2010).

The variety of data sources and in particular the storage of this data for 3D visualization showed that the adoption of open-standards is a promising solution to interoperability, scalability and to harvest the maximum benefit from a 3D

[10] http://www.cybercity3d.com/#!streaming/r2pa2.

[11] https://code.google.com/p/o3d/.

[12] http://www.web3d.org/x3d/what-x3d.

[13] http://x3dom.org/.

web-based GIS interface not only for data models (e.g. CityGML and X3D) but also for data distribution services such as W3DS across different platforms and visualization (WebGL).

Future enhancements of this interface will include further cartographic functionalities such as integration of map tiles from 2D mapping services such as OSM or other topographic map providers, integration of a Digital Elevation Model (DEM) for terrain visualization and incorporation of textures as well as shading of buildings. Furthermore, at the time of the development of this interface, libraries such as Cesium.js weren't fully available to use and future development should be focused in incorporating these recent technologies. In addition further benchmarking will be undertaken with a focus on user experience such as browsing (panning) of 3D models as well as a comparison of the performance of different implementation strategies such as the usage of different open-source APIs versus proprietary plug-in based approaches by using the same 3D data set and spatial extent.

Acknowledgements The authors would like to extend sincere gratitude towards the two anonymous reviewers for improving the paper. Furthermore, the authors would like to thank their colleagues at EIFER and EDF SINETICS for valuable support during the implementation process as well as EDF for funding this research.

References

Autodesk Ecotect Analysis (2010) Training packages, demo toolkit, CFD analysis, air flow rate

Bahu JM, Koch A, Kremers E, Murshed SM (2014) Towards a 3D spatial Urban energy modelling approach. Int J 3-D Inf Model (IJ3DIM) 3(3):1–16

Barth A, Felt AP, Saxena P, Boodman A (2010) Protecting browsers from extension vulnerabilities. In: NDSS 2010 Feb

Bell D (2005) Software engineering for students. Pearson Education, New York

Brewer CA, Buttenfield BP (2007) Framing guidelines for multi-scale map design using databases at multiple resolutions. Cartography Geogr Inf Sci 34(1):3–15

Bostock M (2011) d3—A javascript visualization library for HTML and SVG. https://github.com/mbostock/d3. Accessed 15 May 2015

Butler H, Daly M, Doyle A, Gillies S, Schaub T, Schmidt C (2008) The GeoJSON format specification. Rapp Tech 67

Deslauriers M (2015) Drawing lines is hard. http://mattdesl.svbtle.com/drawing-lines-is-hard. Accessed 20 Dec 2015

Dix A, Finlay J, Abowd GD, Beale R (2004) Human-computer interaction. Pearson Prentice Hall, Harlow

Döllner J, Kolbe T, Liecke F, Sgouros T, Teichmann K (2006) The virtual 3D city model of Berlin—managing, integrating and communicating complex Urban information. In: 25th international symposium on Urban data management UDMS. Aalborg, Denmark

Giovannini L, Pezzi S, Di Staso U, Prandi F, de Amicis R (2014) Large-scale assessment and visualization of the energy performance of buildings with ecomaps-project SUNSHINE: smart Urban services for higher energy efficiency. DATA 170–177

Goetz M, Zipf A (2012) OpenStreetMap in 3D detailed insights on the current situation in germany. In: City, proceedings of the AGILE 2012 international conference on geographic information science. Avignon, pp 24–27

Goodchild MF (1991) The technological setting of GIS. In: Geographical information systems: principles and applications (1st ed), vol 1, pp 45–54
Grünbaum B, Shephard GC (1990) Rotation and winding numbers for planar polygons and curves. Trans Am Math Soc 322(1):169–187
Harrower M, Brewer CA (2003) ColorBrewer.org: an online tool for selecting colour schemes for maps. Cartographic J 40(1):27–37
Huber H, Schickler W, Hinz S, Baumgartner A (2003) Fusion of Lidar data and aerial imagery for automatic reconstruction of building surfaces. In: Proceedings of the 2nd GRSS/ISPRS joint workshop on data fusion and remote sensing over urban areas. Berlin, pp 82–86
Jenny B, Šavrič B, Liem J (2015) Real-time raster projection for web maps. Int J Digital Earth 1–15
Lavoué G, Chevalier L, Dupont F (2013) Streaming compressed 3D data on the web using JavaScript and WebGL. In: Proceedings of the 18th international conference on 3D web technology (pp 19–27). ACM
Ledermann F, Gartner G (2015) Mapmap. js: a data-driven web mapping API for thematic cartography. Rev Bras Cartografia 5:67/5
Liang YE (2010) JavaScript testing beginner's guide test and debug JavaScript the easy way. Packt Pub, Birmingham
McNamara R, McCormack J, Jouppi NP (2000) Prefiltered antialiased lines using half-plane distance functions. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on graphics hardware, pp 77–85
Ming W (2008) A 3D web GIS system based on VRML and X3D. In: Genetic and evolutionary computing, 2008. WGEC'08. Second international conference on (pp 197–200). IEEE
Mr. Doob (2010) three.js—JavaScript 3D library. three.js—JavaScript 3D library. http://threejs.org/. Accessed 14 June 2015
Neumann A, Winter AM (2001) Time for SVG—towards high quality interactive web-maps. Int Cartographic Assoc. (2001 Aug 6)
Nouvel R, Zirak M, Dastageeri H, Coors V, Eicker U (2014) Urban energy analysis based on 3D city model for national scale applications. In: IBPSA Germany conference, vol 8
Peterson MP (2015) Evaluating mapping APIs. In: Modern trends in cartography. Springer, pp 183–197
Raghothama J, Meijer S (2015) Gaming, urban planning and transportation design process. In: Planning support systems and smart cities. Springer, pp 297–312
Resch B, Wohlfahrt R, Wosniok C (2014) Web-based 4D visualization of marine geo-data using WebGL. Cartography Geogr Inf Sci 41(3):235–247
Saed S, Wendel J (2015) Estimating heating energy consumption and CO2 production—a novel modeling approach. In: Proceedings of the 14th international conference on computers in urban planning and urban management. Cambridge
Sandvik B (2013) Showing GPS tracks in 3D with three.js and d3.js. Thematic Mapping Blog. http://blog.thematicmapping.org/2013/11/showing-gps-tracks-in-3d-with-threejs.html. Accessed 11 May 2015
Slocum TA, McMaster RB, Kessler FC, Howard HH (2009) Thematic cartography and geovisualization. Prentice Hall, New Jersey
Simons A, Nichersu A (2014) Development of a CityGML infrastructure for the implementation of an energy demand method with different data sources. GIScience 2014. Vienena, Austria. (25th September 2014)
Suderaraman P (2013) Practical Ext JS 4, 1st edn. Apress, New York
Sutherland A (2012) asutherland/d3. GitHub. https://github.com/asutherland/d3-threeD. Accessed 7 Dec 2015
Soltani A, Canty S, Mayo Q, Thomas L, Hoofnagle CJ (2010) Flash cookies and privacy. In AAAI spring symposium: intelligent information privacy management, Vol 2010, pp 158–163
Thiele A, Wurth MM, Hinz S, Even M (2013) Extraction of building shape from TanDEM-X Data. International archives of the photogrammetry. Remote Sens Spat Inf Sci XL-1/W1:345–350

Wendel J, Nicerhsu A (2015) An open-source data infrastructure for storage, analysis and visualization of city energy geospatial data. Energy, science and technology 2015. Book of abstracts, EST, energy science technology, international conference & exhibition. Karlsruhe, Germany. (20–22 May 2015)

Wieland M, Nichersu A, Murshed S M, Wendel J (2015) Computing solar radiation on CityGML building data. In: 18th AGILE international conference on geographic informaton science 2015. Lisbon, Portugal. (9th June 2015)

Zipf A (2010) Heidelberg-3D. http://www.heidelberg-3d.de/index.en.html. Accessed 18 Dec 2015