TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

MASTERARBEIT

Implementing building visualization technology in SEMERGY web-based building performance optimization tool

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs

unter der Leitung von
Univ.-Prof. Dipl.-Ing. Dr. techn. Ardeshir Mahdavi
E 259/3 Abteilung für Bauphysik und Bauökologie
Institut für Architekturwissenschaften

eingereicht an der
Technischen Universität Wien
Fakultät für Architektur und Raumplanung

von
Amirali Sadeghi Khafri

Matrikelnr. 1128709
Sebastianplatz 5/4/9, 1030 Wien

TABLE OF CONTENTS

ABSTRACT

Building Information Model visualization capability is an informative and essential feature of building performance simulation tools. Presenting the result of a performance simulation routine on a building while providing the end user with real time visual feedback on building's geometrical properties, components and semantic information, prevents potential faults and inaccuracies in data input stage which may be caused by human error or malfunction of geometry import procedures. These errors will influence and falsify entire simulation processes and the optimization outcomes. This thesis project investigates and compares different aspects of current web-based visualization technologies, focusing on their implementation and adaptability. It then proposes a framework for mapping SBM, a recently developed building model of the SEMERGY project, to data format of the selected visualization technology, BIM Surfer. The SEMERGY project is a developing web-based tool aimed at comparative simulation and analysis of building design and retrofit options utilizing semantic web technologies. A serializer software solution is developed as the result of the investigations which can be implemented in the context of SEMERGY project.

Keywords

BIM, SEMERGY, 3D visualization, BIM Surfer, WebGL

KURTZFASSUNG

Dreidimensionale Visualisierungen von Gebäudedaten und Modellen sind ein informatives und mächtiges Hilfsmittel für Planer, Architekten, Facility Manager und Gebäudenutzer. Im Bereich der Building-Performance Simulation besitzen zwar viele Werkzeuge inzwischen überzeugende Eingabe-Interfaces, jedoch oftmals keine entsprechenden Visualisierungen. Dabei kann gerade bei der Simulation von physikalischen Prozessen im Gebäude zur Gebäudeperformance-Evaluierung eine solche (Echtzeit-)Visualisierung extrem nützlich sein: Die Visualisierung der Gebäudegeometrie, der Kompontenten und semantischen Eingabedaten macht ein Performancemodell virtuell angreifbar und zeigt potentielle Fehlerquellen und Ungenauigkeiten bei der Dateneingabe an (die in der Regel durch Nutzerverhalten oder durch Funktionsstörungen der Performancewerkzeuge verursacht werden). Solche Fehler sind in späteren Schritten in der Simulation nur schwer identifizierbar, sind aber – im schlechtesten Falle – im Stande die Aussage eines Gebäudeperformancemodells falsch und/oder unbrauchbar zu machen. Darüber hinaus können Sie bei einem Optimierungsvorgang falsche Resultate infolge der fehlerhaften Eingabedaten verursachen.

In dieser Masterarbeit wurden die verschiedenen Aspekte von aktuellen web-basierten Visualisierungstechnologien auf deren Verwendbarkeit und Anpassbarkeit – auch im Kontext der Building-Information-Modeling-Konzepte - untersucht und verglichen. Im Zuge dieser Untersuchung wurde „BIM Surfer“ ausgewählt, um ein Mapping eines neuentwickelten Gebäude-Datenmodells (Das „Semergy-Building-Model“ aus dem SEMERGY-Projekt) darauf durchzuführen. Bei SEMERGY handelt es sich um eine web-basierte Umgebung zur Gebäudeevaluierung und Optimierung. Die angestrebten Ergebnisse im Zuge dieser Master-Arbeit sind somit die Schaffung einer Schnittstelle zwischen dem SEMERGY-Projekt und der BIM-Surfer-Visualisierungsumgebung, sowie die Erstellung eines Software-Serialisierers für Objekte die in Semergy erstellt wurden.

Stichworte : BIM, SEMERGY, 3D visualisierung, BIM Surfer, WebGL

ACKNOWLEDGEMENTS

I hereby would like to express my sincere gratitude to my supervisor, Professor Dr. Ardeshir Mahdavi, for his precious advice, exemplary guidance, support and encouragement throughout the course of this project.

I would also like to thank the entire SEMERGY team, especially Ulrich Pont, Neda Ghiassi, Kristopher Hammerberg and Mahnameh Taheri. My special appreciation goes to Johannes Heurix for his cordial support, valuable information and wonderful collaboration.

Above all, I profoundly thank my family and friends, for their continuous support and encouragement without which this assignment would not be possible.

The SEMERGY project is funded under the FFG Research Studio Austrian Program (grant No. 832012) by the Austrian Federal Ministry of Economy, Family and Youth (BMWFJ). In addition to the aforementioned members, the SEMERGY team includes: F. Shayeganfar, S. Fenz, A. Anjomshoaa, A.M. Tjoa, D. Wolosiuk, T. Neubauer, C. Sustr, V. Jain, and A. Wurm.

LIST OF FIGURES

LIST OF TABLES

MOTIVATION

1.1 Significance of Visualization

Genius seems to consist merely in trueness of sight.
- Ralph Waldo Emerson (1840)

It is needless to say that the great majority of our understanding regarding our surrounding environment comes from the visual perception. Such external perceptual interaction is almost utterly tied with the interior mental actions in all aspects of human activities. This close relationship between mental activity and perception is the essence of our expanded intelligence. One of the most significant devices aiding us to gain such intelligence is "graphical representations". Graphics, which can be defined as "visual images" or "pictorial representations of data", serve two related but at the same time distinct functions. One is to be used as a tool for communicating the already existing idea or entity, whilst the second is to help creating and discovering the idea itself. Visualization or seeing, in a contemporary and technical context, can be described as the use of computer-supported, interactive, visual representations of data to amplify cognition (Card et al., 1999). Such cognition gives an insight on the subject of study and paves the way of the user to thoroughly explore, examine, understand and take further decisions regarding the represented entity. With the introduction and fast pace development of computer technologies, graphical mediums in modern day are hugely benefited by improved renderings, real time interaction and dynamic features while they offer relative affordability, availability and ease of publishing.

Communication is mentioned as one major role of visual representations however, it can be seen that representative models not only act as means of communication between people, but they also fulfill an auto-communicative function within the individuals mind (Mahdavi 2004). Such "internal dialogue" is a fundamental aspect in a creative design process in a thought provoking manner, leading to one's self-brainstorming to tackle design and exploration obstacles.

1.2 Architectural model visualization in performance simulation tools

Ultimate outcome of a real architectural project is a physical building while the final product of computer generated visualization is an image on a display monitor. Hence, it will be of little purpose to create abstract or realistic visualizations of buildings evoking particular responses from the user while the resulting actual buildings do not evoke similar response in real world (Aish 1986). It has been studied that two dimensional plan drawings were considered less satisfactory as a representation method whereas simple three dimensional models were more effective. Such models offer the facility for users to unambiguously achieve an understanding of the size and shape of the exterior overall form along with the interior spaces by changing eye and viewpoint positions. This, so to say, perceptual interaction with the model, helps users to achieve a compatibility between visual and positional cues which at end will occur in the perception of the real building (Sorte 1975). Interaction between the designer and graphical physical descriptions such as drawings, computer generated visualizations or photographic images, will therefore become a necessary part of an effective design process (Brown 2003).

In the field of building performance simulation, conveying simulated entity's design information by means of visualization and graphical feedback which are based on direct or indirect user input, greatly helps preventing the occurrence of potential faults and inaccuracies in definition of geometries and construction components as well as semantic data. The consequence of an erroneous procedure, which may be caused by human error or malfunction of model import routines, is the falsification of entire simulation process and unreliable optimization results. As a precaution to such vulnerabilities, software developers started incorporating Graphical User Interfaces (GUI) for the already established and common simulation engines such as EnergyPlus and DOE-2 (EERE 2014, DOE2 2014). This has facilitated users and designers to overcome the laborious and error prone task of manual data input in spreadsheet-like format of the calculation engines. These tools provide greater flexibility in data input

and validation. GUI environments such as OpenStudio and DesignBuilder for EnergyPlus offer realtime, interactive and bidirectional visualization of the building geometry in addition to construction element properties and relative data. The building is represented as a three dimensional geometric model, allowing many degrees of freedom for the user to explore, edit and validate the integrity of the design.

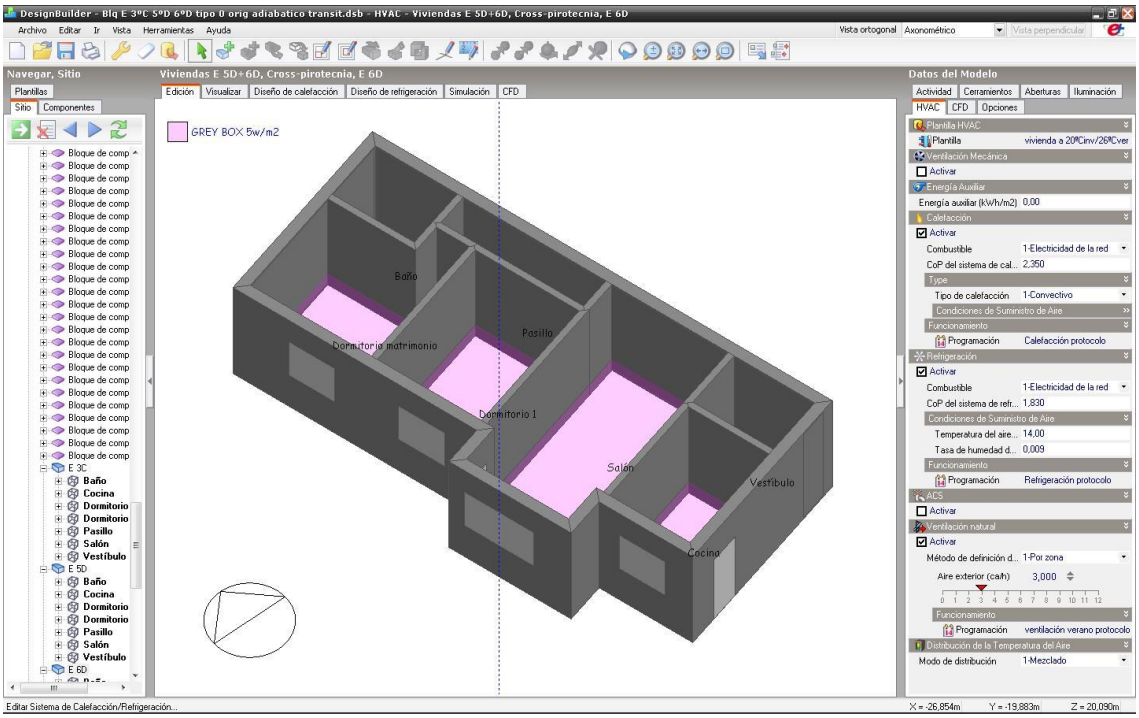


Figure 1: 3D building model represented in DesignBuilder GUI for Energy Plus (Martes 2013)

Although such graphical user interfaces were designed to reduce usage complexity, in some cases they may seem more complicated, especially when exploring specific building technologies and heat transfer phenomena (Cymru 2013). Even though trained and professional users are accordingly more confident in working with such tools and their sophisticated CAAD environments, the ordinary in-experienced user does not necessarily have the familiarity and awareness to spot potential faults at first glance in the designed model or by inspecting the output of simulation results. Needless to mention is the amount of time and challenge to familiarize one's self with the design

environment and workflow. For instance, creating the geometrical 3D model of the building requires manual and accurate reconstruction of the subject building by using a variety of different building elements (i.e. walls, slabs, windows, etc.) and defining their semantic properties. Although there is the possibility of geometry import through BIM systems, major interoperability issues can hinder the efficient exchange of data between different design and simulation platforms (Ghiassi 2013). Individual BIM software developers have developed their own solutions for interoperability - mostly based on the IFC format, yet the interface issues does not stop on a technical level only, but is also present in the processes and organization of the company in which the BIM software is being implemented (Kiesel et al., 2013).

As one of its main approaches to overcome the inefficiencies and complexities of current building performance simulation tools in data acquisition phase, the SEMERGY project has introduced a simplified two dimensional design environment. This system allows users to draw buildings in plan form on a grid, using two-dimensional straight linear segments representing conventional three-dimensional building elements such as walls, doors and windows.

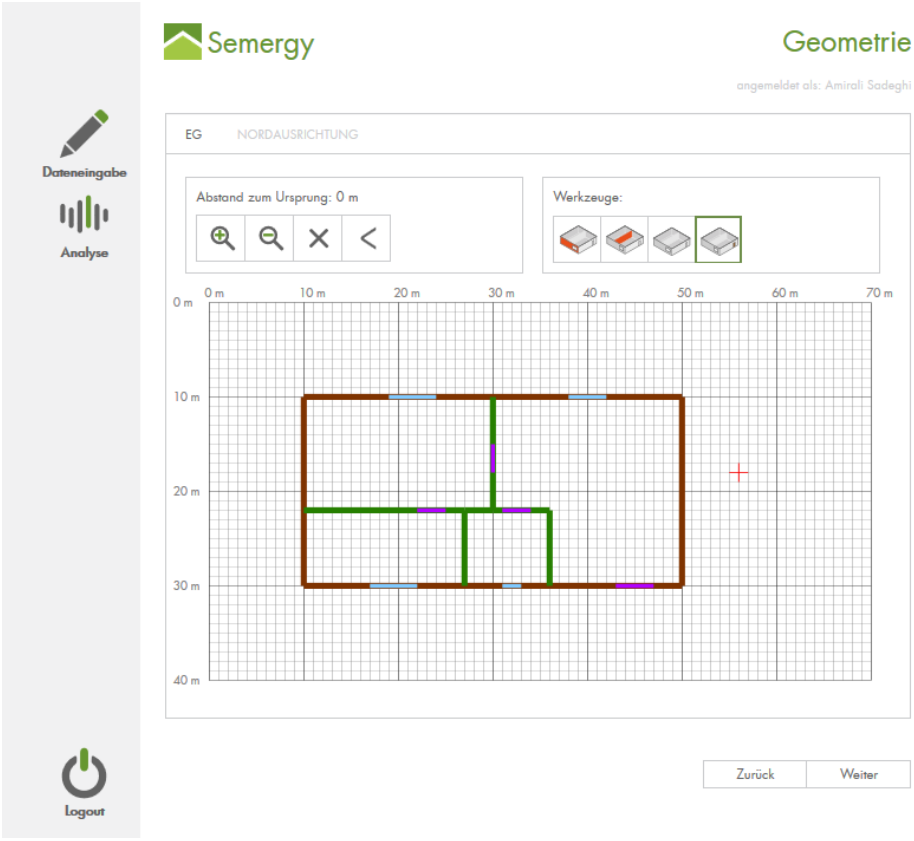


Figure 2: Drawing environment in SEMERGY

Although this tool facilitates a relatively easier design method for novice users, yet it lacks a satisfactory and efficient visualization of the building model as formerly stated through Aish's argument. This raises the question of whether the user understood the relation between the plan drawings and the final three dimensional objects or spaces which are implied by the plan and if they appreciate the three dimensional attributes of the building they are designing.

The reasonable solution for these questions would be a simplified, yet not too abstract three dimensional representation model which is understandable and sufficiently coherent for the end user, with the necessary degrees of freedom such as variable viewing angles, visibility options and component selections, which provides a proof of concept and design validation.

2 BACKGROUND

2.1 SEMERGY introduction and approach

The SEMERGY project is considered as a developing tool for comparative simulation and analysis of building design and retrofit options. Utilizing semantic web technologies, this web-based tool aims to bridge the gap between users' abstract representations of building components and the real world products with their complex specifications. The general workflow of this tool starts by providing initial project information. Such information include geographical location, building age, simplified geometrical data, construction components and their properties, energy sources and consumption method and information on the available budget for performance optimization. Further on, this data, along with additional information about product prices, microclimatic boundary conditions, legal constraints and possible applicable subsidies which are extracted from the web, is used to run a series of semantic permutations based on the initial case and the constraints which user has already defined. After creation of a set of feasible design alternatives, they are being evaluated

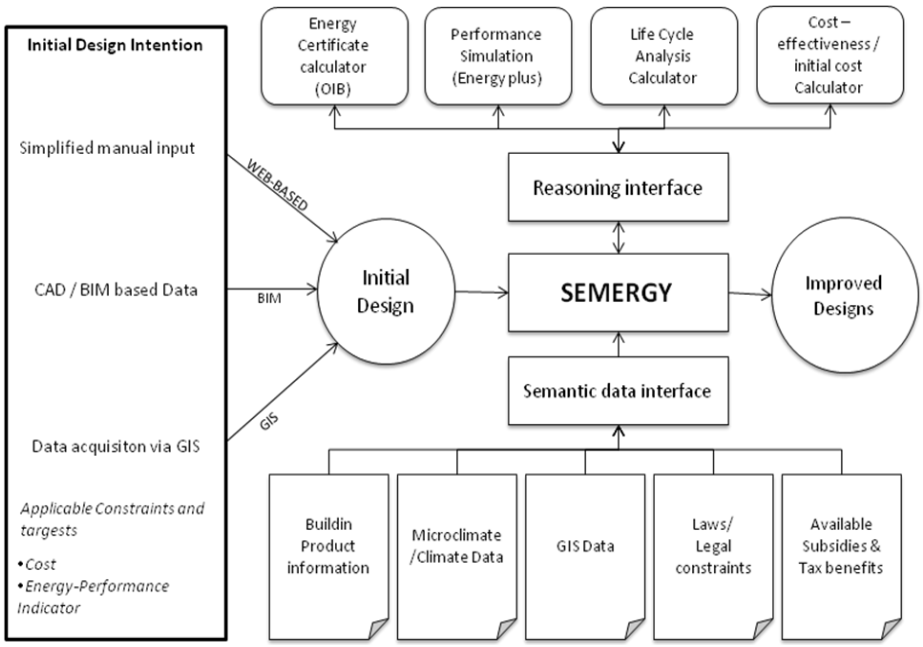


Figure 3: SEMERGY system design

in order to generate energy certificates and/or life cycle analysis and additionally to be fed into light or performance simulation applications. Figure 1 schematically shows the design of SEMERGY performance analysis and evaluation environment.

2.2 SEMERGY Building Model

Despite being comprehensive, common building information models such as Industry Foundation Class, IFC (Building Smart 2013) and the Green Building XML, (gbXML 2013) carry large amounts of variables in a complex data structure or lack interoperability features. For instance the language definition of IFC version 2x3TC1 includes 327 data types, 653 entity definitions and 317 property sets (Stee et al., 2012). Thus performance assessment and simulation procedures of such models using SEMERGY tool are not practically feasible without considerable amount of overwork (Ghiassi 2013). Furthermore, geometrical representation of the building differs according to the CAD programs and drawing methods used to generate the model which also makes the evaluation process complicated. Since spaces are defined as areas bounded by three dimensional architectural elements in IFC, the required definition of a space for performance evaluations will not be readily achievable due to possible gap generation between 3D elements. On the other hand, while gbXML seems to resolve the space boundary problem, it lacks the ability of modification and extensibility. Upon addition of new attributes to the gbXML schema the file becomes invalid and unrecognizable by applications.

To overcome the complexity and performance issues of current models and facilitate the implementation of semantic web technologies in order to create a web-based performance evaluation environment, SEMERGY Building Model (SBM) has been developed, by providing only necessary variables and data for normative procedures and performance simulations and eliminating non-vital information. Shared Object Model (SOM), which is an object-based building model developed for the SEMPER project, formed the basis for creation of the SBM (Mahdavi 2000). Shared Object Model

though, does not represent a building in its entirety since it was developed to meet the informational requirements of particular analytical applications, however the representation of buildings through spaces (similar to gbXML and unlike IFC) as main composing elements and its clear data structure with a performance oriented approach, made it the perfect foundation for SBM.

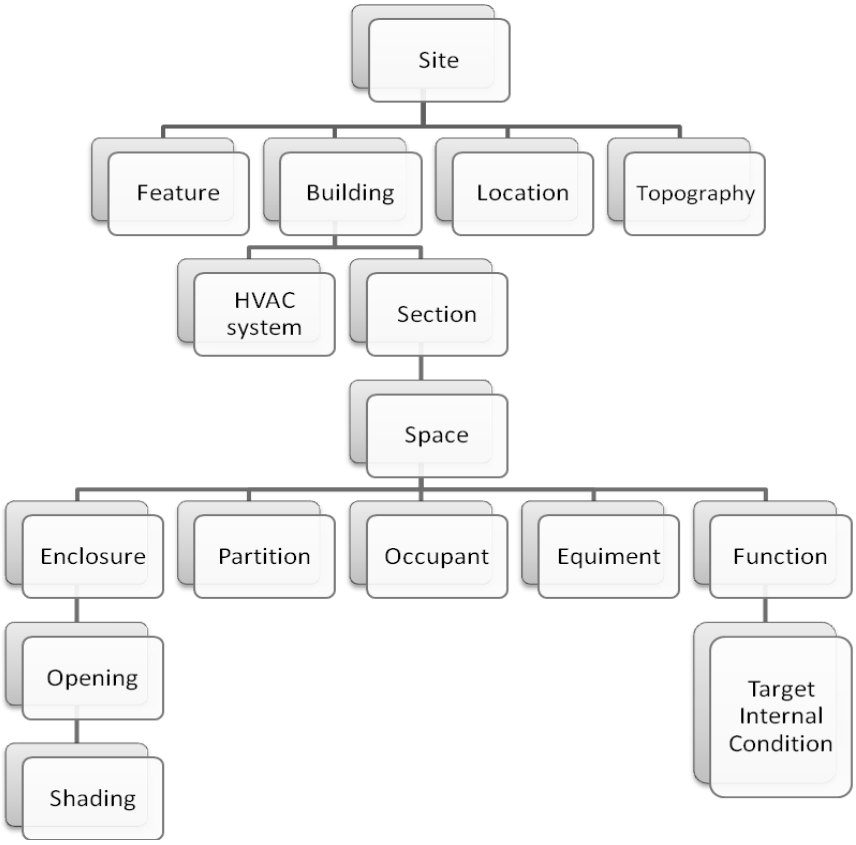


Figure 4: SEMERGY Building Model in general

Figure 2 shows the hierarchical structure of SBM. Detailed insight on main objects in the data model which are required for the 3D geometrical visualization will be discussed in chapter 3.2.

2.3 Three-dimensional web graphics

Fundamentally, any computer graphic technology which facilitates displaying and interacting with three-dimensional content in World Wide Web (WWW) environment is commonly referred to as Web3D. Many formats and platforms have been developed for this purpose and are distinguished by features such as:

- **Development and operation simplicity**: Straightforward software development toolkit which is well documented and ease of future installation and usage
- **Compatibility**: Operating system independence and cross-platform functionality
- **Quality and Performance**: Adequate render quality and frame rate while keeping resources usage at minimum through efficient algorithms.
- **Interactivity**: Providing necessary basis in development phase and offering features for end-user to incorporate various levels of interactivity.
- **Standardization**: Ability to function alongside and interact seamlessly with established web standards such as HTML, HTML5 and network protocols such as HTTP.

Amongst the most prominent of these technologies are VRML, X3D, Java3D, Shockwave and WebGL. The first ISO standard for Web3D was the Virtual Reality Modeling Language (VRML 97) which was introduced by the Web3D Consortium towards collective efforts in development of three-dimensional web graphics technology. Although it gained much attention in educational and experimental divisions at the time, it soon was succeeded by X3D which greatly improved the performance and functionality by providing an XML-based data structure and enhanced communication features. By the introduction of OpenGL (Open Graphics Library) which is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D computer graphics locally, developers started utilizing the power of Graphics Processing Unit (GPU) to perform hardware acceleration renderings. Soon

the technology was adopted by the consortium Khronos Group and lead to development of WebGL (Web Graphics Library).

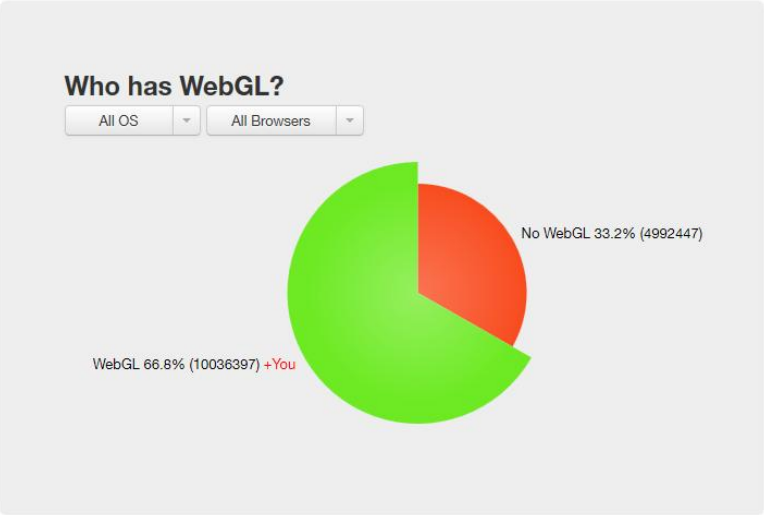


Figure 5: WebGL usage statistics (http://webglstats.com)

Thanks to its simplicity of application (trough JavaScript API), greatly improved performance (hardware-accelerated) and extensive documentation and developing community, WebGL is picked up by many developers and end-users as the preferred 3D API. One the most significant advantages of WebGL over other rivals is that nearly all other technologies need the installation of a viewer plug-in or add-on software (i.e. Adobe Flash player, VRML and Unity3D viewers, JAVA JRE, etc.) which makes browser independency and cross-platform interoperability an issue, whereas WebGL is completely integrated into web standards of most modern browsers through JavaScript and HTML5. As of December 2013, 66.6 percent of mobile and desktop visitors of 500+ participating websites with interactive 3D content through WebGL had WebGL capabilities on their web browsers.

To utilize the powers of WebGL and ease its usage in general scenarios, a number of JavaScript libraries have been developed to provide higher level functionalities in creating 3D graphics applications. For example WebGLU, which is the WebGL correspondent of GLU, provides wrappings for placing the camera in the scene or for creating simple geometric primitives while other libraries such as GLGE and SceneJS

use WebGL for implementing a scene graph based rendering and animation (Di Benedetto, 2010).

2.4 Building Information Model visualization on the web

Thanks to its non-proprietary and open standard architecture and vendor-specific independency, IFC (Industry Foundation Classes) has been used as the most common building information model data exchange format in architecture, engineering and construction (AEC) industry and has been supported and implemented by major CAD vendors as a standard exchange format. Towards this end, special online systems such as IFC Web Server (IFC Web Server 2013) and Building Information Modelserver (BIM Server 2013) have been developed to facilitate project data centralization, storage, visualization, query, and export capabilities. Moreover, platforms such as BIM Server offer a collaborative environment through which multiple users have the ability to work simultaneously on various parts of a project while it's being updated on the fly.

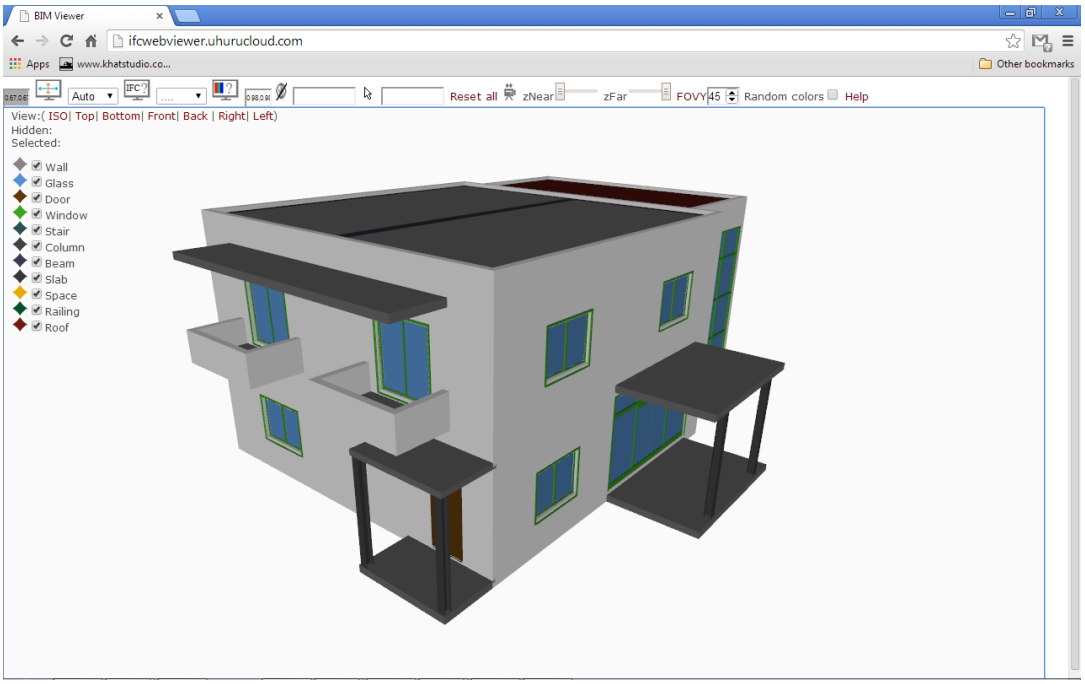


Figure 6: IFC Web Viewer, default viewer for IFC Web Server.

All of the aforementioned online systems along with other special purpose tools such as IFC WebGL Viewer (IFC Tools Project 2013) benefit from the WebGL engine to visualize 3D model data in the web browser. While producing a faithful 3D representation of the building, they also offer features such as:

- Simple layer management (hide, show, color change)
- Viewpoint alteration (mouse & keyboard navigation)
- Creating simple sections (sectional cuts parallel to the camera)
- Hiding/selecting 3D objects by name/global Id

The typical workflow of the visualization process, in case of the IFC WebServer for instance, starts with the conversion of the geometric representation of IFC objects into the open and standard 3D file format COLLADA (collaborative design activity) by the IFC WebServer software. The resulted COLLADA file is then parsed to IFC Web Viewer tool which renders the scene in the web browser with help of the SpiderGL (Di Benedetto 2010) JavaScript library. [Figure 7]

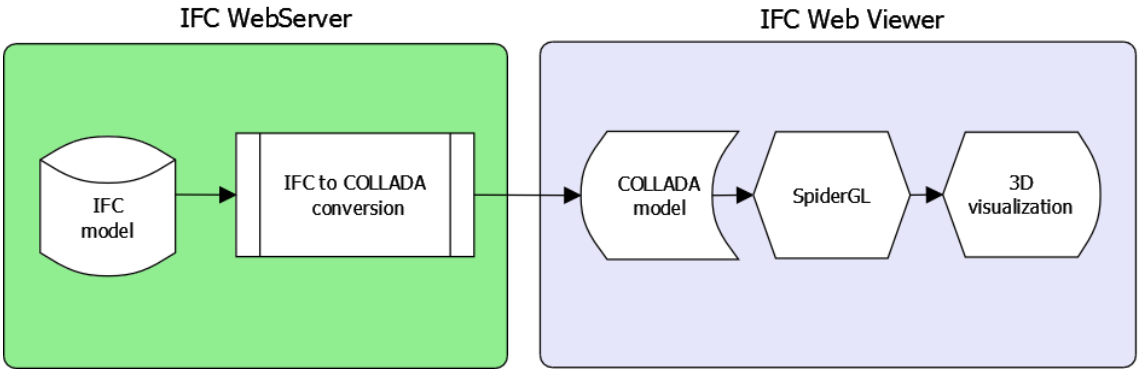


Figure 7: IFC visualization workflow in IFC WebServer with IFC Web Viewer

2.5 BIM Surfer

BIM Surfer is the open source web-based WebGL 3D viewer for the IFC/BIM model which is created as an initiative from the open source BIM collective (BIM Surfer 2013). Recent efforts show that Industry Foundation Class (IFC) building information models which are extended for use in BIM Surfer demonstrate more appropriate performance results than others in the context of building performance simulation (Zach, 2012). Due to its proven functionality and established adaptation in building management and data visualization tools such as MOST (Zach, 2012), BIM Surfer is chosen as the preferred visualization environment for SEMERGY building models. In addition to typical visualization of a 3D model of the building, BIM Surfer provides tools to manage the visibility of layers (building elements such as different types of walls, slabs, doors and windows) in single or group modes as well as inspecting the properties of particular objects. The type of data shown in properties section and their availability may vary according to the source application which the model is exported from. A unique feature to expose building stories progressively is also provided to give insight on the spatial arrangements of each floor.

Figure 8: BIM Surfer visualizing a typical building model

Much alike IFC Web Viewer, BIM Surfer uses a similar approach to visualize the 3D building model; however the process differs in the generated target file format and the JavaScript library which is utilizing WebGL core functionalities. BIM Surfer only identifies input data in JavaScript Object Notation (JSON). JSON is an open standard format which uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application as an alternative to XML. Further in the visualization process, BIM Surfer JSON files, which are typically created via the built-in export tool of BIM Server, are parsed to and rendered by SceneJS engine. SceneJS is an open source JavaScript library based on WebGL which benefits from JSON application programming interface. This enables 3D scenes to be created in a light and declarative format which can be generated, filtered, queried, stored and transported easily (Kay, 2013).

2.6 SceneJS schema and data structure

SceneJS incorporates a scene graph to store scene information. A scene graph in principle is a data structure in form of a graph or tree which includes logical and spatial representations of a graphical scene [Figure 9].

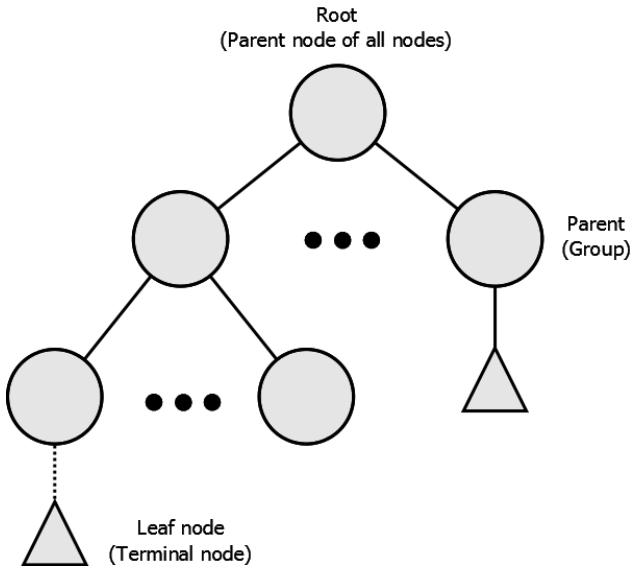


Figure 9: Scene graph general structure

A scene graph is a collection of nodes (parents) and leaves (childs). An advantage of such structure is efficiency in performing calculations in a hierarchical organization. Generally an operation which is a performed on a node (group) will propogate its effects to all its children. One particular usage of this feature is in building modeling where constructing elements are grouped based on their type, function and level according to building hierarchy. A scene graph also facilitates the application of particular visual filters at render time with less effort. These may include operations such as color/transparency changes.

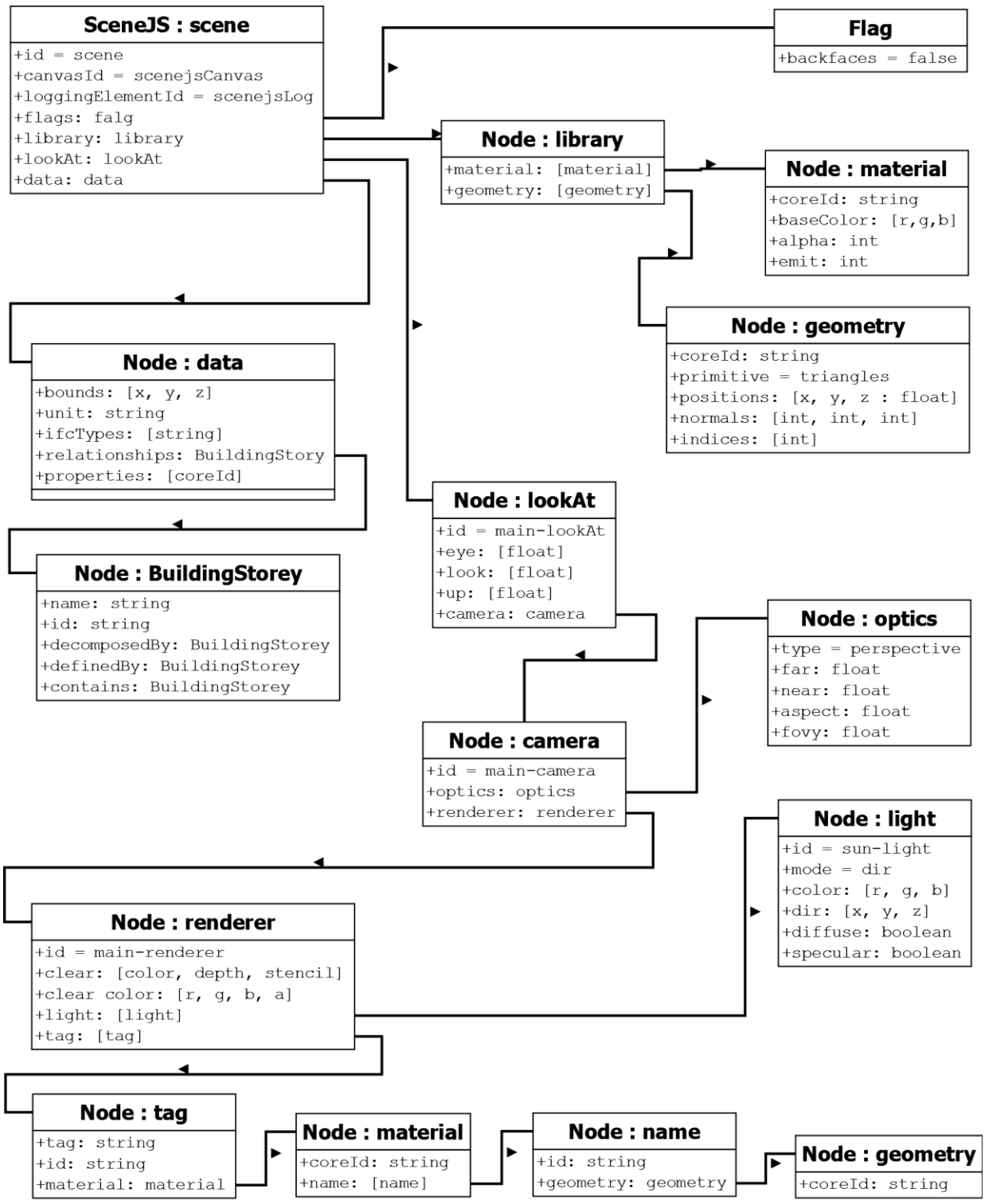


Figure 10: SceneJS class diagram for BIM Surfer JSON schema

As shown in Figure 10, BIM Surfer uses a modified adaptation of the generic SceneJS JSON tree schema (Scene graph) to incorporate necessary data required for building layer/object management and storey exposition in addition to standard geometrical information needed for visualization. A brief description of major SceneJS nodes is shown in Table 1.

Table 1: SceneJS node description for BIM Surfer JSON schema

Node name	*Description*
Scene	Root node of the scene graph tree, containing header settings and main superclasses
Library	Container class for individual 3D objects and materials. Objects and materials are stored in an array.
Material	Objects' material settings such as color, opacity and self-illumination
Geometry	Objects' geometrical information, containing primitive face shape type, faces coordinates, face normals and face indices
LookAt & Optics	Scene camera settings, such as position of camera, target, field of view, up axis, etc.
Light	Scene light settings, including color, intensity, direction, etc.
Data	Container class for ifcTypes (construction categories), object relationships and properties and additional scene settings such as scale unit and scene boundaries
BuildingStorey	Superclass containing building elements' relationships tree required for expose functionality.
Tag	Container class for material assigning for individual objects

Since the main objective of this project is to visually simulate the building geometry in 3D, it is relevant to discuss how SceneJS handles the instantiation and structuring of geometrical information. Figure 11 shows a sample cube to be generated and stored. In order to properly display and shade the 3D object, SceneJS demands values for three

parameters: positions, normals and indices. Since the primitive type of the geometry node is “triangles”, the object has to be decomposed into its constructing triangular faces (F_i). Having 6 sides (S_i), the sample cube will result in 12 triangle faces (F_0, F_1, … , F_{11}):

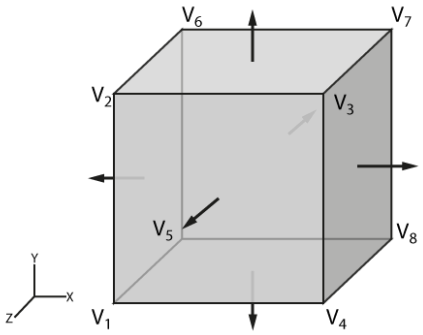


Figure 11: Sample cube geometry with labeled vertices

$$S_1 = \{F_0, F_1\} \qquad F_0 = \{V_1, V_2, V_3\}\ ,\ F_1 = \{\ V_3, V_4, V_1\ \}$$
$$S_2 = \{F_2, F_3\} \qquad F_2 = \{V_2, V_6, V_7\}\ ,\ F_3 = \{\ V_7, V_3, V_2\ \}$$
$$\ldots$$
$$S_6 = \{F_{10}, F_{11}\} \qquad F_{10} = \{V_4, V_3, V_7\}\ ,\ F_{11} = \{\ V_7, V_8, V_4\ \}$$

“Positions” array will thus include ordered list of coordinates of all vertices for every triangle face ($V_i.c$ = [$V_i.x$, $V_i.y$, $V_i.z$], coordinate triplet for vertex i):

$$positions = \{\underbrace{V_1.c, V_2.c, V_3.c}_{F_0}, \underbrace{V_3.c, V_4.c, V_1.c}_{F_1}, \underbrace{V_2.c, V_6.c, V_7.c}_{F_2}, \ldots , \underbrace{V_7.c, V_8.c, V_4.c}_{F_{11}}\ \}$$

“Normals” array includes vertex normal value of each vertex. ($V_i.N$ = [$V_i.N.x$, $V_i.N.y$, $V_i.N.z$] , normal vector triplet for vertex i):

$$normals = \{\underbrace{V_1.N, V_2.N, V_3.N}_{F_0}, \underbrace{V_3.N, V_4.N, V_1.N}_{F_1}, \underbrace{V_2.N, V_6.N, V_7.N}_{F_2}, \ldots , \underbrace{V_7.N, V_8.N, V_4.N}_{F_{11}}\ \}$$

“Indices” array includes indices which organize positions and normals into geometric primitives in accordance with the primitive type set earlier (triangle). In this case each triangle face is identified by a set of three indices corresponding to each ordered triplet value in “positions” and “normals” arrays (vertex index).

$$Indices = \{ \underbrace{1, 2, 3,}_{F_0} \underbrace{3, 4, 1,}_{F_1} \underbrace{2, 6, 7,}_{F_2} \underbrace{7, 3, 2,}_{F_3} \dots , \underbrace{4, 3, 7,}_{F_{10}} \underbrace{7, 8, 4}_{F_{11}} \}$$

3 PROBLEM DESCRIPTION

3.1 Overview

Since SEMERGY project incorporates the newly developed SEMERGY Building Model (SBM) which is a specifically tailored comprehensive object-based building representation model with a particular data structure, the conventional procedures of serialization of IFC model data to a JSON standard, suitable for visualization in BIM Surfer, is not applicable anymore and a custom data translation procedure to the target format needs to be developed.

Developing such software solution has to be able to address two major obstacles towards this approach. First and foremost is the discrepancy of data containers between SEMERGY output data model and BIM Surfer input data format, meaning that SEMERGY is not capable of communicating with BIM Surfer on its own and the result of the former is unidentifiable to the latter. This problem has to be dealt with by performing a data transformation process to generate valid target data required for visualization. The other concern is the issue of three dimensional mesh generations and modification of original planar surfaces. Geometrical operations, such as triangulations and extrusions are used to create the accurate mesh representations in this stage. Furthermore, the present graphical user interface of BIM Surfer is improved to incorporate functions and information panels with regards to the requirements of a performance simulation tool.

3.2 SBM structure and data organization

SEMERGY building model stores all the physical and operational data and calculation parameters in a hierarchical structure. This information is categorized into four major sections: 1- Physical Data: Geometries, 2- Physical Data: Semantics, 3- Operational Data, 4- Calculation Parameters. Each group contains various classes and attributes representing building parameters and properties.

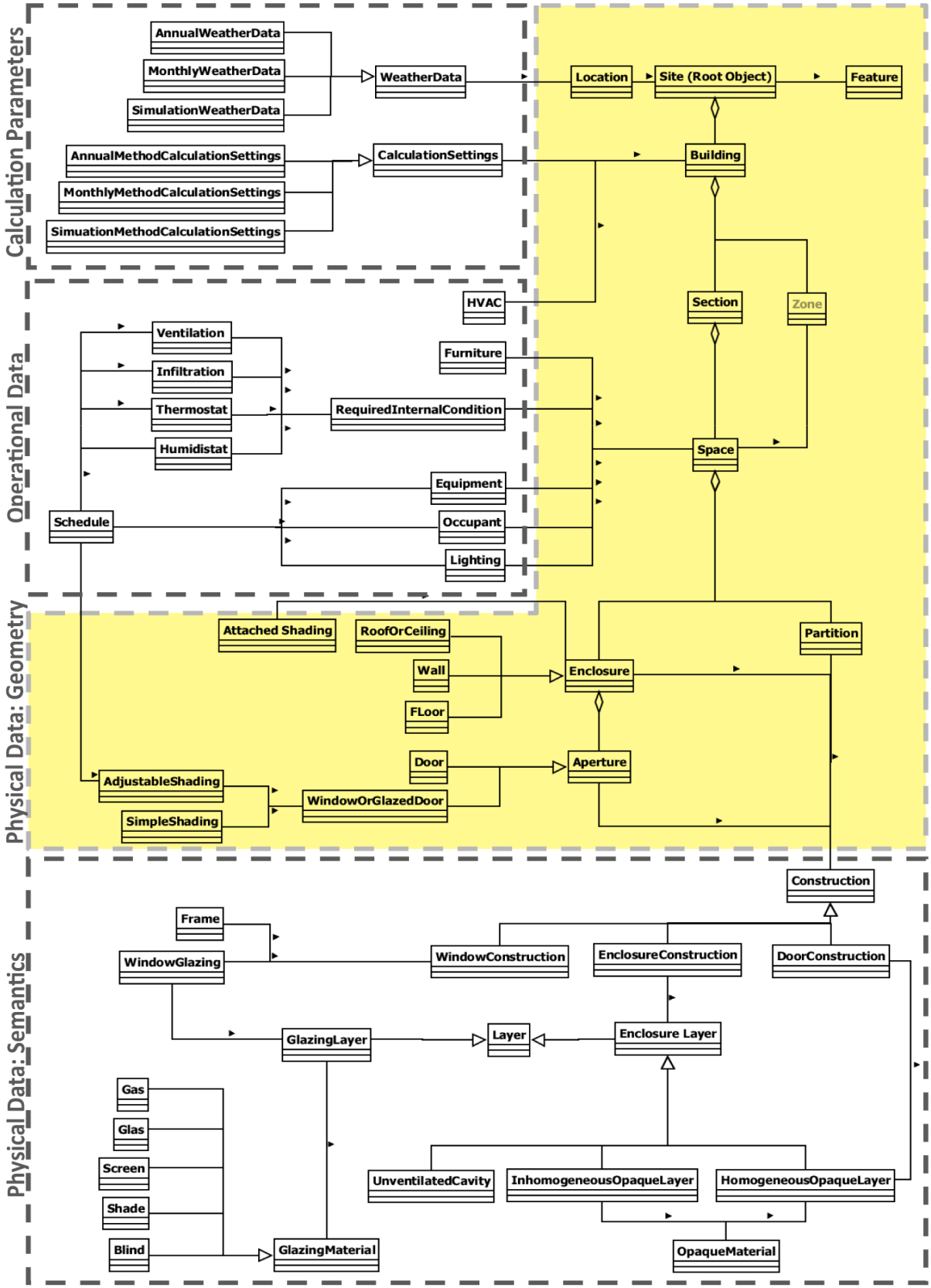


Figure 12: Semergy Building Model schema

As apparent in Figure 12, the SBM XML scheme includes many elements which are not essential to perform the geometrical model visualization. The highlighted area, Physical data: Geometry group, along with “Construction” class from Semantics group, contains all the information required for generating 3D meshes. A detailed description of each of the related classes follows with regards to Figure 13.

3.2.1 Building and section

“Building” is the main logical unit containing general information mostly required for energy calculations such as construction type and year, usage and heated volumes. It decomposes to “Sections” which are grouping of spaces that are on the same level and have identical elevation. The required three dimensional model for normative calculations is generated based on ceiling height property and provided floor plan, both as user input in GUI. Each section decomposes to one or more “Spaces”.

3.2.2 Space, Enclosure, Aperture

“Space” is a volume bounded by a series of “Enclosures” forming a polyhedron. In terms of performance, a space maintains a single thermal zone all over.

“Enclosures” are physical boundaries of space which can be any of the three types of “wall”, “roof/ceiling” or “floor”. They are represented by their four bounding vertex coordinates. In terms of adjacency to other spaces, the adjacency property may include "Outside", for components adjacent to outside air, "Ground" for earth adjacent enclosure elements, "Adiabatic", for enclosures, through which no heat transfer occurs (e.g., boundaries of the building to neighbor buildings) and "surface", for enclosures separating two spaces (e.g., walls separating two rooms). Such enclosures share exact vertex coordinate values but in reverse order which results in surface normals of opposite directions. In this case the “adjacencyObject” property will be populated with the related element.

“Aperture” is any opening of the enclosure which can be of type “door” or “window/glazeddoor”. Similar to enclosures, apertures are also defined by their bounding vertex coordinates.

3.2.3 Construction

"Construction" is layered assembly of material applicable to enclosures and apertures. The overall thickness of an enclosure/aperture as well as total resistance is defined in its corresponding construction class.

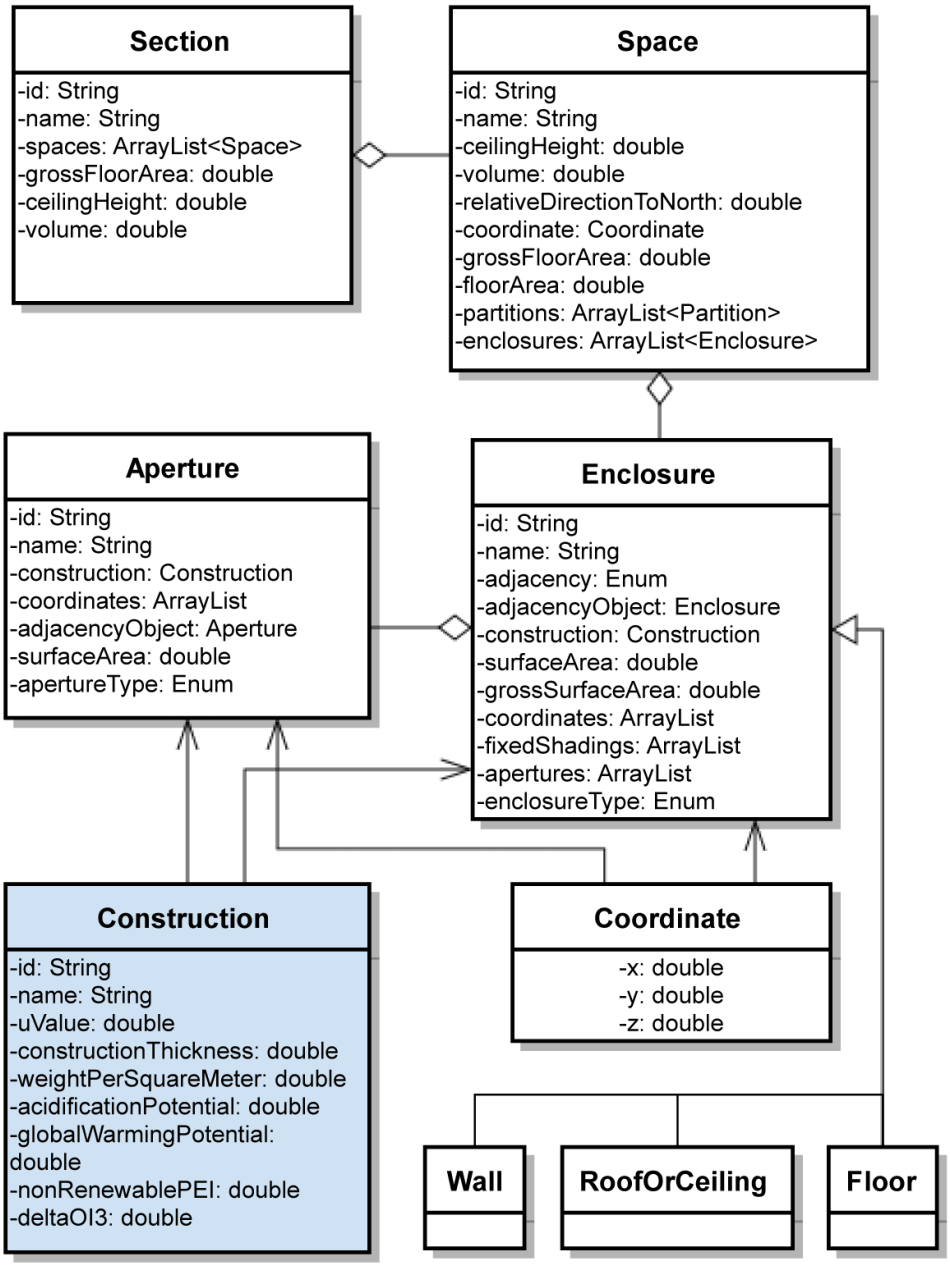


Figure 13: SBM building components' classes

3.3 Data transformation

As mentioned earlier, BIM Surfer employs SceneJS as its rendering engine which subsequently works based on input scene graph data in JSON format. Although XML and JSON are very similar in terms of data structure, properties and function in general, however the special categorization of data required for 3D visualization with SceneJS demands special translation of source XML into destination JSON format. In the field of data storage and warehousing, such operation on data structures is generally referred to as “data transformation”. In the context of this project, data transformation is comprised of the following major tasks:

1) **Data mapping**: Identification of element relationships between source (SBM XML) and destination (BIM Surfer JSON) formats, which demands the construction of a conversion (mapping) table.
2) **Data processing**: Extraction, refinement, transformation and classification of related data elements.
3) **Data serialization**: Preparation and packaging of the processed data for output format.

3.4 Generating precise 3D volumes

The architectural model in real world uses solid geometries which can be described by vertex coordinates or dimensions in three axes, since a faithful physical representation of the building is important. However the performance simulation model is only interested in the wall’s thermal coefficient and not its thickness, which results in usage of “centerline” simplified geometries (Stee et al. 2012). As seen in Figure 14, evidently such simplification in representation influences the volumes and useful areas of spaces as the thicknesses of building components are ignored (Ghiassi 2013).

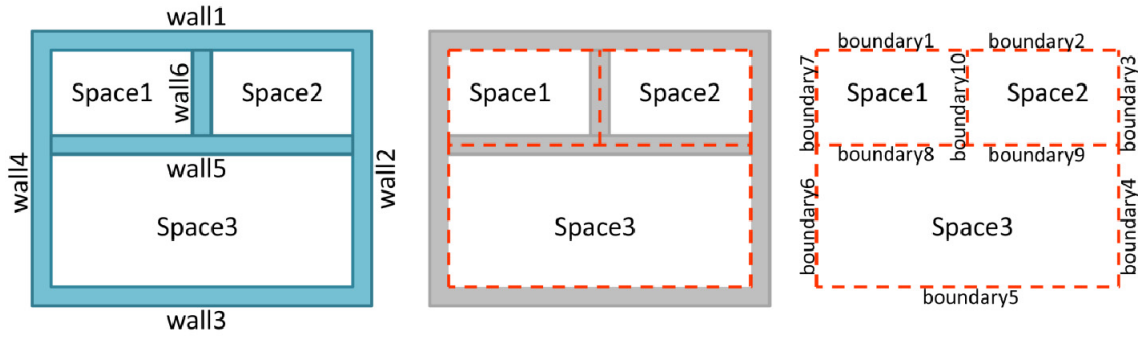


Figure 14: Difference between architectural walls and performance simulation model's space boundaries (Ghiassi, 2013)

Similar to gbXML (Green Building XML), SBM data models incorporate the same concept. Building information is stored at a different abstraction level than an architectural model with regards to the ultimate function which is performance simulation and calculations. To achieve a faithful representation of the building, three dimensional objects (meshes) have to be reconstructed out of existing planar surfaces. This task which is essentially the reverse procedure of simplifying the architectural model, demands a process of creating more from less, in terms of information.

3.4.1 General approach towards building elements

Geometrically speaking, in SBM, building elements are simplified down to their planar two dimensional constructing polygons. This results in a surface represented by an array of four coordinates in three dimensional space, disregarding object's thickness. Thus, positioning of individual surfaces is based on inner dimensions of the space which is defined by its enclosing elements. Figure 15 shows the actual three dimensional representation of an SBM model prior to processing for visualization.

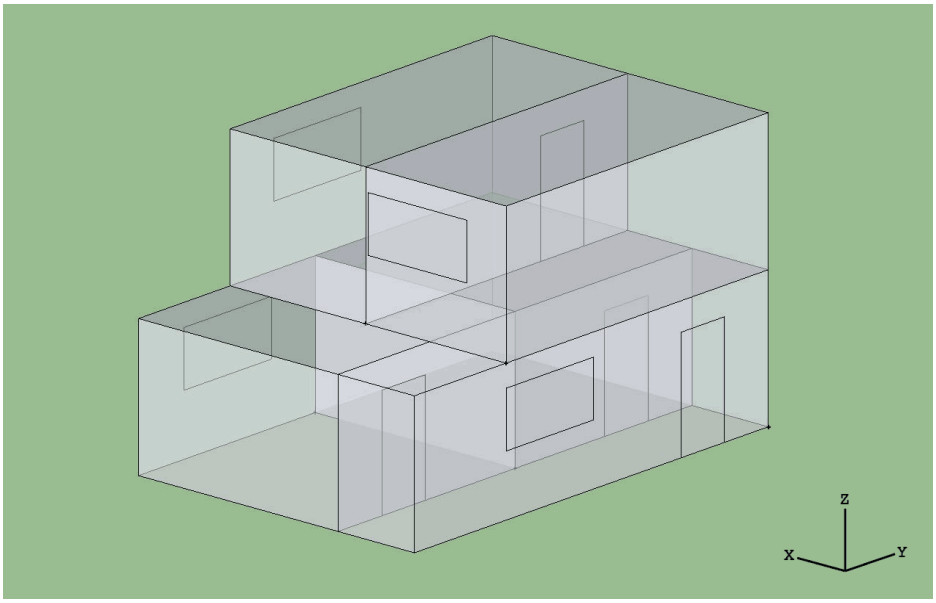


Figure 15: SBM unprocessed 3D geometry (IDF output in Sketchup)

The constructing polygon of each building element defines only the outer boundaries of that element in a plane and lacks the coordinates of the openings. The openings' boundaries are stored in apertures class as a child element which is used further on in performing a constrained Delaunay triangulation to achieve polygons with holes.

Once the overall accurate 2D polygonal representation of the building element is achieved, it needs to be extruded in the correct direction according to its specified construction thickness. Extrusion and prospective extension of faces will produce overlapping of meshes, mainly at wall corners and slab edges, which has to be handled by proper chamfering and vertex displacement [Figure 16]. Another issue which has to be taken care of after extrusion is the repositioning of horizontal elements (slabs). Since the element's thickness is being considered in the generation of new vertices, the additional value of thickness has to be accounted for in repositioning of new and current vertices.

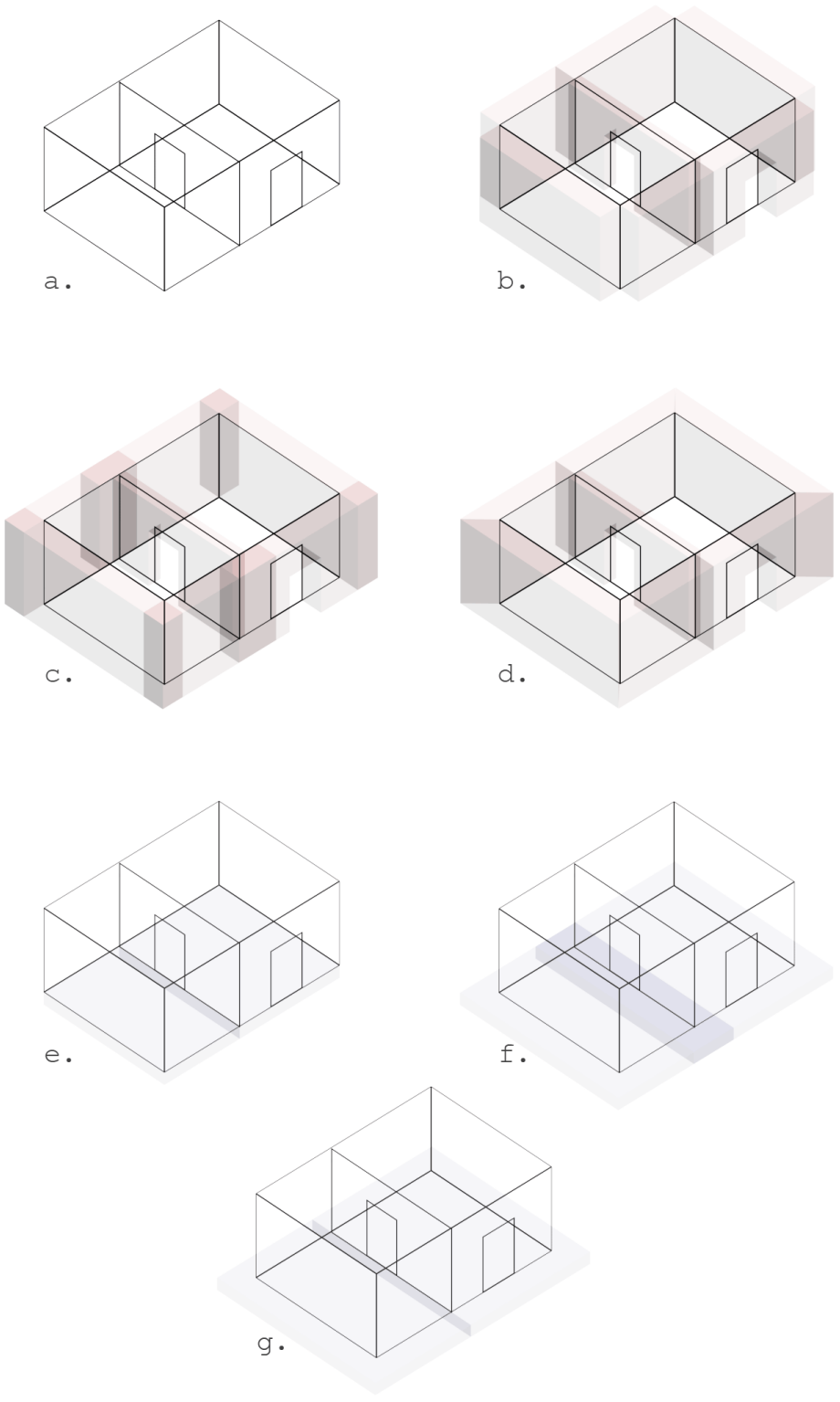


Figure 16: 3D mesh generation process: A) Tessellated planes ready for extrusion. B) Extrusion of walls based on surface normal and construction thickness. C) Extension of walls to fill empty spaces at corners which result in mesh overlaps. D) Chamfering and reposition of problematic vertices. E,F,G) similar procedure for horizontal components.

3.4.2 Roof forms

Although creation of accurate 3D meshes to some extents is a rather straightforward task of extruding planar surfaces, but on certain cases the result may not be as exact as the real world scenario. This is due to diversity of possibilities in generation of the 3D form and arrangement of spatial objects in certain conditions which cannot be determined by the limited information provided in source SBM data. An example of such case is roof generation.

Automatic roof reconstruction from sparse input data has been an important aspect in the field of remote sensing and photogrammetric research for the last two decades and many sophisticated algorithmic methods have been developed to tackle the issue (Haala 2010). As apparent in Figure 17 the number of possibilities for arrangement of symmetrical sloped wings of the roof is 2^n in total, where n is the number of rectangular sections the building perimeter is subdivided to.

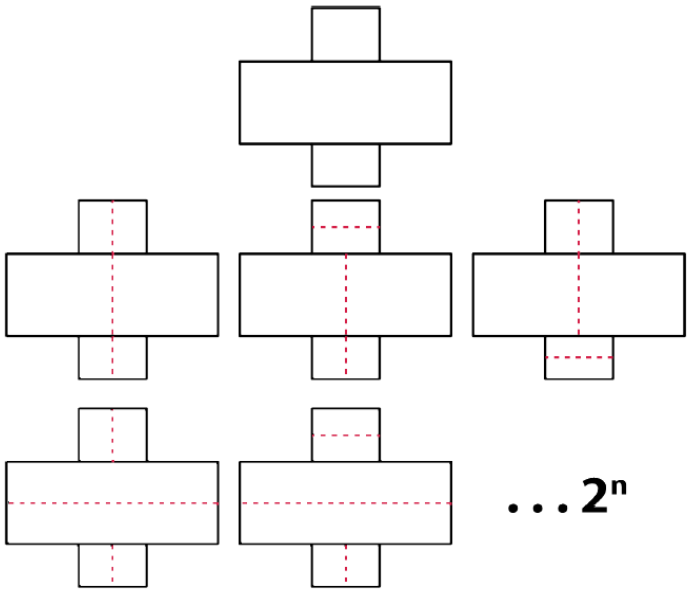


Figure 17 : Roof form decomposition possibilities (K. Hammerberg 2013)

Since SEMERGY performance calculations are made prior to 3D visualization based on user input parameters for roofs (type, pitch angle, height, etc), the 3D representation of roof form at a later stage will have no effect on the accuracy of the calculations. On the

other hand, creating a comprehensive solution to cover every possible iteration of roof combinations is an open topic and out of the scope of this project at the time of authoring. Therefore the generation of roof elements is restricted to the most common hip and gable roof types and is realized using the straight skeleton algorithm. This area has the potential of further improvements by incorporating specially tailored algorithms in future.

4 SYSTEM DESIGN AND DEVELOPMENT

4.1 Data mapping

As described earlier, development of the data transformation software is initially dependent on identification of entity relationships between source and destination file formats. Towards this approach a conversion table is developed from the available data elements which have a direct relationship and an equivalent in both source and destination name spaces. The remaining required information for data transformation had to be generated based on other source data with indirect counterparts.

Table 2: Existing entity relationships between SBM and SceneJS

SBM XML		ScenJS JSON	
Building	name	data.relationships.decomposedBy (type = Building)	name
	id		Id
Section	name	data.relationships.decomposedBy (type = BuildingStorey)	name
	id		Id
Enclosure	id	library.geometry	coreid
	type	library.material	name

As apparent in Table 2, only the main categorizations of the SBM model have direct counterparts in BIM Surfer SceneJS. These directly related data mostly include components groupings which are required for the “expose” functionality of BIM Surfer. Due to its uniqueness, the “id” property of every enclosing element is the key connecting component which facilitates complex attribute allocations later in the transformation process and data retrieval stage. The remaining required information which needs to be extracted and processed includes coordinate values of enclosures and apertures as well as construction data.

4.2 General approach and data processing

Having discussed the state of the problem and considering the existing entity relationships, a comprehensive solution is proposed based on an object oriented approach. The whole procedure is divided into three main stages of data processing with each followed by an output stage. This incremental output generation eliminates the need for further storage of the entire large JSON file prior to final export. The overall workflow is described in Figure 18.

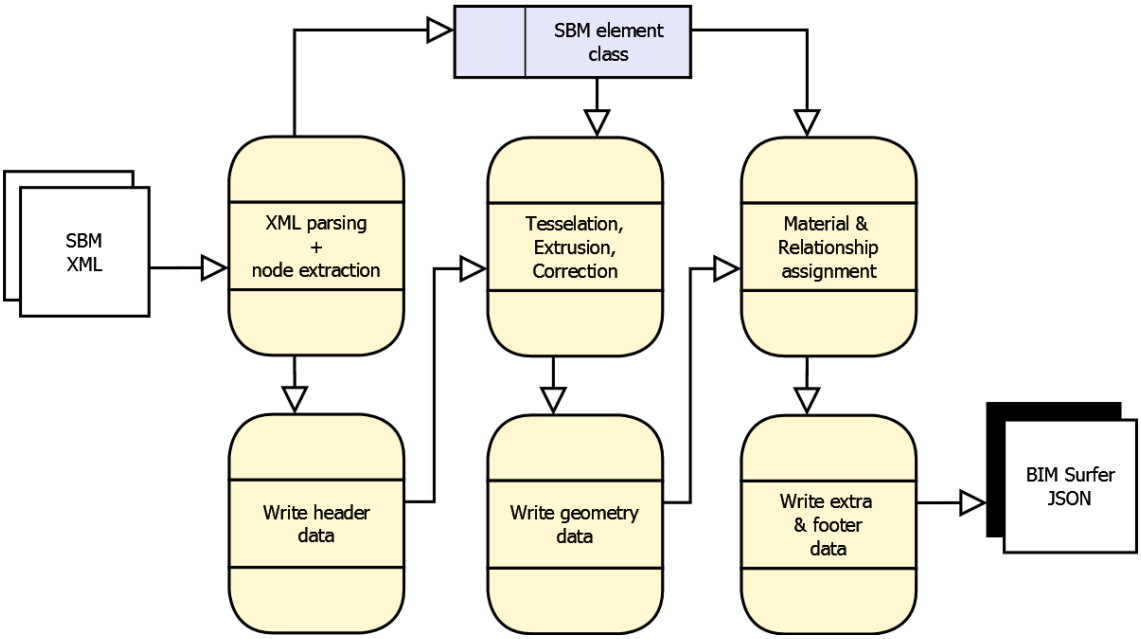


Figure 18: Software solution workflow diagram

Java is chosen as the prefered developing environment because of two main reasons. First is the interoperability and compatibility aspect with the SEMERGY project which has been developed in Java. This facilitates seemless implemention of the visualization tool into the project with less effort and reduces the incompatibility issues. Second is the availability of various standard and established software libraries developed for Java, cutting short the time and effort required for laborious routine tasks such as input/output actions and complex mathematical and geometrical operations.

4.2.1 XML parsing and node extraction

Upon initialization of the application, the output building model of SEMERGY encapsulated as an XML, is parsed as a DOM (Document Object Model) document to be queried. Because of the abilities to select XML tree nodes based on various certain criterias which is the case in this project, XPath is chosen as the query language for the source. To organize queries, simplfying the query codes and reduce the instances of I/O, an XPath query string generator function (XPathQuery) is developed:

Definition:

```
XPathQuery(sectionCounter,spaceCounter,objCounter,switch);
```

Sample usage:

```
String Wallquery = XPathQuery(1,1,1,"w");
NodeList Wallnodes = (NodeList) xpath.evaluate(Wallquery,xmldoc,
XPathConstants.NODESET);
```

The first two arguments of XPathQuery function refer respectively to the integer values of the particular element's section value and space number. "objCounter" is the index of the enclosing object in the said space, out of total number of enclosures. "switch" is the query's target parameter which returns specific information based on its value. Valid values for this parameter are described in Table 3:

Table 3: XPathQuery function: switch argument

switch	description	switch	description
w	Wall coordinates	*Conr*	Construction Ref.
aw	Adjacent walls' coord.	*F*	Floor coord.
h	Apertures' coord.	*Fr*	Floor Ref. coord.
ah	Adjacent apertures' coord.	*Fc*	Floor Construction
htyp	Aperture type	*Hcon*	Aperture construction
en	Enclosure Type	*Sec*	All enclosure types
con	Construction thickness	*Id*	Object ID
conmat	Construction material	*conres*	Construction resistance

Querying and extraction of all required data nodes is performed by traversing the XML tree from the root, therefore every single building section and their related child objects are scanned and stored in an implementation of an AbstractList.

In the latest version of SEMERGY, data storage of multiple storey buildings is done by referencing the spaces in higher building sections to adjacent spaces of the lowest building section. In this manner, instead of having parallel branches with many identical leaves for every section, they only carry a referencing unique ID which points to the target space. This space is located deep in the first section branch of the XML tree. For this case a dedicated function fetches referenced data to the AbstractList by generating the corresponding XPath query.

Accessing members of an AbstractList in multiple occasions is a laborious task due to the nature of its data structure. Therefore to keep an organized dynamic data structure of the queried elements for fast and effortless recovery, a dedicated class is created.

The “SbmElement” class contains major attributes required for instantiation of all types of building elements as well as helper parameters and several methods for geometrical corrections [Figure 19].

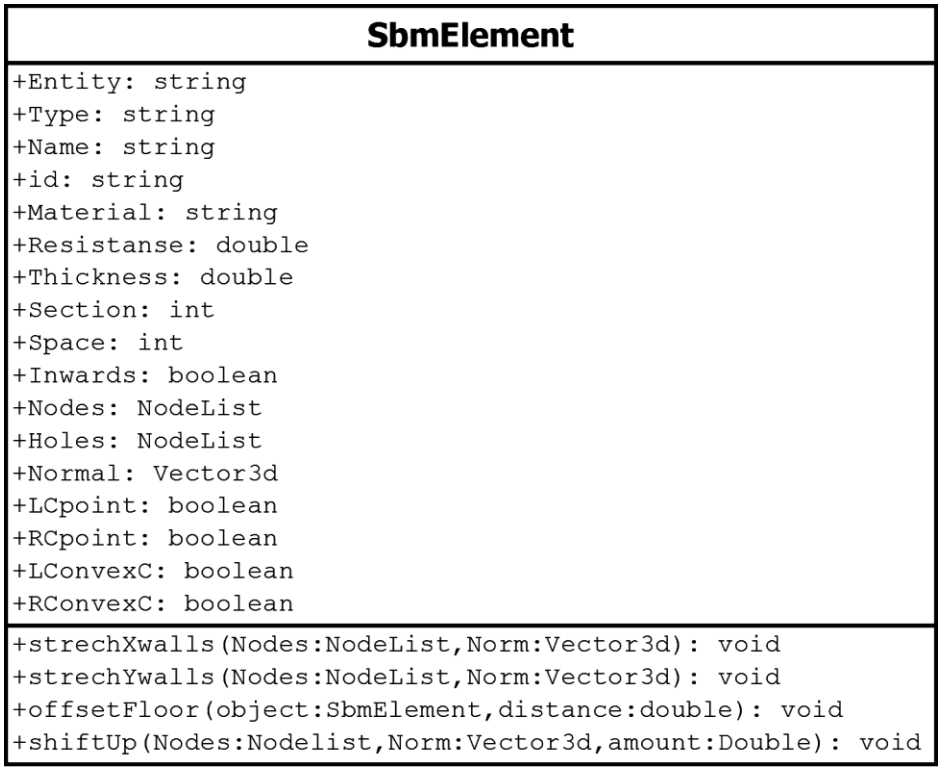


Figure 19: SbmElement class

At this stage, instances of SbmElement class are 2D planar surfaces yet to be processed. The SbmElement class attributes and methods are described in Table 4.

Table 4: SbmElement class - description of attributes and methods (* : Depricated)

Attribute / Method	*Description*
Attributes	
Entity	Elements entity being either a solid or a void (aperture).
Type	Enclosure type of the building component
Name	Auto generated Name of the building component
Id	Unique identification string of the building component identical to SBM value
Material	Construction material name
Resistance	Total thermal resistance of the construction in (m^2K)/W.
Thickness	Overall construction thickness assigned to the building component
Section	The SBM section (building storey) in which the building component belongs to.
Space	The SBM space in which the building component belongs to.
Inwards	Flag stating whether the component's surface normal points towards the inside of the space.
Nodes	A nodelist containing four corner coordinates of the component.
Holes	A nodelist containing corner coordinates of apertures of the component.
Normal	Normal vector of the component.
LCpoint	Flag stating whether a vertical object has a connected neighbor at its left-side vertices.
RCpoint	Flag stating whether a vertical object has a connected neighbor at its right-side vertices.
LConvexC	Flag stating whether a vertical object forms a convex corner at its left-side vertices.
RConvexC	Flag stating whether a vertical object forms a convex corner at its right-side vertices.
Methods	
strechXwalls*	Method for 2 dimensional scaling of wall components in YZ plane
strechYwalls*	Method for 2 dimensional scaling of wall components in XZ plane
offsetFloor	Method for offsetting the perimeter polygon of the building for roof/slab generation.
shiftUp	Method for repositioning of components based on thickness of horizontal components

Once all space enclosing elements have been queried and stored as instances of SbmElement, a series of geometrical inspections are performed on various components to identify shared vertex coordinate values between multiple objects. The results of these inspections are then used in geometrical corrections of the planar objects prior to triangulation and extrusion.

4.2.1.1 Finding building perimeter

Building perimeter or footprint coordinates of the building is required to perform certain procedures to identify convex corners on adjacent walls. It is also used to calculate the centroid of the footprint polygon to re-position the model and camera target in accordance to the origin of BIM Surfer coordinate system.

Building perimeter is calculated by traversing through the X and Y coordinates of all external wall elements and creating related edges between each pair of vertices. This value is stored in "`allPerimEdges`" which is a list of the type "edge".

4.2.1.2 Identifying adjacent objects overlap status

In order to perform correct chamfering of wall ends especially at corner junctions, vertices located at both ends of adjacent walls are being compared. For this operation, two types of corners have been defined:

- **Concave corners (internal corners)**: corners which their constructing walls have an angle of 90 degrees between them; therefore their surface normals intersect inside the boundaries of the building perimeter.
- **Convex corners (external corners)**: corners which their constructing walls have an angle of 270 degrees between them; therefore their surface normals intersect outside the boundaries of the building perimeter.

As mentioned in Table 4, concave corner operations toggle the LCpoint and RCpoint flags by comparing the surface normal of the two objects and their left and right vertices. In case of surface normal equality, both flags are set to False. If the shared vertices are located at the right side of the wall, RCpoint will be True and contrariwise the LCpoint.

Convex corners however undergo the extra step of boundary containing check for their surface normals intersection point, of which in case of validity, LConvexC and RConvexC are being set accordingly.

4.2.1.3 Roof element generation

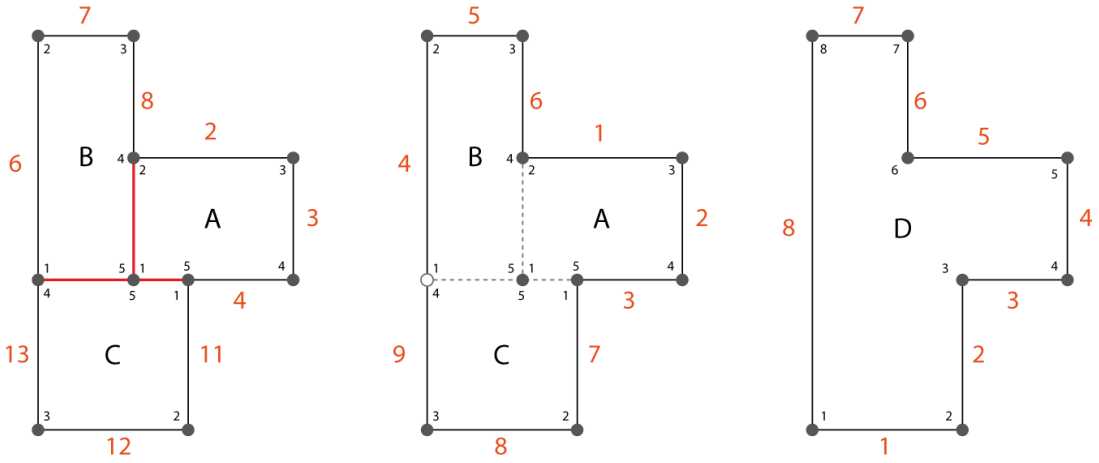
In general, roof element generation process consists of the following stages:

- roof perimeter calculation
- filtering segments to achieve main boundary edges
- edge sorting
- identification of gable walls
- straight skeleton generation

Results of the final stage will then be parsed to triangulation function along with all other building elements after geometrical adjustments.

To achieve roof perimeter for generation of roof forms, the same procedure for finding overall building perimeter is performed, however only on “roof” elements. Since in most cases the overall roof is made up of many sections (smaller roofs of individual spaces on same level), this results in a list of edges which includes not only the main perimeter edges but the edges of all constructing roof sections. This list has to be filtered to accommodate only the required edges for roof generation. Below diagram depicts roof perimeter edges in each step:

Table 5 : Roof edges processing



Original polygons	Shared edge removal	Edge sorted result
Edge list		
A(1,2) B(1,2) C(1,2) A(2,3) B(2,3) C(2,3) A(3,4) B(3,4) C(3,4) A(4,5) B(4,5) C(4,5) A(5,1) B(5,1) C(5,1)	A(2,3) B(1,2) C(1,2) A(3,4) B(2,3) C(2,3) A(4,5) B(3,4) C(4,5)	D(1,2) , D(2,3) , D(3,4) D(4,5) , D(5,6) , D(6,7) D(7,8) , D(8,1)
Edge order		
2,3,4,6,7,8,11,12,13 (tweested polygon)	1,2,3,4,5,6,7,8,9 (tweested polygon)	1,2,3,4,5,6,7,8 (valid polygon)

Since current version of SEMERGY does not provide information regarding the gable walls for gable roof type, an algorithm is developed which estimates the possible expected position of gable walls for majority of roof boundary shapes.

The algorithm first determines the boundary box of the roof polygon (a) and marks the edges which lay on this rectangular boundary (b). In the next step, the edges which have the exact same length (start and end vertices) as of the boundary box (box's width or

height) are removed from the list of possible edges (c). The remaining edges are then flagged as gable walls for triangulation process. Figure 20 shows the process of gable edge identification.

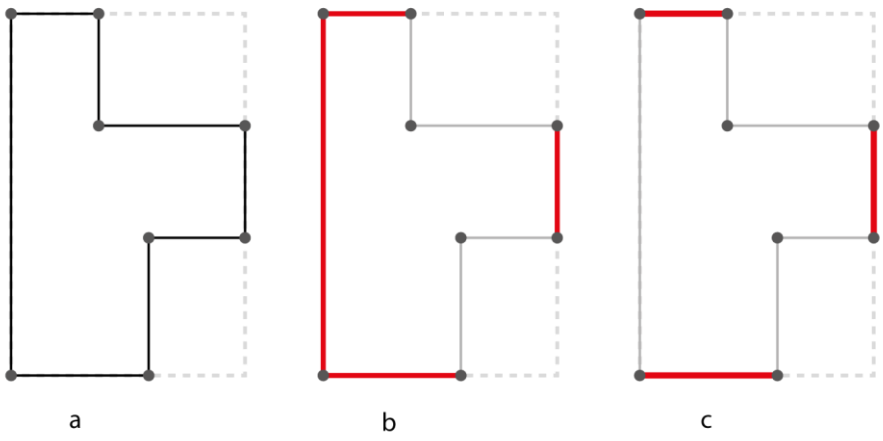


Figure 20 : Gable wall identification process

Once the valid roof polygon with counter clockwise ordering of edges is achieved, a straight skeleton algorithm (Kelly 2014) is used to create face subdivisions of the roof shape based on polygon's straight skeleton. The straight skeleton of a polygon is a topological path made of angular bisectors of the polygons corners, traced by wavefront vertices [Figure 21].

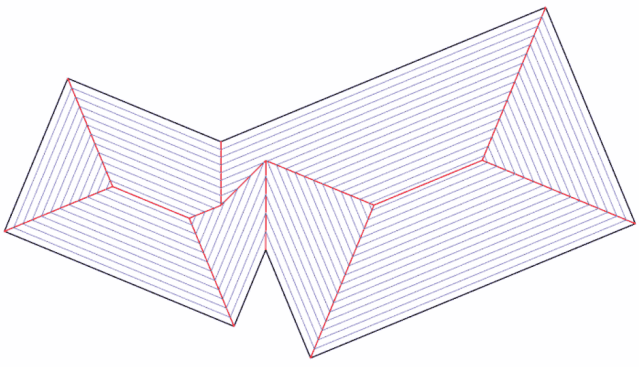


Figure 21: Straight Skeleton of a polygon; blue: wavefronts, red: straight skeleton (Bélanger 2000)

Having achieved the coordinates of roof ridges, they are then elevated based on roof pitch angle. The outcome of this process is a typical hip roof. To achieve gable roof variation, same algorithm is used again taking into account the previously flagged gable edges as weighted edges with a weight value of “0”. The weight value is equal to face angle in radians. The resulting vertices of both procedures are then used as different SbmElement objects in triangulation and later toggling in BIM Surfer UI.

4.2.1.4 Stretching walls elements and slab offsetting

Applying coordinate corrections to 2D polygons prior to 3D mesh generation is more efficient and straightforward in terms of process complexity and performance since operations has to be performed on much fewer data nodes. Since this project was an ongoing effort with constant refinement to various sections over time, some procedures which may seem trivial or ineffective at the current stage, are left intentionally in the workflow to demonstrate solution enhancements and progress. Extrusion of wall elements outwards to maintain true inner space dimensions will result in empty corners where two perpendicular walls meet. As mentioned earlier this can be detected and fixed in 3D mesh generation stage by appropriate chamfering. However this was initially dealt with through extending wall elements by their thickness amount to fill the empty corners.

Likewise, horizontal elements such as floors and ceilings have to be extended to fill the gap between walls of different levels. This is achieved via an offsetting function to enlarge the original polygon based on the surrounding wall thicknesses.

4.2.1.5 Vertical shifting

Since horizontal elements of the building are getting extruded upwards to gain various thicknesses, all other elements have to move up correspondingly to maintain accurate heights and inner dimension.

4.2.1.6 Reposition to systems origin

BIM Surfer is configured by default to use the coordinate system's origin as camera target in all viewports. On the other hand, majority of SBM models are usually located away from the origin. In order to center the model in the viewing window, the repositioning process alters coordinates of each vertex in relation to the calculated centroid of the building's boundary polygon.

4.2.2 Writing header data

Once all building element types and their constructions are extracted from the SBM, information regarding the initialization of the scene and various material properties are written as the header of the JSON file in the root. Scene initialization parameters are constant in all cases whereas every material node is identified by a unique ID having different RGB value as its attributes. This exclusive ID, which in this case is a textual meaningful label, is then used to link all geometries of the same type to their corresponding material. Below is a sample generated material data for the "basement ceiling" object type:

```
"nodes" : [{
      "type" : "material",
      "coreId" : "basementceiling",
      "baseColor" : {
            "r" : 0.7,
            "g" : 0.7,
            "b" : 0.7
      },
      "alpha" : 1,
      "emit" : 0.0
}
```

Figure 22: JSON representation of a material node

Material nodes are generated for all types of building elements/construction.

4.2.3 Triangulation, extrusion and corrections

Till this stage SbmElement objects are inspected, adjusted and ready for further process. As mentioned earlier, objects in SceneJS three dimensional space have to be made up of triangular faces. This necessitates a Delaunay triangulation of polygons to achieve desired faces. At this stage, single SbmElement objects are triangulated at a time, giving a list of constructive triangles which are yet planar. By cloning a second layer of the same triangles at a distance equal to SbmElement's thickness, the building element comes closer to a 3D volumetric mesh though still missing the surrounding faces at sides, top and bottom. Chamfering of the edges is done at this stage according to element's corner status flags by repositioning of the second layer vertices as required. For every triangle in the list of triangles of an element, a set of 6 vertices, 3 on the front and 3 at the back, form a prism. A dedicated algorithm processes each prism and creates 8 faces for all 5 sides [Figure 23].

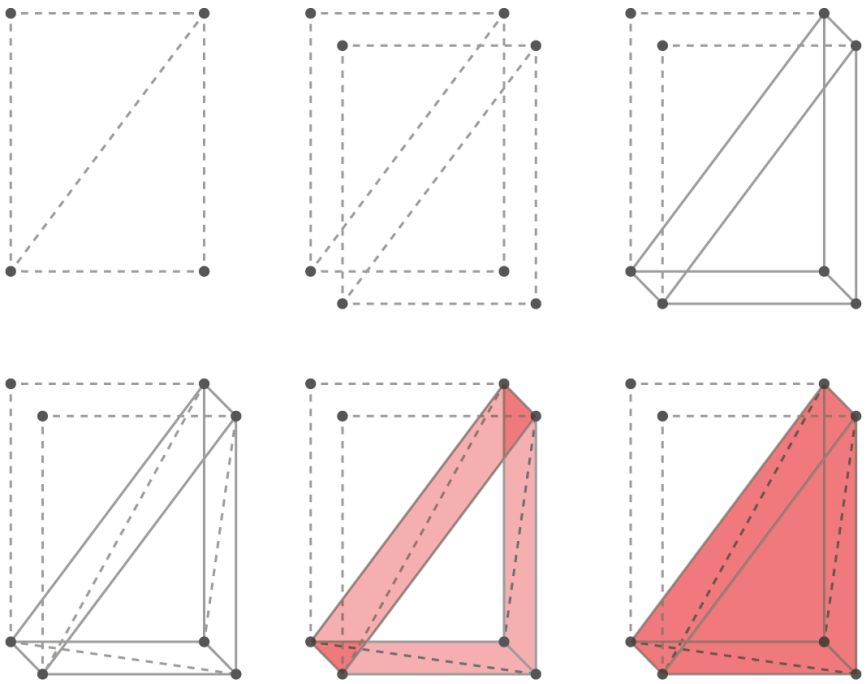
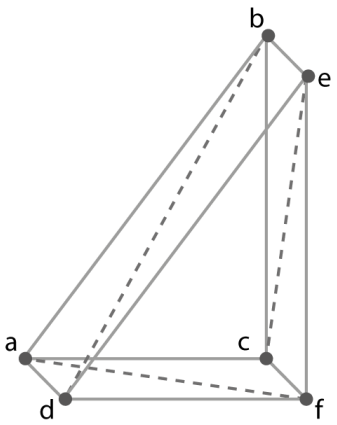


Figure 23:Triangulation, extrusion and mesh generation of faces

Once all triangles of an element are extruded after necessary modifications, they are returned as an ordered list of vertex coordinates for writing of geometry data to the JSON file. The order of vertex and face generation is of utmost importance since it has to be maintained in the JSON geometry definition structure (Indices array). Although it is possible to reduce the number of repetitions of face vertex coordinates in the list, but the order in which the faces have to be generated will then be a complex sequence of integers. Therefore, it is decided to repeat vertices per face to be able to generate a mesh using natural number sequence as indices.

Figure 24 describes this concept for a simple prism. The value of an index corresponds to the position of the vertex in the "positions" array.



First approach

Positions	[a,b,c, d,e,f]
Indices	[1,2,3,1,2,4,2,4,5,2,3,4,3,5,6,3,6,1,1,4,6,4,5,6]

Second approach

Positions	[a,b,c, a,b,d, b,d,e, b,c,e, c,e,f, a,c,f, a,d,f, d,e,f]
Indices	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20, 21,22,23,24]

Figure 24 : Positions and indices generation concept

Although the positions list of the second approach may seem to have the same complexity as indices of the first, however this list already exists as the outcome of previous steps of face generation and no further processing is required to create such array. The purpose of this concept is solely to keep an ordered index array for easy reference, reduced complexity for objects with numerous faces and to follow SceneJS and BIM Surfer's file conventions.

4.2.4 Writing geometry data

In the next step, the ordered list of vertex coordinates of single building elements is appended to the JSON file as the "positions" attribute of "geometry" node. This is

followed by face normals calculated from vertex coordinates cross product in form of a treble. This is an optional state which is a requisite of true shaded rendering. The next dataset to be written is the list of indices which has been defined earlier. Therefore such list is a simple sequence of positive integer numbers starting with “0” and ends with number of total positions coordinates divided by three. Below example shows how a simple box is stored in JSON format:


```
{
                    "type" : "geometry",
                    "coreId" : "1XNle7gSL5qg_hJOOjcGCq",
                    "primitive" : "triangles",
                    "positions" : [140, 145, -200, 140, 145, 200, -
140, 145, 200, 140, 145, -200, -140, 145, 200, -140, 145, -200, 140, -
145, -200, -140, -145, 200, 140, -145, 200, 140, -145, -200, -140, -
145, -200, -140, -145, 200, 140, 145, -200, 140, -145, -200, 140, 145,
200, 140, -145, 200, 140, 145, 200, 140, -145, -200, 140, 145, 200,
140, -145, 200, -140, 145, 200, -140, -145, 200, -140, 145, 200, 140,
-145, 200, -140, 145, 200, -140, -145, 200, -140, 145, -200, -140, -
145, -200, -140, 145, -200, -140, -145, 200, -140, 145, -200, -140, -
145, -200, 140, 145, -200, 140, -145, -200, 140, 145, -200, -140, -
145, -200],
                    "normals" : [-0, 1, -0, -0, 1, -0, -0, 1, -0, -0,
1, -0, -0, 1, -0, -0, 1, -0, -0, -1, -0, -0, -1, -0, -0, -1, 0, -0, -
1, -0, -0, -1, -0, -0, -1, -0, 1, -0, -0, 1, -0, -0, 1, -0, -0, 1, -0,
-0, 1, -0, -0, 1, -0, -0, 0, 0, 1, -0, -0, 1, -0, -0, 1, -0, -0, 1, -
0, -0, 1, -0, -0, 1, -1, -0, -0, -1, -0, -0, -1, -0, -0, -1, -0, -0, -
1, -0, -0, -1, -0, -0, 0, -0, -1, -0, 0, -1, -0, 0, -1, -0, 0, -1, -0,
0, -1, -0, 0, -1],
                    "indices" : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35]
                }
```


Figure 25: JSON representation of a geometry node

This step is repeated for all SbmElement instances, creating necessary geometrical data for entire building components.

4.2.5 Material and relationship allocation

In order to be rendered with correct visual properties during visualization, geometries of

the same type sharing a commonly defined material have to be grouped under a particular “tag”. This tag is used by BIM Surfer software to assign render elements in the scene graph at runtime. Below is an example of grouping two groundfloor objects under a “groundfloor” material:

```
{
    "type" : "tag",
    "tag" : "groundfloor",
    "id" : "groundfloor",
    "nodes" : [{
            "type" : "material",
            "coreId" : "groundfloor",
            "nodes" : [{
                    "type" : "name",
                    "id" : "e6a5aa95-ad1a-721f1abf1769",
                    "nodes" : [{
                            "type" : "geometry",
                            "coreId" : "e6a5aa95-ad1a-721f1abf1769"
                        }
                    ]
                }, {
                    "type" : "name",
                    "id" : "548d9a8a-ab7d-41eb-8d3a-081ae990dd07",
                    "nodes" : [{
                            "type" : "geometry",
                            "coreId" : "548d9a8a-8d3a-081ae990dd07"
                        }
                    ]
                }
            ]
        }
    ]
}
```

Figure 26: JSON representation of material assignment

Scene visual setup information such as camera settings (position of eye and target, FOV, aspect ratio, etc.) and light source parameters are also defined at this stage, which is usually consistent in all cases.

In order to populate the objects tree in BIM Surfer environment and accordingly setup the expose functionality of building stories, building components hierarchical relationships has to be defined. This is done through the “relationships” node located under “data” superclass in form of a nested structure of attributes. These set of attributes

correspond to collection of spaces by defining their constructing elements and properties. A typical relationship object node includes following attributes:

Table 6: BIM Surfer's Relationship object attributes

Attr. name	Data type	Description
type	string	"relationship object" type , set to "BuildingStorey" by default.
name	string	Textual label of the current storey
id	string	An optional identification tag for the current storey
decomposedBy	relationship object	Building storey which is above the current storey
definedBy	relationship object	Spaces which are defined by elements of the current storey
contains	core id	ID of Individual building elements which form the current storey

4.2.6 Writing additional and footer data

Extra information under the "data" node of the SceneJS tree enables BIM Surfer to display additional semantic information regarding building elements as well as providing further render setting and variables. These render settings include dimension unit and scene boundaries which are calculated and extracted earlier from the SBM.

At this stage, two additional data elements have been added to the default SceneJS tree structure. These are "*heatingDemand*" and "*properties*". By this addition, the semantic data nodes now consist of:

- *ifcTypes* : List of all building element types for visibility toggling at render time
- *heatingDemand*: Annual heating demand for energy certificate generation
- *properties*: Semantic information regarding individual building components.

These include: Unique ID, Component name, Construction material, Construction thickness and Construction thermal resistance.

Appending the above information to the final output concludes the generation of the JSON file, making it ready for direct transload in BIM Surfer.

4.3 Enhancements to BIM Surfer interface

As mentioned earlier, a number of efforts have been made to enhance the user experience of BIM Surfer with regards to requirements of a performance simulation tool. Although SEMERGY tool provides a comprehensive report after final optimization of the building, yet it has been decided to present several performance related information to the user prior to optimization for reference and evaluation. The applied improvements to BIM Surfer include:

- Implementation of building components properties panel.
- Implementation of building energy certificate indicator showing pre-optimization annual heating demand and classification.
- Implementation of an additional shading mode, rendering building objects using a particular spectrum of colors which corresponds to individual building elements thermal conductivity to spot heat losing elements.
- Implementation of option to switch roof type.
- Replacing navigational links with graphical icons.

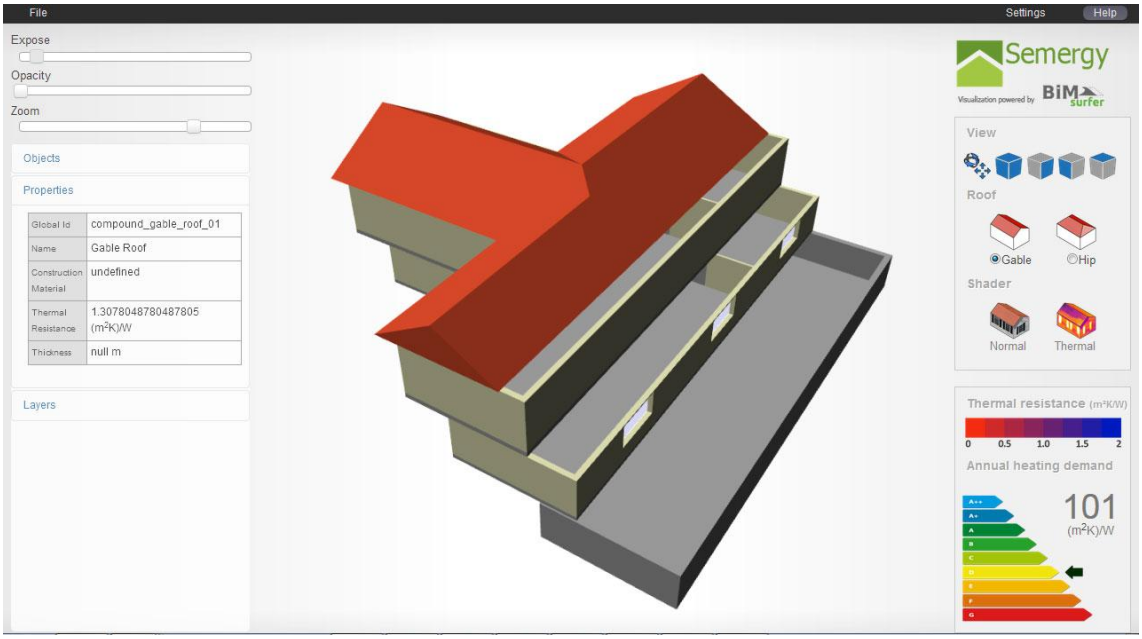


Figure 27: BIM Surfer with final enhancements showing expose function

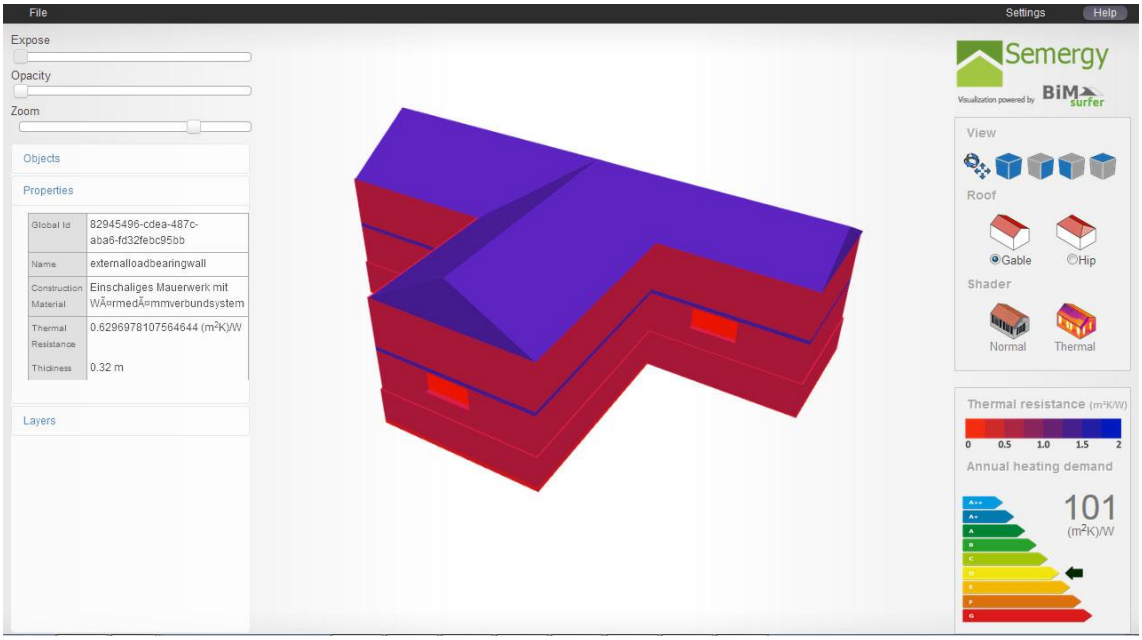


Figure 28: BIM Surfer showing same building with thermal shading

5 CONCLUSION AND FUTURE AREAS OF IMPROVEMENT

The objective of this thesis project was to make efforts towards providing a three dimensional visualization environment for SEMERGY building performance analysis and optimization tool, to elevate the degree of visual perception for the end user.

BIM Surfer has been investigated and explored as the preferred web-based visualization environment due to its performance capabilities and unique features in the area of BIM simulation.

A software solution in form of a data serializer with an object-oriented approach has been developed using Java to facilitate the conversion of SEMERGY Building Model (SBM) data types to BIM Surfer working format. This is essentially a conversion of data containers from source SBM XML to SceneJS JSON as destination, through mapping of corresponding data nodes, performing numerous geometrical operations, and allocation of semantic properties.

Improvements have also been made to the BIM Surfer environment to enhance user experience and level of conveyed information.

The final software is capable of processing the current version of SBM files with an average rate of 260 faces per second, generating an average sized building in ~5 seconds. This process can be improved further by reducing the amount of recursive geometrical operations such as face creation and corrections along with implementing more efficient temporary data storage and retrieval methods.

Support for roof generation is currently limited to gable and hip roof types of buildings with connected roof sections. This can be extended by implementing specifically designed algorithms for automatic roof generation.

The possibility of having a bi-directional edit/view environment to facilitate parametric editing of the building model while in 3D visualization mode is a topic of further development. This though demands mutual dynamic data exchange with core SEMERGY engine which at the time of this writing was not possible due to development restrictions of SEMERGY.

6 REFERENCES

Aish, R. (1986). Three-dimensional input and visualization. *CAAD FUTURES DIGITAL PROCEEDINGS* .

Bélanger, D. (2000). *Designing Roofs of Buildings.* Retrieved Feb 2014, from Sable research group, University of McGill: http://www.sable.mcgill.ca/~dbelan2/roofs/roofs.html

BIM Server. (2013). Retrieved December 2013, from Open source Building Information Modelserver: http://www.bimserver.org

BIM Surfer. (2013). Retrieved September 2013, from BIM Surfer: http://www.bimsurfer.org

Brown, A. G. (2003). Visualization as a common design language: connecting art and science. *Automation in Construction , 12 (6)* , 703-717.

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann.

Cymru. (2013). *Design Builder software review*. Retrieved 2013, from Delivering Low Carbon Buildings Cymru: http://www.lowcarboncymru.org/Software_review/DESIGNBUILDER.pdf

Di Benedetto, M., Ponchio, F., Ganovelli, F., & Scopigno, R. (2010). SpiderGL: A JavaScript 3D Graphics Library for Next-Generation WWW. *15th Conference on 3D Web technology.*

Ghiassi, N. (2013). Development of Building Data Model for a Performance-based Optimization Environment.

Haala, N., & Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing 65* , 570-580.

Hammerberg, K., Jain, V., Ghiassi, N., & Mahdavi, A. (2013). *Generalizing roof geometry from minimal user input for building performance simulation.* Vienna University of Technology.

IFC TOOLS Project. (n.d.). Retrieved December 2013, from IFC TOOLS Project: http://www.ifctoolsproject.com

IFC Tools Project. (2013). Retrieved December 2013, from IFC Tools Project: http://www.ifctoolsproject.com

IFC Web Server. (2013). Retrieved December 2013, from IFC Web Server: http://www.ifcwebserver.org

Kay, L. (2013). *SceneJS*. Retrieved Desember 2013, from SceneJS: www.scenejs.org

Kelly, T. (2014). Unwritten Procedural Modeling with the Straight Skeleton. *PhD thesis, University of Glasgow* .

Kiesel, K., Skoruppa, L., & Mahdavi, A. (2013). Recent advances in BIMSUSTAIN: The application of building information modeling in the context of building physics and building ecology. *Proceedings of the 2nd Central European Symposium on Building Physics 9-11*.

Mahdavi, A. (2004). Notes on representation in computer-aided architectural design. *GCAD Symposium Proceedings.* Pittsburgh, PA, USA.

Mahdavi, A. (2004). Reflections on computational building models. *Building and Environment 39, no. 8 , 39,* 913-925.

Maruna, M. a. (2005). Elements of Urban Planning Methodology Based on Rational Unified Process. *Proceedings of The 9th International Symposium on CUPUM Vol 5.*

Michael Theiler, E. T. (2009). Visualisierung von IFC-Objekten mittels Java3D. *Forum Bauinformatik 2009, Universität Karlsruhe (TH)*, (pp. 149-159).

Sorte, G. J. (1975). Methods for Presenting Planned Environments. *Man-Environment Systems 5, No. 3* .

Stee, J., Drogemuller, R., & Toth, B. (2012). Model interoperability in building information modelling. *Softwares and Systems Modelling , 1-11* .

Various. (n.d.). *SceneJS Wiki*. Retrieved 3 5, 2013, from SceneJS Wiki: http://scenejs.wikispaces.com

Xie, N., & Seng, D. (2012). Application of Two Rendering Techniques in the Visualization of 3D Geospatial Data. *Procedia Environmental Sciences 12* , 1432-1439.

Zach, R., Glawischnig, S., Hönisch, M., Appel, R., & Mahdavi, A. (2012). MOST: An open-source, vendor and technology independent toolkit for building monitoring, data preprocessing, and visualization. *Ework and Ebusiness in Architecture, Engineering and Construction (97)* .

7 FURTHER RESOURCES ON SEMERGY PROJECT

WISS. PUBLIKATIONEN, (Anzahl: 11):

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit Tagungsband; 8 Seiten; validiert (26.09.2012))
A. Mahdavi, U. Pont, F. Shayeganfar, N. Ghiassi, A. Anjomshoaa, S. Fenz, J. Heurix, T. Neubauer, A. Tjoa:
"*SEMERGY: Semantic web technology support for comprehensive building design assessment*";
Vortrag: ECPPM2012 eWork and eBusiness in Architecture, Engineering and Construction,, Reykjavík, Island; 25.07.2012 - 27.07.2012; in: "*eWork and eBusiness in Architecture, Engineering and Construction*", G. Gudnason, R. Scherer (Hrg.); Taylor & Francis, (2012), ISBN: 978-0-415-62128-1; S. 363 - 370. (CODE: 10)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit CD- oder Web-Tagungsband; 7 Seiten; validiert (03.12.2012))
A. Mahdavi, U. Pont, F Shayeganfar, N. Ghiassi, A. Anjomshoaa, S. Fenz, J. Heurix, T. Neubauer, A. Tjoa:
"*Exploring the utility of semantic web technology in building performance simulation*";
Vortrag: BauSim2012 - "Gebäudesimulation auf den Größenskalen Bauteil, Raum, Gebäude, Stadtquartier", Berlin; 26.09.2012 - 28.09.2012; in: "*BauSIM 2012 - "Gebäudesimulation auf den Größenskalen Bauteil, Raum, Gebäude, Stadtquartier"*", C. Nytsch-Geusen et al. (Hrg.); Eigenverlag / wissenschaftliches Kommittee der IBPSA Germany-Austria, 1 (2012), Paper-Nr. 114, 7 S. (CODE: 11)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit CD- oder Web-Tagungsband; 6 Seiten; validiert (14.02.2013))
N. Ghiassi, F Shayeganfar, U. Pont, A. Mahdavi, S. Fenz, J. Heurix, A. Anjomshoaa, T. Neubauer, A. Tjoa:
"*Improving the usability of energy simulation applications in processing common building performance inquiries*";
Vortrag: Simulace Budov a Techniky Prostredi - 7. narodni konference s mezinarodni ucasti, Brno, Tschechien; 08.11.2012 - 09.11.2012; in: "*Simulace Budov a Techniky Prostredi*", O. Sikula, J. Hirs (Hrg.); Ceska Technika - nakladatelstvi CVUT, 1 (2012), ISBN: 978-80-260-3392-9; Paper-Nr. 121, 6 S. (CODE: 12)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit Tagungsband; 8 Seiten; validiert (14.10.2013); speziell begutachtet)
U. Pont, F Shayeganfar, N. Ghiassi, M. Taheri, C. Sustr, A. Mahdavi, J. Heurix, S. Fenz, A. Anjomshoaa, T. Neubauer, A. Tjoa:
"*Recent advances in SEMERGY: A semantically enriched optimization environment for performance-guided building design and refurbishment*";
Vortrag: CESBP - 2nd Central European Symposium on Building Physics, Wien, Österreich; 09.09.2013 - 11.09.2013; in: "*Proceedings of the 2nd Central European Symposium on Building Physics 9-11 September 2013, Vienna, Austria*", A. Mahdavi, B. Martens (Hrg.); ÖKK-Editions, 1 (2013), ISBN: 978-3-85437-321-6; S. 19 - 26. (CODE: 13)

Zeitschriftenartikel: (Zeitschriftenartikel; 7 Seiten; speziell begutachtet)
D. Wolosiuk, N. Ghiassi, U. Pont, F Shayeganfar, A. Mahdavi, S. Fenz, J. Heurix, A.

Anjomshoaa, A. Tjoa:
"*SEMERGY: Performance-Guided Building Design and Refurbishment within a Semantically Augmented Optimization Environment*";
Advanced Materials Research - Web, **899** (2014), S. 589 - 595. (CODE: 14)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit Tagungsband; 8 Seiten; validiert (14.10.2013); speziell begutachtet)
J. Heurix, S. Fenz, A. Anjomshoaa, T. Neubauer, A. Tjoa, M. Taheri, F. Shayeganfar, U. Pont, N. Ghiassi, C. Sustr, A. Mahdavi:
"*Multi-objective optimization in the SEMERGY environment for sustainable building design and retrofit*";
Vortrag: CESBP - 2nd Central European Symposium on Building Physics, Wien, Österreich; 09.09.2013 - 11.09.2013; in: "*Proceedings of the 2nd Central European Symposium on Building Physics 9-11 September 2013, Vienna, Austria*", A. Mahdavi, B. Martens (Hrg.); ÖKK-Editions, 1 (2013), ISBN: 978-3-85437-321-6; S. 27 - 34. (CODE: 15)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit Tagungsband; 8 Seiten; validiert (14.10.2013); speziell begutachtet)
N. Ghiassi, F Shayeganfar, U. Pont, A. Mahdavi, J. Heurix, S. Fenz, A. Anjomshoaa, A. Tjoa:
"*A comprehensive building model for performance-guided decision support*";
Vortrag: CESBP - 2nd Central European Symposium on Building Physics, Wien, Österreich; 09.09.2013 - 11.09.2013; in: "*Proceedings of the 2nd Central European Symposium on Building Physics 9-11 September 2013, Vienna, Austria*", A. Mahdavi, B. Martens (Hrg.); ÖKK-Editions, 1 (2013), ISBN: 978-3-85437-321-6; S. 35 - 42. (CODE: 16)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit Tagungsband; 8 Seiten; validiert (15.10.2013); speziell begutachtet)
K. Hammerberg, V. Jain, N. Ghiassi, A. Mahdavi:
"*Generalizing roof geometry from minimal user input for building performance simulation*";
Vortrag: CESBP - 2nd Central European Symposium on Building Physics, Wien, Österreich; 09.09.2013 - 11.09.2013; in: "*Proceedings of the 2nd Central European Symposium on Building Physics 9-11 September 2013, Vienna, Austria*", A. Mahdavi, B. Martens (Hrg.); ÖKK-Editions, 1 (2013), ISBN: 978-3-85437-321-6; S. 277 - 284. (CODE: 17)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Posterpräsentation mit Tagungsband; 7 Seiten; validiert (17.10.2013); speziell begutachtet)
F Shayeganfar, A. Anjomshoaa, J. Heurix, C. Sustr, N. Ghiassi, U. Pont, S. Fenz, T. Neubauer, A. Tjoa, A. Mahdavi:
"*An ontology-aided Optimization Approach to Eco-Efficient Building Design*";
Poster: BS2013 - Building Simulation 2013, Chambery / Frankreich; 25.08.2013 - 29.08.2013; in: "*Building Simulation 2013 - 13th International Conference of the International Building Performance Simulation Association.*", IBPSA (Hrg.); IBPSA, (2013), ISBN: 978-2-7466-6294-0; S. 2193 - 2199. (CODE: 18)

Vorträge und Posterpräsentationen (mit Tagungsband-Eintrag): (Vortrag mit CD- oder Web-Tagungsband; 6 Seiten; validiert (28.01.2014))
D. Wolosiuk, N. Ghiassi, U. Pont, F Shayeganfar, A. Mahdavi, S. Fenz, J. Heurix, A. Anjomshoaa, A. Tjoa:
"*SEMERGY: Performance-Guided Building Design and Refurbishment within a Semantically Augmented Optimization Environment*";
Vortrag: enviBUILD 2013 - Buildings and Environment, Bratislava; 17.10.2013; in: "*enviBUILD 2013 - Buildings and Environment*", J. Hraska et al. (Hrg.); STU - Nakladateľstvo STU, Bratislava 2013, 1 (2013), ISBN: 978-80-227-4070-8; 6 S. (CODE: 19)

ANGENOMMEN: ECPPM2014 – Wien: *Title: SEMERGY: Utilizing semantic web technologies for performance-guided building design optimization (in Erstellung begriffen)* (CODE: 20)

ANGENOMMEN: PLEA2014 – Ahmedabad, Indien: *Efficient building design model generation and evaluation: The SEMERGY Approach (in Erstellung begriffen)* (CODE: 21)

VORTRÄGE (ohne Tagungsbände), (Anzahl: 4):

Vorträge und Posterpräsentationen (ohne Tagungsband-Eintrag): (Vortrag ohne Tagungsband)
U. Pont:
"*SEMERGY - Exploring the Utility of Semantic Web Technology in Building Performance Simulation*";
Vortrag: Current Topics in Building Performance (Ringseminar WS2012), Wien, TU Wien; 07.12.2012. (CODE: 06)

Vorträge und Posterpräsentationen (ohne Tagungsband-Eintrag): (Vortrag ohne Tagungsband)
K. Hammerberg et al.:
"*SEMERGY: A comprehensive building model for performance-guided decision support*";
Vortrag: Current Topics in Building Performance (Ringseminar WS2013), Wien, TU Wien; 2013. (CODE: 07)

Vorträge und Posterpräsentationen (ohne Tagungsband-Eintrag): (Vortrag ohne Tagungsband)
M. Taheri et al.:
"*Multi-Objective Optimization in the SEMERGY Environment*";
Vortrag: Current Topics in Building Performance (Ringseminar WS2013), Wien, TU Wien; 2013. (CODE: 08)

Teilnahmen an Ausstellungen ohne Katalog: (Teilnahme an Ausstellung ohne Katalog)
N. Ghiassi, K. Hammerberg, U. Pont, A. Mahdavi et al.:
"*SEMERGY - Planung energieeffizienter Gebäude*";
Beteiligung; veranstaltet von: Informationsveranstaltung: "Mehr Energieeffizienz! Zwischen EU-Richtlinie und Umsetzung - Impulse aus der Forschung, Kurator: .. TU Wien, WKÖ; TU Wien, Festsaal, 15.10.2013. (CODE: 09)

AKADEMISCHE ARBEITEN (Anzahl: 5):

Diplomarbeiten – abgeschlossen:

Diplom- und Master-Arbeiten (eigene und betreute): (Diplom- oder Master-Arbeit)
N. Ghiassi:
"*Development of a Building Data Model for Performance-Based Optimization Environment*";
Betreuer/in(nen): A. Mahdavi; Institut für Architekturwissenschaften, Abteilung Bauphysik und Bauökologie, 2013; Abschlussprüfung: 19.04.2013. (Prüfer: A.Mahdavi, B.Martens, A:Anjomshoaa) (CODE: 01)

Dissertationen - ongoing:

Dipl.Ing. U.Pont:

Semergy (Arbeitstitel)

*B*etreuer/in(nen): A. Mahdavi; Institut für Architekturwissenschaften, Abteilung Bauphysik und Bauökologie, (CODE: 05)

8 APPENDIX

8.1 Appendix A: SbmElement class

```
public class SbmElement {

        public String Entity;
        public String Type;
        public String Name;
        public String id;
        public String Material;
        public Double Thickness;
        public Double Resistance;
        public int Section;
        public int Space;
        public boolean Inwards;
        public AbstractList<String> Cords;
        public ArrayList<String> hCords;
        public NodeList Nodes;
        public NodeList Holes;
        public Vector3d Normal;
        public boolean LCpoint;
        public boolean RCpoint;
        public boolean LConvexC;
        public boolean RConvexC;


        void main(){
        for (int i=0; i<12;i++){
        double item = Double.parseDouble(Nodes.item(i).getNodeValue());

Nodes.item(i).setNodeValue(String.valueOf(Math.round(item*100.0)/100.0));
        }
        }
        /// Geometry and position correction functions

        public void strechXwalls(NodeList nodes, Vector3d norm) {
                if (norm.x > 0 ){

        nodes.item(7).setNodeValue(String.valueOf(Double.valueOf(nodes.item(7).getNodeVal
ue())+Thickness));

        nodes.item(10).setNodeValue(String.valueOf(Double.valueOf(nodes.item(10).getNodeV
alue())+Thickness));

        nodes.item(1).setNodeValue(String.valueOf(Double.valueOf(nodes.item(1).getNodeVal
ue())-Thickness));

        nodes.item(4).setNodeValue(String.valueOf(Double.valueOf(nodes.item(4).getNodeVal
ue())-Thickness));
                }
                if (norm.x < 0){

        nodes.item(7).setNodeValue(String.valueOf(Double.valueOf(nodes.item(7).getNodeVal
ue())-Thickness));

        nodes.item(10).setNodeValue(String.valueOf(Double.valueOf(nodes.item(10).getNodeV
alue())-Thickness));

        nodes.item(1).setNodeValue(String.valueOf(Double.valueOf(nodes.item(1).getNodeVal
ue())+Thickness));
```

```
        nodes.item(4).setNodeValue(String.valueOf(Double.valueOf(nodes.item(4).getNodeVal
ue())+Thickness));
                }
   }

        public void strechYwalls(NodeList nodes, Vector3d norm) {
                if (norm.y > 0 ){

        nodes.item(0).setNodeValue(String.valueOf(Double.valueOf(nodes.item(0).getNodeVal
ue())+Thickness));

        nodes.item(3).setNodeValue(String.valueOf(Double.valueOf(nodes.item(3).getNodeVal
ue())+Thickness));

        nodes.item(6).setNodeValue(String.valueOf(Double.valueOf(nodes.item(6).getNodeVal
ue())-Thickness));

        nodes.item(9).setNodeValue(String.valueOf(Double.valueOf(nodes.item(9).getNodeVal
ue())-Thickness));
                }
                if (norm.y < 0){

        nodes.item(0).setNodeValue(String.valueOf(Double.valueOf(nodes.item(0).getNodeVal
ue())-Thickness));

        nodes.item(3).setNodeValue(String.valueOf(Double.valueOf(nodes.item(3).getNodeVal
ue())-Thickness));

        nodes.item(6).setNodeValue(String.valueOf(Double.valueOf(nodes.item(6).getNodeVal
ue())+Thickness));

        nodes.item(9).setNodeValue(String.valueOf(Double.valueOf(nodes.item(9).getNodeVal
ue())+Thickness));
                }
   }


        public  void offsetFloor (SbmElement Bobject, Double h){

                class  Point {
                         double x;
                         double y;

                        //constructor
                    public Point() {
                        x = 0;
                        y = 0;
                    }
                }


                Point[] poly = new Point[Bobject.Nodes.getLength()/3];


                //populate polygon array and vertices list
                ArrayList<Point2d> vertices = new ArrayList<Point2d>();
                Vector2d centroid = new Vector2d();
                int j=0;
                for(int i=0;i<Bobject.Nodes.getLength();i+=3){
                        Point2d temppoint = new Point2d();
                        poly[j] = new Point();
                        poly[j].x = Double.valueOf(Bobject.Nodes.item(i).getNodeValue());
                        poly[j].y =
Double.valueOf(Bobject.Nodes.item(i+1).getNodeValue());

                        temppoint.x = poly[j].x;
                        temppoint.y = poly[j].y;
                        vertices.add(temppoint);
                        centroid.add(temppoint);
                        j++;
                }
```

```
            int N = poly.length;
            centroid.scale(1D/N);

         /////creating JTS offset
           GeometryFactory fact = new GeometryFactory();
           Coordinate[] coords = new Coordinate[vertices.size()+1];
           int k = 0;
           for (int i=0; i<vertices.size();i++){
                  Coordinate coord = new Coordinate();
                  coord.x = vertices.get(i).x;
                  coord.y = vertices.get(i).y;
                  coords[i] = coord;
                  k++;
           }
           coords[k]=coords[0];
           LinearRing linear = new GeometryFactory().createLinearRing(coords);
           Polygon JTSpoly = new Polygon(linear, null, fact);
           Geometry outer = new Polygon(null,null,fact);
           BufferParameters bp = new BufferParameters();
           //set join style to MITRE (2)
           bp.setJoinStyle(2);
           BufferOp buff = new BufferOp(JTSpoly,bp);
           outer = buff.getResultGeometry(h);


                  int c = 0;
                  int outerpoints = outer.getNumPoints()-1;
                  for(int i=0;i<Bobject.Nodes.getLength();i+=3){

                         if(c < outerpoints)
                         {

        Bobject.Nodes.item(i).setNodeValue(String.valueOf(outer.getCoordinates()[c].x));

        Bobject.Nodes.item(i+1).setNodeValue(String.valueOf(outer.getCoordinates()[c].y))
;
                         c++;
                         } else
                         {

        Bobject.Nodes.item(i).setNodeValue(String.valueOf(outer.getCoordinates()[c].x));

        Bobject.Nodes.item(i+1).setNodeValue(String.valueOf(outer.getCoordinates()[c].y))
;
                         }
                  }


        }



        public void shiftUp(NodeList nodes, Double amount){

                  for(int i=2 ; i<=nodes.getLength();i+=3)
                  {

        nodes.item(i).setNodeValue(String.valueOf(Double.valueOf(nodes.item(i).getNodeVal
ue())+amount));
                  }
        }

        public void shiftWinUp(ArrayList<String> nodes, Double amount){

                  for(int i=2 ; i<=nodes.size();i+=3)
                  {
                  nodes.set(i, (String.valueOf(Double.valueOf(nodes.get(i)) + amount)));
                  }
        }
```

```
        public void shiftRoofUp(AbstractList<String> cords, Double amount){

                for(int i=2 ; i<=cords.size();i+=3)
                {
                cords.set(i, String.valueOf(Double.valueOf(cords.get(i)) + amount));


                }
        }
}
```

8.2 Appendix B: JSON Serializer

```
/*
 * SEMERGY SBM to SceneJS JSON Serializer
 * by Amirali Sadeghi (amiraliz@gmail.com)
 *
 * The following software is developed under
 * Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported
 * license. (http://creativecommons.org/licenses/by-nc-sa/3.0/)
 * ---------------------------------------------------------------
 *
 * This software produces SceneJS JSONs compatible with BIMSurfer 2012-10
 * Modify "inputPath" and "outputStream" in main class for specific usage.
 *
 */

import java.io.IOException;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import com.google.common.base.Charsets;
import com.google.gson.stream.JsonWriter;
import com.vividsolutions.jts.geom.Coordinate;
import com.vividsolutions.jts.geom.Geometry;
import com.vividsolutions.jts.geom.GeometryFactory;
import com.vividsolutions.jts.geom.LinearRing;
import com.vividsolutions.jts.operation.buffer.BufferOp;
import com.vividsolutions.jts.operation.buffer.BufferParameters;

import java.io.FileReader;
import java.util.AbstractList;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import java.util.TreeSet;

import org.poly2tri.Poly2Tri;
import org.poly2tri.geometry.polygon.Polygon;
import org.poly2tri.geometry.polygon.PolygonPoint;
import org.poly2tri.triangulation.delaunay.DelaunayTriangle;

import javax.vecmath.Point2d;
import javax.vecmath.Point3d;
import javax.vecmath.Point3f;
import javax.vecmath.Vector3d;
import javax.vecmath.Vector3f;
```

```
import java.awt.geom.Path2D;
import org.xml.sax.InputSource;

import straightskeleton.*;
import straightskeleton.Output.Face;
import utils.*;

public class jsonserializer {

	public static Path2D perim = new Path2D.Float();
	public static Vector3d centroid = new Vector3d();
	public static List<Path2D> roof = new ArrayList<Path2D>();


	public static void main(String[] args) throws
FileNotFoundException,ParserConfigurationException, SAXException,
	  IOException, XPathExpressionException {

		String inputPath = "input.xml";  //input source of sbm XML (out2.xml)
		InputSource xml = new InputSource(new FileReader(inputPath));
		DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
		DocumentBuilder db = dbf.newDocumentBuilder();
		Document xmldoc = db.parse(xml);
		OutputStream outputStream = new FileOutputStream("output.json"); //output
destination for Json
		OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(outputStream, Charsets.UTF_8);

		XPathFactory factory = XPathFactory.newInstance();
	    XPath xpath = factory.newXPath();

	    AbstractList<String> coordinates = new LinkedList<String>();
	    AbstractList<Integer> indices = new LinkedList<Integer>();
	    Set<String> types = new TreeSet<String>();
	    AbstractList<SbmElement> components = new LinkedList<SbmElement>();
	    AbstractList<SbmElement> tempcomponents = new LinkedList<SbmElement>();
	    AbstractList<SbmSection> Bsections = new LinkedList<SbmSection>();
	    AbstractList<Point3d> SlabCords = new LinkedList<Point3d>();
	    AbstractList<Point3d> roofCords = new LinkedList<Point3d>();
	    AbstractList<Point3d> buildingPerimCoords = new LinkedList<Point3d>();
	    AbstractList<edge> allRoofEdges = new LinkedList<edge>();
	    AbstractList<edge> allPerimEdges = new LinkedList<edge>();
	    Double roofResistance = 0.0;


	    NodeList hDemand = (NodeList)
xpath.evaluate("//baseCaseHeatingDemandPerM2/text()",
	            xmldoc,
	            XPathConstants.NODESET);

	    NodeList Sections = (NodeList)
xpath.evaluate("//org.configurator.semergy.sbm.geometry.Section",
	            xmldoc,
	            XPathConstants.NODESET);

	    NodeList Origin = (NodeList)
xpath.evaluate("//org.configurator.semergy.sbm.geometry.Space/coordinate/*/node()",
	            xmldoc,
	            XPathConstants.NODESET);
	    NodeList Area = (NodeList)
xpath.evaluate("//org.configurator.semergy.sbm.geometry.Section/grossFloorArea/node()",
	            xmldoc,
	            XPathConstants.NODESET);

	    double extents = 3 * Math.sqrt(Double.valueOf(Area.item(0).getNodeValue()));

	    double floorMargin = 0;
	    double slabThickness[] = new double[Sections.getLength()+1];
	    slabThickness[0] = 0.0;

		try{
```

```
                        long start = System.nanoTime();
                        JsonWriter jsonWriter = new JsonWriter(new
BufferedWriter(outputStreamWriter));
                        jsonWriter.beginObject();
                        jsonWriter.name("type").value("scene");
                        jsonWriter.name("id").value("Scene");
                        jsonWriter.name("canvasId").value("scenejsCanvas");
                        jsonWriter.name("loggingElementId").value("scenejsLog");

        jsonWriter.name("flags").beginObject().name("backfaces").value(false).endObject()
;
                        jsonWriter.name("nodes");
                        jsonWriter.beginArray();

        jsonWriter.beginObject().name("id").value("library").name("type").value("library"
).name("nodes").beginArray();


                        ////// reading data

                        //  list all enclosure types in each section and write them in
"types"
                        //  This is firstly used to write material section of the Json.
                        // This loop also adds dummy elements for sections as
buildingstories  for "Relationships"
                        for (int sectionCounter = 1; sectionCounter <=
Sections.getLength(); ++sectionCounter){

                                SbmElement dummyelement = new SbmElement();
                                dummyelement.Type = "BuildingStorey";
                                dummyelement.Name = "Level
"+String.valueOf(sectionCounter-1);
                                dummyelement.Section = 1;
                                components.add(dummyelement);

                                String Sectionquery =
XPathQuery(0,0,sectionCounter,"sec");
                                NodeList EncTypes = (NodeList)
xpath.evaluate(Sectionquery,
                                                xmldoc,
                                                XPathConstants.NODESET);
                                String apTypesqQuery =
XPathQuery(0,0,sectionCounter,"apty");
                                NodeList apTypes = (NodeList)
xpath.evaluate(apTypesqQuery,
                                                xmldoc,
                                                XPathConstants.NODESET);
                                if (EncTypes.getLength()==0) continue;
                                for(int i=0 ; i<EncTypes.getLength();i++){
                                        String Enctype = EncTypes.item(i).getNodeValue();
                                        types.add(Enctype.toLowerCase());
                                }
                                for(int i=0 ; i<apTypes.getLength();i++){
                                        String aptype = apTypes.item(i).getNodeValue();
                                        types.add(aptype.toLowerCase());
                                }


                        }
                if(types.contains("gableroof")) {
                        types.remove("gableroof");
                        types.add("roof");
                }
                        writeMaterials(jsonWriter, types);

        /////////// BEGIN of Get floor plane coords

                        //Sections.getLength()
```

```
                        for (int sectionCounter = 1; sectionCounter
<=Sections.getLength(); ++sectionCounter){

                                NodeList Spaces = (NodeList)
xpath.evaluate("(//org.configurator.semergy.sbm.geometry.Section)["+sectionCounter+"]//o
rg.configurator.semergy.sbm.geometry.Space",
                                     xmldoc,
                                    XPathConstants.NODESET);

                                for (int spaceCounter = 1; spaceCounter <=
Spaces.getLength(); ++spaceCounter){

                                        //return referenced space if exists

                                        String FloorConquery = null;
                                        String fRefquery = null;

                                        if (Spaces.item(spaceCounter-1).hasChildNodes() ==
false) {
                                                String path =
returnRefSpacePath(xpath,xmldoc,Spaces,sectionCounter,spaceCounter);

                                                FloorConquery =
XPathRefQuery(path,sectionCounter,spaceCounter,0,"fc");
                                                fRefquery =
XPathRefQuery(path,sectionCounter,spaceCounter,0,"fconr");
                                        } else {
                                                FloorConquery =
XPathQuery(sectionCounter,spaceCounter,0,"fc");
                                                fRefquery =
XPathQuery(sectionCounter,spaceCounter,0,"fconr");
                                        }


                                        NodeList fnodes = (NodeList)
xpath.evaluate(FloorConquery,
                                                                xmldoc, XPathConstants.NODESET);

                                        //// geting referenced floors
                                        if (fnodes.getLength() == 0){
                                                NodeList fnodesRef = (NodeList)
xpath.evaluate(fRefquery,
                                                                        xmldoc,
                                                                        XPathConstants.NODESET);

                                                if(fnodesRef.getLength()==0) continue;

                                                String RefQ =
"//construction[@id="+fnodesRef.item(0).getNodeValue()+"]"+"/constructionThickness/node(
)";

                                                fnodes = (NodeList) xpath.evaluate(RefQ,
xmldoc,
                                                                        XPathConstants.NODESET);

                                        }

                                        if (fnodes.item(0) != null)
                                        slabThickness[sectionCounter] =
Double.valueOf(fnodes.item(0).getNodeValue());
                                        break;

                                        }
                        }

                                        /////////// END of Get floor planes coords


                        for (int sectionCounter = 1; sectionCounter <=
Sections.getLength(); ++sectionCounter){
```

```
                                NodeList Spaces = (NodeList)
xpath.evaluate("(//org.configurator.semergy.sbm.geometry.Section)["+sectionCounter+"]//o
rg.configurator.semergy.sbm.geometry.Space",
                                                xmldoc,
                                                XPathConstants.NODESET);

                                for (int spaceCounter = 1; spaceCounter <=
Spaces.getLength(); ++spaceCounter){

                                        //return referenced space if exists
                                        NodeList BuildingObjs =
returnRefSpace(xpath,xmldoc,Spaces,sectionCounter,spaceCounter);

                                        int objsCount = BuildingObjs.getLength();

                                        for (int objCounter = 1; objCounter <= objsCount;
++objCounter)
                                        {

                                                String Wallquery = null;
                                                String Holequery = null;
                                                String AdjWallquery = null;
                                                String AdjHolequery = null;
                                                String Enclosuretypequery = null;
                                                String Constquery = null;
                                                String resquery = null;
                                                String Refquery =null;
                                                String idquery = null;
                                                String hidquery = null;
                                                String htypquery = null;
                                                String hconquery =null;
                                                String hconresquery = null;
                                                String apConRef = null;
                                                String conMatquery = null;
                                                String apConMatquery = null;

                                                if (Spaces.item(spaceCounter-
1).hasChildNodes() == false) {

                                                        String path =
returnRefSpacePath(xpath,xmldoc,Spaces,sectionCounter,spaceCounter);
                                                        path = path.concat("/node()");

                                                         Wallquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"w");
                                                         Holequery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"h");
                                                         AdjWallquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"aw");
                                                         AdjHolequery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"ah");
                                                         Enclosuretypequery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"en");
                                                         Constquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"con");
                                                         Refquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"conr");
                                                         resquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"conres");
                                                         idquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"id");
                                                         hidquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"hid");
                                                         htypquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"htyp");
                                                         hconquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"hcon");
                                                         hconresquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"hconr");
```

```
                                                    apConRef =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"apconr");
                                                    conMatquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"conmat");
                                                    apConMatquery =
XPathRefQuery(path,sectionCounter,spaceCounter,objCounter,"apconmat");

                                                } else {

                                                    Wallquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"w");
                                                    Holequery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"h");
                                                    AdjWallquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"aw");
                                                    AdjHolequery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"ah");
                                                    Enclosuretypequery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"en");
                                                    Constquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"con");
                                                    Refquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"conr");
                                                    resquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"conres");
                                                    idquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"id");
                                                    hidquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"hid");
                                                    htypquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"htyp");
                                                    hconquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"hcon");
                                                    hconresquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"hconr");
                                                    apConRef =
XPathQuery(sectionCounter,spaceCounter,objCounter,"apconr");
                                                    conMatquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"conmat");
                                                    apConMatquery =
XPathQuery(sectionCounter,spaceCounter,objCounter,"apconmat");
                                                }

                                                NodeList nodes = (NodeList)
xpath.evaluate(Wallquery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);
                                                NodeList holes = (NodeList)
xpath.evaluate(Holequery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);
                                                NodeList adjholes = (NodeList)
xpath.evaluate(AdjHolequery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);
                                                NodeList EncType = (NodeList)
xpath.evaluate(Enclosuretypequery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);

                                                NodeList Coreid = (NodeList)
xpath.evaluate(idquery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);
                                                NodeList ConsT = (NodeList)
xpath.evaluate(Constquery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);
                                                NodeList ConRes = (NodeList)
xpath.evaluate(resquery,
                                                        xmldoc,
                                                        XPathConstants.NODESET);
```

```
                                        NodeList ConMat = (NodeList)
xpath.evaluate(conMatquery,
                                                xmldoc,
                                                XPathConstants.NODESET);


                                        //skip refrenced objects
                                        if (nodes.getLength()==0) continue;

                                        //find referenced construction's thickness
and resistance values
                                        if (ConsT.getLength() == 0){
                                                NodeList ConsRef = (NodeList)
xpath.evaluate(Refquery,
                                                        xmldoc,

        XPathConstants.NODESET);

                                                String RefQ = "//construction[@id='"
+ConsRef.item(0).getNodeValue()+"']/constructionThickness/node()";
                                                ConsT = (NodeList)
xpath.evaluate(RefQ, xmldoc,

        XPathConstants.NODESET);
                                                String ResQ = "//construction[@id='"
+ConsRef.item(0).getNodeValue()+"']/resistance/node()";
                                                ConRes = (NodeList)
xpath.evaluate(ResQ, xmldoc,

        XPathConstants.NODESET);
                                                String MatQ = "//construction[@id='"
+ConsRef.item(0).getNodeValue()+"']/name/node()";
                                                ConMat = (NodeList)
xpath.evaluate(MatQ, xmldoc,

        XPathConstants.NODESET);
                                        }


                                        boolean hasAdj = false;
                                        String ref = null ;
                                        //find referenced apertures
                                        if(adjholes.getLength()!=0)
                                        {
                                                hasAdj = true;
                                                String RefQ =
"//adjacencyObject[@id='" +adjholes.item(0).getNodeValue()+"']/coordinates/*/*/node()";
                                                holes = (NodeList)
xpath.evaluate(RefQ, xmldoc,

        XPathConstants.NODESET);
                                                ref = "//adjacencyObject[@id='"
+adjholes.item(0).getNodeValue()+"']";
                                        }



                                        // apertures initialization
                                        if (holes.getLength() != 0 ){


                                                NodeList apCords = (NodeList) holes;
                                                List<String> cords = new
ArrayList<String>();

                                                for (int
m=0;m<holes.getLength();m++){

        cords.add(holes.item(m).getNodeValue());
```

```
                }

if (hasAdj) hidquery = ref +
"/id/node()";
NodeList apId = (NodeList)
xpath.evaluate(hidquery,
                xmldoc,
        XPathConstants.NODESET);

if (hasAdj) htypquery = ref +
"/apertureType/node()";
NodeList apType = (NodeList)
xpath.evaluate(htypquery,
                xmldoc,
        XPathConstants.NODESET);

if (hasAdj) hconquery = ref +
"/construction/constructionThickness/node()";
NodeList apCon = (NodeList)
xpath.evaluate(hconquery,
                xmldoc,
        XPathConstants.NODESET);

if (hasAdj) hconresquery = ref +
"/construction/resistance/node()";
NodeList apRes = (NodeList)
xpath.evaluate(hconresquery,
                xmldoc,
        XPathConstants.NODESET);

if (hasAdj) apConMatquery = ref +
"/construction/name/node()";
NodeList apMat = (NodeList)
xpath.evaluate(apConMatquery,
                xmldoc,
        XPathConstants.NODESET);

//find referenced construction's
thickness and resistance values
if (apCon.getLength() == 0){
        if (hasAdj) apConRef = ref +
"/construction/@reference";
        NodeList apConsRef =
(NodeList) xpath.evaluate(apConRef,
                        xmldoc,
        XPathConstants.NODESET);

                String aRefQ =
"//construction[@id='"
+apConsRef.item(0).getNodeValue()+"']/constructionThickness/node()";
                apCon = (NodeList)
xpath.evaluate(aRefQ, xmldoc,
        XPathConstants.NODESET);
                String aResQ =
"//construction[@id='" +apConsRef.item(0).getNodeValue()+"']/resistance/node()";
                apRes = (NodeList)
xpath.evaluate(aResQ, xmldoc,
        XPathConstants.NODESET);
                String aMatQ =
"//construction[@id='" +apConsRef.item(0).getNodeValue()+"']/name/node()";
                apMat = (NodeList)
xpath.evaluate(aMatQ, xmldoc,
```

```
        XPathConstants.NODESET);


thickness

apType.item(0).getNodeValue();

apId.item(0).getNodeValue();

apMat.item(0).getNodeValue();

Double.valueOf(apCon.item(0).getNodeValue());

Double.valueOf(apRes.item(0).getNodeValue());


horizontaly

getObjectNormal(apCords);


<0){



object

i+=12)  {

ArrayList<String>();

SbmElement();









apid+"   "+String.valueOf(i);



aptype.toLowerCase();


apertureElement.Type;

sectionCounter;

spaceCounter;
```

```
}

//get construction type,id and

String aptype =

String apid =

String apmat =

Double apcon =

Double apres =


//check if wall planes need shifting

Vector3d aobjNormal =

boolean ainwards = false;

if(aobjNormal.x > 0 || aobjNormal.y

        ainwards = true;
} else ainwards = false;

// consider multiple holes in same


for(int i=0; i<apCords.getLength();

ArrayList<String> tempcords = new

SbmElement apertureElement = new


for(int k=0; k<12;k++){
        String x = cords.get(k+i);
        tempcords.add(x);
}
apertureElement.hCords = tempcords;
apertureElement.id=

apertureElement.Thickness = apcon;
apertureElement.Material = apmat;
apertureElement.Resistance = apres;
apertureElement.Type =

apertureElement.Entity = "hole";
apertureElement.Name =

apertureElement.Section =

apertureElement.Space =

apertureElement.Inwards = ainwards;
apertureElement.Nodes = apCords;
apertureElement.Normal = aobjNormal;
apertureElement.Normal.normalize();
apertureElement.LCpoint = false;
apertureElement.RCpoint = false;

tempcomponents.add(apertureElement);

}
}
```

```
                                        //get construction type,id and thickness
                                        String Enctype =
EncType.item(0).getNodeValue();
                                        String cid = Coreid.item(0).getNodeValue();
                                        String material =
ConMat.item(0).getNodeValue();
                                        Double Thickness =
Double.valueOf(ConsT.item(0).getNodeValue());
                                        Double Resistance =
Double.valueOf(ConRes.item(0).getNodeValue());


                                        //check if wall planes need shifting
horizontaly
                                        Vector3d objNormal =
getObjectNormal(nodes);
                                        boolean inwards = false;

                                        if(objNormal.x > 0 || objNormal.y <0){
                                                inwards = true;
                                        } else inwards = false;



                                        /////// Creating an instance of the SBM
element object
                                        SbmElement thiselement = new SbmElement();

                                        thiselement.id= cid;
                                        thiselement.Material = material;
                                        thiselement.Thickness = Thickness;
                                        thiselement.Resistance = Resistance;
                                        thiselement.Type = Enctype.toLowerCase();
                                        thiselement.Entity = "solid";
                                        thiselement.Name = thiselement.Type;
                                        thiselement.Section = sectionCounter;
                                        thiselement.Space = spaceCounter;
                                        thiselement.Inwards = inwards;
                                        thiselement.Nodes = nodes;
                                        thiselement.Holes = holes;
                                        thiselement.Normal = objNormal;
                                        thiselement.Normal.normalize();
                                        thiselement.LCpoint = false;
                                        thiselement.RCpoint = false;

                                        components.add(thiselement);




                                        /// find Floor(section) bounding
coordinates
                                        if
(thiselement.Type.equals("outdoorceiling")|| thiselement.Type.equals("groundfloor")||

        thiselement.Type.equals("floorceiling") ||
thiselement.Type.equals("basementceiling") || thiselement.Type.equals("gableroof"))
                                        {
                                                thiselement.main();

        //System.out.println(thiselement.Type+" "+thiselement.id);

                                                for (int i=0;i<11;i+=3){
                                                Point3d Vertex = new Point3d();

        Vertex.set(Double.valueOf(thiselement.Nodes.item(i).getNodeValue()),Double.valueO
f(thiselement.Nodes.item(i+1).getNodeValue()),Double.valueOf(thiselement.Nodes.item(i+2)
.getNodeValue()));
                                                SlabCords.add(Vertex);
```

```
        //System.out.println(Vertex.toString());
                                                }
                                        }


                                        /// find External walls coordinates
                                        if
((thiselement.Type.equals("externaladiabaticwall")||

        thiselement.Type.equals("externalloadbearingwall")) )
                                        {

                                                thiselement.main();

                                                for (int i=0;i<11;i+=9){
                                                Point3d Vertex = new Point3d();

        Vertex.set(Double.valueOf(thiselement.Nodes.item(i).getNodeValue()),Double.valueO
f(thiselement.Nodes.item(i+1).getNodeValue()),Double.valueOf(thiselement.Nodes.item(i+2)
.getNodeValue()));
                                                buildingPerimCoords.add(Vertex);
                                                }


        allPerimEdges.addAll(buildSlabEdges(buildingPerimCoords));
                                                buildingPerimCoords.clear();
                                        }

                                        /// find roof coordinates
                                        if ( thiselement.Type.equals("gableroof"))
                                        {

                                                for (int
i=0;i<thiselement.Nodes.getLength();i+=3){
                                                Point3d Vertex = new Point3d();

        Vertex.set(Double.valueOf(thiselement.Nodes.item(i).getNodeValue()),Double.valueO
f(thiselement.Nodes.item(i+1).getNodeValue()),Double.valueOf(thiselement.Nodes.item(i+2)
.getNodeValue()));
                                                roofCords.add(Vertex);

        //System.out.println(Vertex.toString());
                                                }
                                                Point3d Vertex = new Point3d();

        Vertex.set(Double.valueOf(thiselement.Nodes.item(0).getNodeValue()),Double.valueO
f(thiselement.Nodes.item(0+1).getNodeValue()),Double.valueOf(thiselement.Nodes.item(0+2)
.getNodeValue()));
                                                roofCords.add(Vertex);

        allRoofEdges.addAll(buildSlabEdges(roofCords));
                                                roofCords.clear();

                                                roofResistance =
thiselement.Resistance;
                                        }


                                }
                        }


                }


                SbmElement BObject;
                SbmElement compObject;
                SbmSection BSection;
```

```
                        double externalWallThickness = 0.0;
                        double groundWallThickness = 0.0;


                        ////populate Bsections
                        Set<Double> Zvalues = new TreeSet<Double>();
                        for (int k=0;k<SlabCords.size(); k++)
                                if(Zvalues.add(SlabCords.get(k).z))
                                {
                                        SbmSection inits = new SbmSection();
                                        inits.Zvalue = SlabCords.get(k).z;
                                        Bsections.add(inits);
                                }



                        ////populate PerimeterNodes of the current section
                        allPerimEdges = tidyEdges(allPerimEdges);
                        allPerimEdges = reduceEdges(allPerimEdges);
                        List<edge> listOfPerimEdges= new ArrayList<edge>();
                        listOfPerimEdges.addAll(allPerimEdges);
                        edgeComparator com = new edgeComparator();
                        Collections.sort(listOfPerimEdges, com);

                        Set<Point3d> cornerPoints = new TreeSet<Point3d>(new
pointComparator());
                        SbmSection currentfloor = new SbmSection();
                        //Vector3d centroid = new Vector3d();
                        Point3d temppoint = new Point3d();
                        int vertices = 0;


                        perim.moveTo(listOfPerimEdges.get(0).v1.x,
listOfPerimEdges.get(0).v1.y);
                        for (int k=0;k<listOfPerimEdges.size(); k++)
                                //if(cornerPoints.add(buildingPerimCoords.get(k)))

                                {
                                        currentfloor =
Bsections.get(findZ(Bsections,listOfPerimEdges.get(k).v1.z));

        currentfloor.PerimeterNodes.add(listOfPerimEdges.get(k).v1.toString());
                                        perim.lineTo(listOfPerimEdges.get(k).v1.x,
listOfPerimEdges.get(k).v1.y);
                                        temppoint.x = listOfPerimEdges.get(k).v1.x;
                                        temppoint.y = listOfPerimEdges.get(k).v1.y;
                                        centroid.add(temppoint);
                                        vertices++;
                                }
                        perim.closePath();
                        centroid.scale(1D/vertices);


                        // set camera target coords

                        Origin.item(0).setNodeValue(String.valueOf(centroid.x));
                        Origin.item(1).setNodeValue(String.valueOf(centroid.x));




                        ////////////////////////////  check for wall overlaps
//////////////////////

                        for(int i=Sections.getLength(); i< components.size(); i++)
                        {
                                BObject = null;
                                BObject = components.get(i);
                                BObject.main();
                                if (BObject.Entity != "hole") {
                                for(int j=Sections.getLength(); j< components.size(); j++)
                                {
```

```
                                        compObject = null;
                                        compObject = components.get(j);
                                        //compObject.main();

                                        if (BObject.id == compObject.id ||
(!BObject.Normal.equals(compObject.Normal)) ) continue ;
                                        //////check for wall overlaps
                                        if
(BObject.Nodes.item(9).getNodeValue().equals(compObject.Nodes.item(0).getNodeValue()) &&

        BObject.Nodes.item(10).getNodeValue().equals(compObject.Nodes.item(1).getNodeValu
e()) &&

        BObject.Nodes.item(11).getNodeValue().equals(compObject.Nodes.item(2).getNodeValu
e()))
                                                if (BObject.RCpoint!=true) BObject.RCpoint
= true;

                                        if
(BObject.Nodes.item(0).getNodeValue().equals(compObject.Nodes.item(9).getNodeValue()) &&

        BObject.Nodes.item(1).getNodeValue().equals(compObject.Nodes.item(10).getNodeValu
e()) &&

        BObject.Nodes.item(2).getNodeValue().equals(compObject.Nodes.item(11).getNodeValu
e()))
                                                if (BObject.LCpoint!=true) BObject.LCpoint
= true;


        if(BObject.Type.equals("internalnonloadbearingwall") ||
BObject.Type.equals("internalloadbearingwall"))
                                        {BObject.LCpoint = true;
                                        BObject.RCpoint = true;}  // if both set to true,
internal walls look ok

                                }

                                ////  CHECK FOR CONVEX CORNERS
                                for(int m=Sections.getLength(); m< components.size(); m++)

                                {
                                        compObject = null;
                                        compObject = components.get(m);

                                        if (BObject.id == compObject.id ||
(BObject.Normal.equals(compObject.Normal)) ) continue ;

                                        if
(BObject.Nodes.item(9).getNodeValue().equals(compObject.Nodes.item(0).getNodeValue()) &&

        BObject.Nodes.item(10).getNodeValue().equals(compObject.Nodes.item(1).getNodeValu
e()))
                                        {
                                                if
(getIntersection(BObject,compObject)!=null)
                                                if
(perim.contains(getIntersection(BObject,compObject).x,getIntersection(BObject,compObject
).y)==false){
                                                        BObject.RConvexC = true;
                                                }
                                        }
                                        if
(BObject.Nodes.item(0).getNodeValue().equals(compObject.Nodes.item(9).getNodeValue()) &&

        BObject.Nodes.item(1).getNodeValue().equals(compObject.Nodes.item(10).getNodeValu
e()))
                                        {
```

```
                                        if
(getIntersection(BObject,compObject)!=null)
                                        if
(perim.contains(getIntersection(BObject,compObject).x,getIntersection(BObject,compObject
).y)==false){
                                                BObject.LConvexC = true;
                                        }
                                }

                        }

                        }
                        // assigning boundary thicknesses
                        if (BObject.Type.equals("externaladiabaticwall")||
BObject.Type.equals("externalloadbearingwall")
                                        || BObject.Type.equals("groundwall"))
                        {
                        if(BObject.Type.equals("groundwall"))
                        groundWallThickness = BObject.Thickness;
                        else
                        externalWallThickness = BObject.Thickness;
                        }


                }
                ////////////// END OF OVERLAP CHECK ////////////////////


                //// populate roof edges
                allRoofEdges = tidyEdges(allRoofEdges);
                //allRoofEdges = offsetRoof(allRoofEdges,externalWallThickness);
                List<edge> listOfRoofEdges= new ArrayList<edge>();
                listOfRoofEdges.addAll(allRoofEdges);
                Collections.sort(listOfRoofEdges, com);
                Collections.reverse(listOfRoofEdges);
                listOfRoofEdges =
roofOperation(listOfRoofEdges,externalWallThickness);


                //// Gable Roof generation and adding roof element to components
list

                AbstractList<String> groofVertices = new LinkedList<String>();
                groofVertices = buildRoof(skeleton(listOfRoofEdges, "gable"));

                SbmElement gableRoof = new SbmElement();
                gableRoof.Name = "Gable Roof";
                gableRoof.Cords = groofVertices;
                gableRoof.id = "compound_gable_roof_01";
                gableRoof.Type = "roof";
                gableRoof.Entity = "solid";
                gableRoof.Resistance = roofResistance;
                gableRoof.Section =
findZ(Bsections,Double.valueOf(groofVertices.get(2)));
                components.add(gableRoof);


                //// Hip Roof generation and adding roof element to components
list
                AbstractList<String> hroofVertices = new LinkedList<String>();
                hroofVertices = buildRoof(skeleton(listOfRoofEdges, "hip"));

                SbmElement hipRoof = new SbmElement();
                hipRoof.Name = "Hip Roof";
                hipRoof.Cords = hroofVertices;
                hipRoof.id = "compound_hip_roof_01";
                hipRoof.Type = "roof";
                hipRoof.Entity = "solid";
                hipRoof.Resistance = roofResistance;
                hipRoof.Section =
findZ(Bsections,Double.valueOf(hroofVertices.get(2)));
                components.add(hipRoof);
```

```
                        ///// adding windows
                        components.addAll(tempcomponents);

                        ///////    Performing position & geometrical corrections
//////////////////

                        for(int k=Sections.getLength(); k< components.size(); k++)
                        {


                                BObject = components.get(k);

                                if(!BObject.Type.equals("roof"))
                                BSection =
Bsections.get(findZ(Bsections,Double.valueOf(BObject.Nodes.item(2).getNodeValue())));

                                //correct slabs section number
                                if(!BObject.Type.equals("roof"))
                                if (BObject.Type.equals("outdoorceiling")||
BObject.Type.equals("floorceiling")  || BObject.Type.equals("groundfloor")
                                                || BObject.Type.equals("gableroof") ||
BObject.Type.equals("basementceiling"))
                                {
                                        for (int i=1; i<Bsections.size();i++){

       if(Double.valueOf(BObject.Nodes.item(2).getNodeValue()) ==
Bsections.get(i).Zvalue)
                                                        BObject.Section = i;
                                        }
                                }


                                //// streching walls
                                if (BObject.Type.equals("externaladiabaticwall")||
BObject.Type.equals("externalloadbearingwall")
                                                || BObject.Type.equals("groundwall"))
                                {
                                BObject.strechXwalls(BObject.Nodes, BObject.Normal);
                                BObject.strechYwalls(BObject.Nodes, BObject.Normal);
                                }

                                if (BObject.Type.equals("internalnonloadbearingwall")||
BObject.Type.equals("internalloadbearingwall"))
                                {
                                BObject.strechXwalls(BObject.Nodes, BObject.Normal);
                                BObject.strechYwalls(BObject.Nodes, BObject.Normal);

                                }


                                /// offsetting slabs
                                if(!BObject.Type.equals("roof"))
                                if (BObject.Type.equals("outdoorceiling")||
BObject.Type.equals("floorceiling")  || BObject.Type.equals("groundfloor")
                                                || BObject.Type.equals("gableroof") ||
BObject.Type.equals("basementceiling"))
                                {

                                        if(BObject.Type.equals("groundfloor") &&
groundWallThickness != 0.0)
                                                floorMargin = groundWallThickness;
                                        else
                                                floorMargin = externalWallThickness;

                                        BObject.offsetFloor(BObject, floorMargin);
                                }
```

```
                                //Performing vertical shift
                                if (!BObject.Type.equals("groundfloor") || BObject.Entity
!= "hole") {
                                        if(!BObject.Type.equals("roof")){
                                                BObject.shiftUp(BObject.Nodes,
shiftAmount(BObject.Section,slabThickness));
                                        if (BObject.Entity != "hole")
                                        BObject.shiftUp(BObject.Holes,
shiftAmount(BObject.Section,slabThickness));
                                        } else
                                                BObject.shiftRoofUp(BObject.Cords,
shiftAmount(BObject.Section,slabThickness));
                                }
                                if (BObject.Entity == "hole")
                                        BObject.shiftWinUp(BObject.hCords,
shiftAmount(BObject.Section,slabThickness));


                                ///world reset
                                if(!BObject.Entity.equals("hole")){
                                if(!BObject.Type.equals("roof")){
                                for (int x=0;x<BObject.Nodes.getLength();x+=3)

        BObject.Nodes.item(x).setNodeValue(String.valueOf(Double.valueOf(BObject.Nodes.it
em(x).getNodeValue())-Double.valueOf(Origin.item(0).getNodeValue())));
                                for (int x=1;x<BObject.Nodes.getLength();x+=3)

        BObject.Nodes.item(x).setNodeValue(String.valueOf(Double.valueOf(BObject.Nodes.it
em(x).getNodeValue())-Double.valueOf(Origin.item(1).getNodeValue())));
                                for (int x=0;x<BObject.Holes.getLength();x+=3)

        BObject.Holes.item(x).setNodeValue(String.valueOf(Double.valueOf(BObject.Holes.it
em(x).getNodeValue())-Double.valueOf(Origin.item(0).getNodeValue())));
                                for (int x=1;x<BObject.Holes.getLength();x+=3)

        BObject.Holes.item(x).setNodeValue(String.valueOf(Double.valueOf(BObject.Holes.it
em(x).getNodeValue())-Double.valueOf(Origin.item(1).getNodeValue())));

                                } else {
                                        for (int x=0;x<BObject.Cords.size();x+=3)
                                                BObject.Cords.set(x,
String.valueOf(Double.valueOf(BObject.Cords.get(x)) -
Double.valueOf(Origin.item(0).getNodeValue())));
                                        for (int x=1;x<BObject.Cords.size();x+=3)
                                                BObject.Cords.set(x,
String.valueOf(Double.valueOf(BObject.Cords.get(x)) -
Double.valueOf(Origin.item(1).getNodeValue())));
                                }
                                } else
                                {
                                        for (int x=0;x<BObject.hCords.size();x+=3)
                                                BObject.hCords.set(x,
String.valueOf(Double.valueOf(BObject.hCords.get(x))-
Double.valueOf(Origin.item(0).getNodeValue())));
                                        for (int x=1;x<BObject.hCords.size();x+=3)
                                                BObject.hCords.set(x,
String.valueOf(Double.valueOf(BObject.hCords.get(x))-
Double.valueOf(Origin.item(1).getNodeValue())));
                                }




                                        //Tesselation of all faces
                                if(!BObject.Type.equals("roof"))
                                {
                                coordinates.addAll(Tesselate(BObject));

                                writeGeometries(jsonWriter, BObject.id, coordinates,
null); }
                                else if(BObject.Name.equals("Gable Roof"))
```

```
                                {
                                        coordinates.addAll(BObject.Cords);
                                        writeGeometries(jsonWriter,
"compound_gable_roof_01", coordinates, "roof");
                                } else if(BObject.Name.equals("Hip Roof"))
                                {
                                        coordinates.addAll(BObject.Cords);
                                        writeGeometries(jsonWriter, "compound_hip_roof_01",
coordinates, "roof");
                                }
                                coordinates.clear();
                        }


                        ////////////////////////////////// Roof   ////////////////////

                        ////// for 2d ouput drawing test
                        // for(int i =0;i<coordinates.size();i+=9){
                                // Path2D temppath = new Path2D.Double();
                                //
temppath.moveTo(Double.valueOf(coordinates.get(i)),Double.valueOf(coordinates.get(i+1)))
;
                                //
temppath.lineTo(Double.valueOf(coordinates.get(i+3)),Double.valueOf(coordinates.get(i+4)
));
                                //
temppath.lineTo(Double.valueOf(coordinates.get(i+6)),Double.valueOf(coordinates.get(i+7)
));
                                // temppath.closePath();
                                // roof.add(temppath);

                        // }



                        ////////////////////////////////// END of Geometry calcs
///////////////

                        //writeGeometries(jsonWriter, coordinates, indices);
                        jsonWriter.endArray();
                        jsonWriter.endObject();
                        writeVisualScenes(jsonWriter, types, components);
                        jsonWriter.endArray();


                        // Append additional custom data to the scene node
                        jsonWriter.name("data").beginObject();
                        jsonWriter.name("bounds");
                        writeBounds(jsonWriter,extents);
                        jsonWriter.name("unit").value("1 centimeter");

        jsonWriter.name("heatingDemand").value(Double.valueOf(hDemand.item(0).getNodeValu
e()));
                        // writeUnit(jsonWriter);
                        jsonWriter.name("ifcTypes");
                        writeIfcTypes(jsonWriter, types);
                        jsonWriter.name("relationships");
                        writeRelationships(jsonWriter, components);
                        // writeIfcTree(jsonWriter);
                        jsonWriter.name("properties");
                        writeProperties(jsonWriter, components);
                        jsonWriter.endObject();
                        jsonWriter.endObject();

                        jsonWriter.flush();
                        jsonWriter.close();

                        long end = System.nanoTime();
                        System.out.print(((end - start) / 1000000) + " ms");
```

```
        } catch (IOException e) {
                e.printStackTrace();
        }

}


static class edgeComparator implements Comparator<edge> {

    @Override
    public int compare(edge o1, edge o2) {
          Point3d p1=o1.v1;
          Point3d p2=o2.v2;

          if(p1.equals(p2))
              return +1;
          else
              return -1;
    }
}


static class pointComparator implements Comparator<Point3d> {

    @Override
    public int compare(Point3d o1, Point3d o2) {
        if(!(o1.x == o2.x && o1.y == o2.y)){
            return -1;
        }

        return 0;
    }
}

static class edge {
        Point3d v1;
        Point3d v2;

        Boolean gWallCandidate;
        Boolean isGWall;

        public  edge(){
                v1 = new Point3d();
                v2 = new Point3d();
                gWallCandidate = false;
                isGWall = false;


        }

        public double getLength (){
                Vector3d vec = new Vector3d();
                vec.sub(this.v2, this.v1);
                return vec.length();
        }


        @Override
    public boolean equals(Object o) {
                // If the object is compared with itself then return true
        if (o == this) {
            return true;
        }

        /* Check if o is an instance of Complex or not
          "null instanceof [type]" also returns false */
        if (!(o instanceof edge)) {
            return false;
        }

        // typecast o to Complex so that we can compare data members
```

```
                edge c = (edge) o;

                // Compare the data members and return accordingly
                if ((this.v1.equals(c.v1) && this.v2.equals(c.v2)) ||
                                (this.v1.equals(c.v2) && this.v2.equals(c.v1)))
                return true;
                else return false;
                }

        }


        public static List<edge> roofOperation(List<edge> edges, double h){


                /////creating JTS offset
                GeometryFactory fact = new GeometryFactory();
                Coordinate[] coords = new Coordinate[edges.size()+1];
                int k = 0;
                for (int i=0; i<edges.size();i++){
                        Coordinate coord = new Coordinate();
                        coord.x = edges.get(i).v1.x;
                        coord.y = edges.get(i).v1.y;
                        coord.z = edges.get(i).v1.z;
                        coords[i] = coord;
                        k++;
                }
                coords[k]=coords[0];
                LinearRing linear = new GeometryFactory().createLinearRing(coords);
                com.vividsolutions.jts.geom.Polygon JTSpoly = new
com.vividsolutions.jts.geom.Polygon(linear, null, fact);
                com.vividsolutions.jts.geom.Polygon envelope = new
com.vividsolutions.jts.geom.Polygon(linear, null, fact);

                // offset operations
                Geometry outer = new com.vividsolutions.jts.geom.Polygon(null,null,fact);
                BufferParameters bp = new BufferParameters();
                //set join style to MITRE (2) for sharp corners
                bp.setJoinStyle(2);
                BufferOp buff = new BufferOp(JTSpoly,bp);
                outer = buff.getResultGeometry(h); // JTS sets point order back to CW!

                int count = outer.getNumPoints()-1;
                edges.clear();
                for(int i=0;i<outer.getNumPoints()-1;i++){

                        edge newEdge = new edge();
                        newEdge.v1.x = outer.getCoordinates()[count].x;
                        newEdge.v1.y = outer.getCoordinates()[count].y;
                        newEdge.v1.z = coords[1].z;

                        if(i == 0)
                        {
                                newEdge.v2.x = outer.getCoordinates()[i+1].x;
                                newEdge.v2.y = outer.getCoordinates()[i+1].y;
                                newEdge.v2.z = coords[1].z;
                        } else
                        {
                                newEdge.v2.x = outer.getCoordinates()[count+1].x;
                                newEdge.v2.y = outer.getCoordinates()[count+1].y;
                                newEdge.v2.z = coords[1].z;
                        }
                        count--;
                        edges.add(newEdge);
                }


                // find gable edges
                envelope = (com.vividsolutions.jts.geom.Polygon) outer.getEnvelope();
```

```
		Coordinate[] envCoords = envelope.getCoordinates();
		double envWidth = envCoords[3].x - envCoords[0].x;
		double envHeight = envCoords[1].y - envCoords[0].y;

		for (int i=0; i<edges.size();i++){  // find possible edges

			if(edges.get(i).v1.x == edges.get(i).v2.x
					&& edges.get(i).v1.x == envCoords[0].x)
edges.get(i).gWallCandidate = true;
			if(edges.get(i).v1.x == edges.get(i).v2.x
					&& edges.get(i).v1.x == envCoords[2].x)
edges.get(i).gWallCandidate = true;
			if(edges.get(i).v1.y == edges.get(i).v2.y
					&& edges.get(i).v1.y == envCoords[0].y)
edges.get(i).gWallCandidate = true;
			if(edges.get(i).v1.y == edges.get(i).v2.y
					&& edges.get(i).v1.y == envCoords[2].y)
edges.get(i).gWallCandidate = true;

			if(i<edges.size()-1) {
				if(edges.get(i).v1.x == edges.get(i).v2.x
						&& edges.get(i+1).v1.x ==
edges.get(i+1).v2.x) {
					edges.get(i).gWallCandidate = false;
					edges.get(i+1).gWallCandidate = false;
					i++;
					continue;
				}
				if(edges.get(i).v1.y == edges.get(i).v2.y
						&& edges.get(i+1).v1.y ==
edges.get(i+1).v2.y) {
					edges.get(i).gWallCandidate = false;
					edges.get(i+1).gWallCandidate = false;
					i++;
					continue;
				}
			}

			if(edges.get(i).getLength() == envWidth ||
					edges.get(i).getLength() == envHeight )
edges.get(i).gWallCandidate = false;
		}


		// traverse edges to mark best candidates
		for (int j=0; j<edges.size();j++)
		{
			if (edges.get(j).gWallCandidate == true )
			{
				if(edges.get(j+1).gWallCandidate == true &&
						edges.get(j).getLength() <
edges.get(j+1).getLength())
				{
					edges.get(j+1).isGWall = true;
					j++;
					continue;
				}
				edges.get(j).isGWall = true;
			}
		}

		return edges;

	}


public static AbstractList<edge> buildSlabEdges(AbstractList<Point3d> points){

		AbstractList<edge> edges = new LinkedList<edge>();
		for(int i=0; i<points.size()-1;i++){  // index goes out of bound here !
			edge tempEdge = new edge();
			tempEdge.v1.set(points.get(i));
```

```
                        tempEdge.v2.set(points.get(i+1));
                        edges.add(tempEdge);
                        }
                return edges;
        }

        public static AbstractList<edge> tidyEdges(AbstractList<edge> edges){

                AbstractList<edge> newEdges = new LinkedList<edge>();
                for(int i=0; i<edges.size();i++){  // index goes out of bound here !
                        edge tempEdge = new edge();
                        tempEdge = edges.get(i);
                        if(i==0) newEdges.add(tempEdge);
                        for(int j=0; j<newEdges.size();j++){
                                if(!tempEdge.equals(newEdges.get(j)))
                                {
                                        if (j==newEdges.size()-1){
                                                newEdges.add(tempEdge);
                                                break;
                                        }
                                        continue;}
                                else {
                                        if(i!=0)
                                        newEdges.remove(j);
                                        break;
                                        }
                        }
                }
                return newEdges;
        }

public static AbstractList<edge> reduceEdges(AbstractList<edge> edges){

                AbstractList<edge> newEdges = new LinkedList<edge>();
                for(int i=0; i<edges.size();i++){  // index goes out of bound here !
                        edge tempEdge = new edge();
                        tempEdge = edges.get(i);
                        if(i==0) {
                                newEdges.add(tempEdge);
                                continue;
                        }
                        if(tempEdge.v1.z == newEdges.get(0).v1.z)
                                newEdges.add(tempEdge);
                        }
                return newEdges;
        }



        public static double shiftAmount (int section, double[] thicknesses){
                double value = 0.0;
                for (int i = 1; i<=section;i++)
                        value = value + thicknesses[i];
                return value;
        }


        public static AbstractList<Point3d> sortVertices (AbstractList<Point3d>
vertices){
                AbstractList<Point3d> newVertices = new LinkedList<Point3d>();
                Vector3d centroid = new Vector3d();
                Point3d temppoint = new Point3d();
                for (int i = 0; i<vertices.size(); i++){
                        temppoint.x = vertices.get(i).x;
                        temppoint.y = vertices.get(i).y;
                        centroid.add(temppoint);
                }
                centroid.scale(1D/vertices.size());
```

```
        while (!vertices.isEmpty()) {
            //find the one with the highest angle
            double biggestAngle = 0;
            Point3d biggestVertex = null;
            // Look through all of them
            for (Point3d v : vertices) {
              // Make a vector that points from center
               Vector3d dir = new Vector3d();
               dir.sub(v, centroid);
              // the heading
               float angle = (float) Math.atan2(-dir.y, dir.x);
               double a = -1*angle + Math.PI;
              if (a > biggestAngle) {
                biggestAngle = a;
                biggestVertex = v;
              }
              dir = null;
            }

            // Put the one we found in the new arraylist
            newVertices.add(biggestVertex);
            vertices.remove(biggestVertex);
          }
          //vertices = newVertices;
        return newVertices;
        }


       public static  List<Polygon> skeleton (List<edge> edges, String sw){

               List<Polygon> roofPolygons = new ArrayList<Polygon>();
               List<PolygonPoint> ppoints = new ArrayList<PolygonPoint>();

               // corner points, in CCW order
               List<Corner> corners = new ArrayList<Corner>();
               for(int i=0; i<edges.size();i++)
               {
                       Corner tempCorner = new Corner(0,0,0);
                       tempCorner.set(edges.get(i).v1);
                       corners.add(tempCorner);

               }
                               Loop<Edge> loop1 = new Loop();
                               final LoopL <Edge> out = new LoopL();
                               Machine machine = new Machine();

                               // create seperate machines for every edge to cinsider
gable walls
                               // Edge angle as machine argument in PI
                               int m = 0;
                               for ( int j=0;j<edges.size();j++)
                   {

                       if (j==corners.size()-1) m = 0; else m = j+1;
                       Edge e = new Edge (corners.get(j), corners.get(m),Math.PI / 4 );

                       if (edges.get(m).isGWall && sw.equals("gable"))
                               machine = new Machine (0.01);
                               else
                               machine = new Machine (2*Math.PI/6);

                       e.machine = machine;

                       loop1.append( e );
                   }


                               out.add(loop1);

                               Skeleton skel = new Skeleton (out, true);
                               skel.skeleton();
```

```
                                for ( Face face : skel.output.faces.values() )
                                {

                                        for (Loop<Point3d>  lp3 : face.points)
                                                for (Point3d pt : lp3)
                                                {
                                                        ppoints.add(new
PolygonPoint(pt.x,pt.y,pt.z));
                                                }
                                        Polygon eachFace = new Polygon(ppoints);
                                        roofPolygons.add(eachFace);
                                        ppoints.clear();

                                }
                                return(roofPolygons);

        }


        public static  NodeList returnRefSpace (XPath xpath, Document xmldoc, NodeList
space, int sec, int index) throws XPathExpressionException
        {

                NodeList enclosures = (NodeList) null;
                if (space.item(index-1).hasChildNodes() == false)
                {
                        NodeList refSpaces = (NodeList)
xpath.evaluate("(//org.configurator.semergy.sbm.geometry.Section["+sec+"]//org.configura
tor.semergy.sbm.geometry.Space)["+index+"]/@reference",
                                        xmldoc, XPathConstants.NODESET);
                        String refId = "\""+refSpaces.item(0).getNodeValue()+"\"";

                        enclosures = (NodeList) xpath.evaluate("//enclosingSpace[@id="+
refId +"]/enclosures/node()",
                                        xmldoc, XPathConstants.NODESET);
                } else

                          enclosures = (NodeList)
xpath.evaluate("(//org.configurator.semergy.sbm.geometry.Section["+sec+"]//org.configura
tor.semergy.sbm.geometry.Space)["+index+"]/enclosures/node()",
                                xmldoc, XPathConstants.NODESET);

                return enclosures;

        }

        public static  String returnRefSpacePath (XPath xpath, Document xmldoc, NodeList
space, int sec, int index) throws XPathExpressionException
        {

                        NodeList refSpaces = (NodeList)
xpath.evaluate("(//org.configurator.semergy.sbm.geometry.Section["+sec+"]//org.configura
tor.semergy.sbm.geometry.Space)["+index+"]/@reference",
                                        xmldoc, XPathConstants.NODESET);
                        String refId = "\""+refSpaces.item(0).getNodeValue()+"\"";

                        return "//enclosingSpace[@id="+ refId +"]/enclosures";

        }


        private static void writeBounds(JsonWriter jsonWriter, Double ex) throws
IOException {
         jsonWriter.beginArray();
         jsonWriter.value(ex);
         jsonWriter.value(ex);
         jsonWriter.value(ex);
         jsonWriter.endArray();
        }
```

```
        private static void writeIfcTypes(JsonWriter outps, Set<String> type) throws
IOException {
         outps.beginArray();
         @SuppressWarnings("rawtypes")
               Iterator it=type.iterator();

         while(it.hasNext())
         {
           String value=(String)it.next();
           outps.value(value);
         }
        outps.endArray();
 }


        private static void writeRelationships(JsonWriter jsonWriter,
AbstractList<SbmElement> elements) throws IOException {

                int s = 0;
                int maxs = 4;
                int i = 0;


                for (int l=0; l<elements.size();l++) {
                if( elements.get(l).Type.equals("floorceiling") ||
elements.get(l).Type.equals("outdoorceiling")
                                || elements.get(l).Type.equals("roof") ||
elements.get(l).Type.equals("basementceiling") &&
                                elements.get(l).Section != maxs       )
                        elements.get(l).Section = elements.get(l).Section+1;

                }

                jsonWriter.beginArray();

                for (int j=0; j<1; j++){
                                wRelatedObj(jsonWriter, elements, i ,j, maxs);
                }

                jsonWriter.endArray();
               }


        public static void wRelatedObj(JsonWriter jsonWriter, AbstractList<SbmElement>
elements,int i, int section, int maxsec) throws IOException{
                jsonWriter.beginObject();
                jsonWriter.name("type").value(elements.get(i).Type);
                jsonWriter.name("name").value(elements.get(i).Name);
                jsonWriter.name("id").value(elements.get(i).id);
         jsonWriter.name("decomposedBy");
         wDecomposedBy(jsonWriter, elements,i, section,maxsec);
         jsonWriter.name("definedBy");
         wDefinedBy(jsonWriter, elements, i,section, maxsec);
         jsonWriter.name("contains");
         wContains(jsonWriter, elements,section);
         jsonWriter.endObject();

 }

        public static void wDecomposedBy(JsonWriter jsonWriter, AbstractList<SbmElement>
elements,int i,int s,int maxs) throws IOException{
                jsonWriter.beginArray();
                if (s<maxs)
                wRelatedObj(jsonWriter, elements, i,s+1,maxs);
                jsonWriter.endArray();

        }

       public static void wDefinedBy(JsonWriter jsonWriter, AbstractList<SbmElement>
elements,int i,int s,int maxs) throws IOException{
               jsonWriter.beginArray();
```

```
                if (s < maxs && s!=0)
                wRelatedObj(jsonWriter, elements, i,s+1,maxs);
                jsonWriter.endArray();
         }

         public static void wContains(JsonWriter jsonWriter, AbstractList<SbmElement>
elements, int s) throws IOException{

                jsonWriter.beginArray();
                for (int i=0; i<=elements.size()-1;i++)
                {
                       if(elements.get(i).Section == s)
                              wProducts(jsonWriter, elements,i);
                }

                jsonWriter.endArray();

         }

        public static void wProducts(JsonWriter jsonWriter, AbstractList<SbmElement>
elements,int i) throws IOException{
                jsonWriter.beginObject();
                jsonWriter.name("type").value(elements.get(i).Type);
                jsonWriter.name("name").value(elements.get(i).Name);
                jsonWriter.name("id").value(elements.get(i).id);
                jsonWriter.endObject();
         }

        private static void writeMaterials(JsonWriter jsonWriter,Set<String> types)
throws IOException {


                @SuppressWarnings("rawtypes")
                       Iterator it=types.iterator();

                while(it.hasNext())
                {
                  String mvalue=(String)it.next();

                  jsonWriter.beginObject();
                  jsonWriter.name("type").value("material");
                  jsonWriter.name("coreId").value(mvalue);
                      if (mvalue.equals("externaladiabaticwall")  ||
mvalue.equals("externalloadbearingwall") || mvalue.equals("internalnonloadbearingwall"))

        jsonWriter.name("baseColor").beginObject().name("r").value(1).name("g").value(1).
name("b").value(0.8).endObject();

                      else if (mvalue.equals("externalwindow"))

jsonWriter.name("baseColor").beginObject().name("r").value(0.2).name("g").value(0.2).nam
e("b").value(0.8).endObject();
                      else if (mvalue.equals("roof"))

jsonWriter.name("baseColor").beginObject().name("r").value(0.9).name("g").value(0.3).nam
e("b").value(0.17).endObject();

                      else

jsonWriter.name("baseColor").beginObject().name("r").value(0.7).name("g").value(0.7).nam
e("b").value(0.7).endObject();

                      if (mvalue.equals("externalwindow"))
                    jsonWriter.name("alpha").value(0.2);
                      else
                    jsonWriter.name("alpha").value(1);
                    jsonWriter.name("emit").value(0.0);
                    jsonWriter.endObject();
                    }
```

```
}

        public static int findZ(AbstractList<SbmSection> Sec, Double z)
        {
                int out = 0;
                for(int i=0; i<Sec.size();i++){
                        if(Sec.get(i).Zvalue != z) continue;
                        else out = i;
                }
                return out;

        }
         private static void writeGeometries(JsonWriter jsonWriter, String
id,AbstractList<String> nodes, String sw) throws IOException, XPathExpressionException,
ParserConfigurationException, SAXException {

                 int nodeCount = nodes.size();
                 jsonWriter.beginObject();
                 jsonWriter.name("type").value("geometry");
                 jsonWriter.name("coreId").value(id);
                 jsonWriter.name("primitive").value("triangles");
                 jsonWriter.name("positions").beginArray();
                 for (int i=0; i < nodeCount; i++) {
                        jsonWriter.value(getPositions(i,nodes));
                 }
                 jsonWriter.endArray();
                 jsonWriter.name("normals").beginArray();

                 float[] normals ;
                 if (sw != "roof")
                 normals = getNormals(nodes);
                 else normals = getRoofNormals(nodes);

                 for (int i=0; i < normals.length; i++) {
                        jsonWriter.value(normals[i]);
                 }
                 jsonWriter.endArray();

                 jsonWriter.name("indices").beginArray();

                 for (int i = 0; i < nodeCount/3; i++)
                        jsonWriter.value(i);



                 jsonWriter.endArray();
                 jsonWriter.endObject();
         }




         private static double getPositions(int c,AbstractList<String> nodes)

                        throws ParserConfigurationException, SAXException,
                        IOException, XPathExpressionException {

                        return(Double.valueOf(nodes.get(c)));

         }

         private static Vector3d getObjectNormal(NodeList fnodes)
         {
                                        Vector3d v1 = new Vector3d();
                                Vector3d v2 = new Vector3d();
                                Vector3d objectNormal = new Vector3d();
                                Point3d pa = new Point3d();
                                Point3d pb = new Point3d();
                                Point3d pc = new Point3d();
```

```
pa.set(Float.valueOf(fnodes.item(0).getNodeValue()),Float.valueOf(fnodes.item(1).getNode
Value()),Float.valueOf(fnodes.item(2).getNodeValue()));

pb.set(Float.valueOf(fnodes.item(3).getNodeValue()),Float.valueOf(fnodes.item(4).getNode
Value()),Float.valueOf(fnodes.item(5).getNodeValue()));

pc.set(Float.valueOf(fnodes.item(6).getNodeValue()),Float.valueOf(fnodes.item(7).getNode
Value()),Float.valueOf(fnodes.item(8).getNodeValue()));

                        v1.sub(pb, pa);
                        v2.sub(pc, pa);
                    objectNormal.cross(v2, v1);
                    return objectNormal;

        }

        private static Point2d getIntersection(SbmElement obj1, SbmElement obj2)
        {
                    Point2d pa = new Point2d();
              Point2d pb = new Point2d();
              Point2d pc = new Point2d();
              Point2d pd = new Point2d();
              Point2d p1 = new Point2d();
              Point2d p2 = new Point2d();
              Point2d Intersection = new Point2d();

pa.set(Float.valueOf(obj1.Nodes.item(0).getNodeValue()),Float.valueOf(obj1.Nodes.item(1)
.getNodeValue()));

pb.set(Float.valueOf(obj1.Nodes.item(9).getNodeValue()),Float.valueOf(obj1.Nodes.item(10
).getNodeValue()));

pc.set(Float.valueOf(obj2.Nodes.item(0).getNodeValue()),Float.valueOf(obj2.Nodes.item(1)
.getNodeValue()));

pd.set(Float.valueOf(obj2.Nodes.item(9).getNodeValue()),Float.valueOf(obj2.Nodes.item(10
).getNodeValue()));


              p1.set(pa.x+(pb.x-pa.x)/2,pa.y+(pb.y-pa.y)/2);
              p2.set(pc.x+(pd.x-pc.x)/2,pc.y+(pd.y-pc.y)/2);



              double a = obj1.Normal.x;
              double b = -obj2.Normal.x;
              double A = obj1.Normal.y;
              double B = -obj2.Normal.y;
              double c = p2.x - p1.x;
              double C = p2.y - p1.y;
              double det = a*B - b*A;
              double t = -(b*C-B*c)/det;
              double s = (a*C-A*c)/det;

              if (Double.isInfinite((p1.x+t*a)) || Double.isInfinite((p1.y+t*A))
                            || Double.isNaN((p1.x+t*a))  ||  Double.isNaN((p1.y+t*A)))
                    return null;
              else {
              Intersection.set((p1.x+t*a), (p1.y+t*A));
              return Intersection;
              }


        }

        private static float[] getNormals(AbstractList<String> nodes)
```

```
                        throws ParserConfigurationException, SAXException,
                         IOException, XPathExpressionException {

                               float normal[] = new float[nodes.size()];

                               Vector3f v1 = new Vector3f();
                           Vector3f v2 = new Vector3f();
                           Vector3f faceNormal = new Vector3f();
                           Point3f pa = new Point3f();
                           Point3f pb = new Point3f();
                           Point3f pc = new Point3f();
                           boolean IsYZ = false;
                           boolean IsXZ = false;
                           boolean IsXY = false;

                               int upl = nodes.size()/3;
                    // steping through all the faces
                    for (int i=0; i<upl; i+=3) {

                        int ja = i *3 ;

pa.set(Float.valueOf(nodes.get(ja)),Float.valueOf(nodes.get(ja+1)),Float.valueOf(nodes.g
et(ja+2)));
                        int jb = ((i+1)) *3 ;

pb.set(Float.valueOf(nodes.get(jb)),Float.valueOf(nodes.get(jb+1)),Float.valueOf(nodes.g
et(jb+2)));
                        int jc = ((i+2)) *3;

pc.set(Float.valueOf(nodes.get(jc)),Float.valueOf(nodes.get(jc+1)),Float.valueOf(nodes.g
et(jc+2)));


                        if(pa.x == pb.x && pb.x == pc.x ) IsYZ = true;
                        if(pa.y == pb.y && pb.y == pc.y ) IsXZ = true;
                        if(pa.z == pb.z && pb.z == pc.z ) IsXY = true;

                        v1.sub(pb, pa);
                        v2.sub(pc, pa);

                        if (IsYZ){
                           if (i%2 == 0)
                                   faceNormal.cross(v1, v2);
                           else
                                   faceNormal.cross(v2, v1);

                        }

                        if (IsXZ){
                           if (i%2 == 0)
                                   faceNormal.cross(v1, v2);
                           else
                                   faceNormal.cross(v2, v1);
                        }

                        if (IsXY){
                           if (i%2 == 0)
                                   faceNormal.cross(v1, v2);
                           else
                                   faceNormal.cross(v2, v1);
                        }

                        if (faceNormal.length()==0) continue;
                        faceNormal.normalize();


                       /* // suppose the faces are nearly flat, with small deflection
                        // check the scalar product is >0, so each face is the same
orientation
                        // otherwise, reverse the normal
                        double scalara = normal[ja] * faceNormal.x + normal[ja+1] *
faceNormal.y + normal[ja+2] * faceNormal.z;
```

```
                    double scalarb = normal[jb] * faceNormal.x + normal[jb+1] *
faceNormal.y + normal[jb+2] * faceNormal.z;
                    double scalarc = normal[jc] * faceNormal.x + normal[jc+1] *
faceNormal.y + normal[jc+2] * faceNormal.z;
                    // don't know which point to check, suppose it wouldbe reversed on
all points
                    // hack : sum them all, but it doesn not mean much mathematically
                   if (scalara + scalarb + scalarc <0) {
                        faceNormal.x = -faceNormal.x;
                        faceNormal.y = -faceNormal.y;
                        faceNormal.z = -faceNormal.z;
                    }*/


                    // Sum the normals on each vertex, re-normalize at the end
                    normal[ja] += (float)faceNormal.x; normal[ja+1] +=
(float)faceNormal.y; normal[ja+2] += (float)faceNormal.z;
                    normal[jb] += (float)faceNormal.x; normal[jb+1] +=
(float)faceNormal.y; normal[jb+2] += (float)faceNormal.z;
                    normal[jc] += (float)faceNormal.x; normal[jc+1] +=
(float)faceNormal.y; normal[jc+2] += (float)faceNormal.z;

                        IsYZ = false;
                        IsXZ = false;
                        IsXY = false;

                }

                return(normal);

        }


         private static float[] getRoofNormals(AbstractList<String> nodes)

                        throws ParserConfigurationException, SAXException,
                        IOException, XPathExpressionException {

                float normal[] = new float[nodes.size()];

                Vector3f v1 = new Vector3f();
                Vector3f v2 = new Vector3f();
                Vector3f faceNormal = new Vector3f();
                Point3f pa = new Point3f();
                Point3f pb = new Point3f();
                Point3f pc = new Point3f();

                int upl = nodes.size()/3;
                // steping through all the faces
                for (int i=0; i<upl; i+=3) {

                        int ja = i *3 ;

pa.set(Float.valueOf(nodes.get(ja)),Float.valueOf(nodes.get(ja+1)),Float.valueOf(nodes.g
et(ja+2)));
                        int jb = ((i+1)) *3 ;

pb.set(Float.valueOf(nodes.get(jb)),Float.valueOf(nodes.get(jb+1)),Float.valueOf(nodes.g
et(jb+2)));
                        int jc = ((i+2)) *3;

pc.set(Float.valueOf(nodes.get(jc)),Float.valueOf(nodes.get(jc+1)),Float.valueOf(nodes.g
et(jc+2)));

                        v1.sub(pb, pa);
                        v2.sub(pc, pa);

                        faceNormal.cross(v1, v2);
                        if (faceNormal.length()==0) continue;
```

```
                        faceNormal.normalize();

                        // Sum the normals on each vertex, re-normalize at the end
                        normal[ja] += (float)faceNormal.x; normal[ja+1] +=
(float)faceNormal.y; normal[ja+2] += (float)faceNormal.z;
                        normal[jb] += (float)faceNormal.x; normal[jb+1] +=
(float)faceNormal.y; normal[jb+2] += (float)faceNormal.z;
                        normal[jc] += (float)faceNormal.x; normal[jc+1] +=
(float)faceNormal.y; normal[jc+2] += (float)faceNormal.z;

                }
                return(normal);
        }


        private static void writeVisualScenes(JsonWriter jsonWriter, Set<String>
types,AbstractList<SbmElement> elements) throws IOException{
                jsonWriter.beginObject()
                .name("type").value("lookAt")
                .name("id").value("main-lookAt")
                .name("eye")
                .beginObject()
                .name("x").value(12)
                .name("y").value(43)
                .name("z").value(30)
                .endObject()
                .name("look")
                .beginObject()
                .name("x").value(2)
                .name("y").value(2)
                .name("z").value(0)
                .endObject()
                .name("up")
                .beginObject()
                .name("x").value(0.0)
                .name("y").value(0.0)
                .name("z").value(1.0)
                .endObject()
                .name("nodes")
                .beginArray()
                .beginObject()
                .name("type").value("camera")
                .name("id").value("main-camera")
                .name("optics")
                .beginObject()
                .name("type").value("perspective")
                .name("far").value(2930)
                .name("near").value(2)
                .name("aspect").value(1.0)
                .name("fovy").value(37.8493)
                .endObject()
                .name("nodes")
                .beginArray()
                .beginObject()
                .name("type").value("renderer")
                .name("id").value("main-renderer")
                .name("clear")
                .beginObject()
                .name("color").value(true)
                .name("depth").value(true)
                .name("stencil").value(true)
                .endObject()
                .name("clearColor")
                .beginObject()
                .name("r").value(0.2)
                .name("g").value(0.2)
                .name("b").value(0.2)
                .name("a").value(0.2)
                .endObject()
                .name("nodes")
                .beginArray()
```

```
                .beginObject()
                .name("type").value("light")
                .name("id").value("sun-light")
                .name("mode").value("dir")
                .name("color")
                .beginObject()
                .name("r").value(0.8)
                .name("g").value(0.8)
                .name("b").value(0.8)
                .endObject()
                .name("dir")
                .beginObject()
                .name("x").value(-0.5)
                .name("y").value(-0.5)
                .name("z").value(-1.0)
                .endObject()
                .name("diffuse").value(true)
                .name("specular").value(true)
                .endObject();

                @SuppressWarnings("rawtypes")
                       Iterator it=types.iterator();

                 while(it.hasNext())
                 {
                   String value=(String)it.next();
                   writeNodes(jsonWriter,elements, value);
                 }

                jsonWriter.endArray()
                .endObject()
                .endArray()
                .endObject()
                .endArray()
                .endObject();

        }

        private static void writeNodes(JsonWriter jsonWriter, AbstractList<SbmElement>
elements, String type ) throws IOException {


                // Output each geometry, grouped by material

                 jsonWriter.beginObject();
                 jsonWriter.name("type").value("tag");
                 jsonWriter.name("tag").value(type);
                 jsonWriter.name("id").value(type);
                 jsonWriter.name("nodes");
                 jsonWriter.beginArray();

                         jsonWriter.beginObject();

                         jsonWriter.name("type").value("material");
                         jsonWriter.name("coreId").value(type);
                         jsonWriter.name("nodes");
                         jsonWriter.beginArray();

                         for (int i=0; i<elements.size();i++) {

                               if(elements.get(i).Type.equals(type) ) {


                                       jsonWriter.beginObject();
                                       jsonWriter.name("type").value("name");
                                       jsonWriter.name("id").value(elements.get(i).id);

jsonWriter.name("res").value(elements.get(i).Resistance);
```

```
jsonWriter.name("nodes").beginArray().beginObject();
                                        jsonWriter.name("type").value("geometry");

jsonWriter.name("coreId").value(elements.get(i).id);
                                        jsonWriter.endObject();
                                        jsonWriter.endArray();
                                        jsonWriter.endObject();
                                  }
                           }

                           jsonWriter.endArray();
                           jsonWriter.endObject();

                 jsonWriter.endArray();
                 jsonWriter.endObject();

         }


private static void writeProperties(JsonWriter jsonWriter, AbstractList<SbmElement>
elements) throws IOException {

                        jsonWriter.beginObject();
                           for (int i=3; i<elements.size();i++) {
                                 jsonWriter.name(elements.get(i).id);
                                 jsonWriter.beginObject();
                                 jsonWriter.name("Name").value(elements.get(i).Name);
                                 jsonWriter.name("Construction
Material").value(elements.get(i).Material);
                                 jsonWriter.name("Thermal
Resistance").value(elements.get(i).Resistance+" (m<sup>2</sup>K)/W");

jsonWriter.name("Thickness").value(elements.get(i).Thickness+" m");
                                 jsonWriter.endObject();
                           }

             jsonWriter.endObject();

         }


         public static AbstractList<String> ConvertNodeList (NodeList nodelist)
         {
                AbstractList<String> result = new LinkedList<String>();
                for (int i = 0; i < nodelist.getLength(); ++i)
                {
                       result.add(nodelist.item(i).getNodeValue());
                }
                return result;
         }


         public static String XPathQuery(int o,int s,int i,String swi)
         {

                if (swi == "w")
                return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                 Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                 Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/coordinates/*/*/node()";
                else if(swi == "h")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                 Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                 Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/coordinates/*/*/text()";
                if(swi == "hid")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                 Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
```

```
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/id/text()";
                if(swi == "hcon")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/construction/constructionThickness/node()";
                if(swi == "hconr")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/construction/resistance/text()";
                if(swi == "apconmat")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/construction/name/text()";

                if(swi == "htyp")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/apertureType/text()";


                if(swi == "ah")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/@reference";
                 if (swi == "aw")
                        return "(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/*/coordinates/*/*/text()";
                 if (swi == "en")
                                 return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/enclosureType/node()";
                 if (swi == "con")
                                 return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/construction/constructionThickness/node()";
                 if (swi == "conmat")
                                 return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                                Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/construction/name/node()";

                 if (swi == "conres")
                                 return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                                Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
```

```
                            Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/construction/resistance/node()";


                if (swi == "conr")
                             return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/construction/@reference";
                if (swi == "apconr")
                             return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/apertures/*/construction/@reference";
                if (swi == "fr")
                             return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s)
+"])/enclosures/org.configurator.semergy.sbm.geometry.Floor/@reference";
                if (swi == "id")
                             return
"(//org.configurator.semergy.sbm.geometry.Section[" +
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/id/text()";
                if (swi == "f")
                             return "//org.configurator.semergy.sbm.geometry.Section["
+
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s)
+"]/enclosures/org.configurator.semergy.sbm.geometry.Floor/coordinates/*/*/node()";
                if (swi == "fc")
                             return "//org.configurator.semergy.sbm.geometry.Section["
+
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s)
+"]/enclosures/org.configurator.semergy.sbm.geometry.Floor/construction/constructionThic
kness/node()";
                if (swi == "sec")
                             return
"(//org.configurator.semergy.sbm.geometry.Section)[" +
                                    Integer.toString(i) +
                                    "]//enclosureType/node()";
                if (swi == "apty")
                            return
"(//org.configurator.semergy.sbm.geometry.Section)[" +
                                    Integer.toString(i) +
                                    "]//apertureType/node()";

            else
                    return "(//org.configurator.semergy.sbm.geometry.Section[" +
                            Integer.toString(o)
+"]//org.configurator.semergy.sbm.geometry.Space[" +
                            Integer.toString(s) +"]/enclosures/*)["
+Integer.toString(i) +"]/*/apertures/*/coordinates/*/*/text()";


        }

        public static String XPathRefQuery(String path, int o,int s,int i,String swi)
        {

                if (swi == "w")
```

```
                return "("+path+")[" +Integer.toString(i) +"]/coordinates/*/*/node()";
                else if(swi == "h")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/coordinates/*/*/text()";
                if(swi == "hid")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/id/text()";
                if(swi == "hcon")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/construction/constructionThickness/node()";
                if(swi == "apconmat")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/construction/name/node()";

                if(swi == "hconr")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/construction/resistance/text()";
                if(swi == "htyp")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/apertureType/text()";

                if(swi == "ah")
                        return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/@reference";
                 if (swi == "aw")
                        return "("+path+")[" +Integer.toString(i)
+"]/*/coordinates/*/*/text()";
                 if (swi == "en")
                                return "("+path+")[" +Integer.toString(i)
+"]/enclosureType/node()";
                 if (swi == "con")
                                return "("+path+")[" +Integer.toString(i)
+"]/construction/constructionThickness/node()";
                 if (swi == "conmat")
                                return "("+path+")[" +Integer.toString(i)
+"]/construction/name/node()";

                 if (swi == "conres")
                                return "("+path+")[" +Integer.toString(i)
+"]/construction/resistance/text()";
                 if (swi == "conr")
                                return "("+path+")[" +Integer.toString(i)
+"]/construction/@reference";
                 if (swi == "apconr")
                                return "("+path+")[" +Integer.toString(i)
+"]/apertures/*/construction/@reference";
                 if (swi == "fconr")
                                return
path+"/org.configurator.semergy.sbm.geometry.Floor/construction/@reference";
                 if (swi == "fr")
                                return
"("+path+"/org.configurator.semergy.sbm.geometry.Floor/@reference";
                 if (swi == "id")
                                return "("+path+")[" +Integer.toString(i) +"]/id/text()";
                 if (swi == "f")
                                return
path+"/org.configurator.semergy.sbm.geometry.Floor/coordinates/*/*/node()";
                 if (swi == "fc")
                                return
path+"/org.configurator.semergy.sbm.geometry.Floor/construction/constructionThickness/no
de()";
                 if (swi == "sec")
                                return
"(//org.configurator.semergy.sbm.geometry.Section)[" +
                                        Integer.toString(i) +
                                        "]//enclosureType/node()";

                else
```

```
                        return "("+path+")[" +Integer.toString(i)
+"]/*/apertures/*/coordinates/*/*/text()";


        }


        public static AbstractList<String> Tesselate (SbmElement BuildingObject)
        {
                NodeList nodes = BuildingObject.Nodes;
                NodeList holes = BuildingObject.Holes;
                Double Depth = BuildingObject.Thickness;
                Boolean inw = BuildingObject.Inwards;
                Boolean Lcp = BuildingObject.LCpoint;
                Boolean Rcp = BuildingObject.RCpoint;
                Boolean Lcc = BuildingObject.LConvexC;
                Boolean Rcc = BuildingObject.RConvexC;
                Double StoredX = 0.0;
                Double StoredY = 0.0;
                Double StoredZ = 0.0;
                Polygon polygon;
                Polygon apertures;
                ArrayList<PolygonPoint> ppoints = new
ArrayList<PolygonPoint>(nodes.getLength()/3);
                ArrayList<PolygonPoint> apoints = new ArrayList<PolygonPoint>();
                if (BuildingObject.Entity != "hole") {
                apoints = new ArrayList<PolygonPoint>(holes.getLength()/3);}
                ArrayList<String> apcords = BuildingObject.hCords;

                Float x1 = Float.valueOf(nodes.item(0).getNodeValue());
                Float x2 = Float.valueOf(nodes.item(0+3).getNodeValue());
                Float x3 = Float.valueOf(nodes.item(0+6).getNodeValue());
                Float x4 = Float.valueOf(nodes.item(0+9).getNodeValue());
                Float y1 = Float.valueOf(nodes.item(1).getNodeValue());
                Float y2 = Float.valueOf(nodes.item(1+3).getNodeValue());
                Float y3 = Float.valueOf(nodes.item(1+6).getNodeValue());
                Float y4 = Float.valueOf(nodes.item(1+9).getNodeValue());

                Boolean IsYZ = false;
                Boolean IsXZ = false;
                Boolean IsXY = false;
                Boolean Isinside = false;

                int loopMax = nodes.getLength();
                if (BuildingObject.Entity == "hole") {
                        Float hx1 = Float.valueOf(apcords.get(0));
                        Float hx2 = Float.valueOf(apcords.get(0+3));
                        Float hx3 = Float.valueOf(apcords.get(0+6));
                        Float hx4 = Float.valueOf(apcords.get(0+9));
                        Float hy1 = Float.valueOf(apcords.get(1));
                        Float hy2 = Float.valueOf(apcords.get(1+3));
                        Float hy3 = Float.valueOf(apcords.get(1+6));
                        Float hy4 = Float.valueOf(apcords.get(1+9));
                for (int i = 0; i< 12; i+=3)
                {

                        if(hx1.compareTo(hx2)==0 && hx2.compareTo(hx3)==0 &&
hx3.compareTo(hx4)==0)
                        {
                                IsYZ = true;
                                StoredX = Double.valueOf(apcords.get(i));
                                ppoints.add( new
PolygonPoint(Double.valueOf(apcords.get(i+1)),
                                                Double.valueOf(apcords.get(i+2))));
                        }
                        else if(hy1.compareTo(hy2)==0 && hy2.compareTo(hy3)==0 &&
hy3.compareTo(hy4)==0)
                        {
                                IsXZ = true;
                                StoredY = Double.valueOf(apcords.get(i+1));
                                ppoints.add( new
PolygonPoint(Double.valueOf(apcords.get(i)),
```

```
                                                Double.valueOf(apcords.get(i+2))));
                        }

                        else {
                                IsXY= true;
                                StoredZ = Double.valueOf(apcords.get(i+2));
                                ppoints.add( new
PolygonPoint(Double.valueOf(apcords.get(i)),
                                                Double.valueOf(apcords.get(i+1))));
                        }

                }
        }

                else
                        for (int i = 0; i< loopMax; i+=3)
                        {

                                if(x1.compareTo(x2)==0 && x2.compareTo(x3)==0 &&
x3.compareTo(x4)==0)
                                {
                                        IsYZ = true;
                                        StoredX =
Double.valueOf(nodes.item(i).getNodeValue());
                                        ppoints.add( new
PolygonPoint(Double.valueOf(nodes.item(i+1).getNodeValue()),

Double.valueOf(nodes.item(i+2).getNodeValue())));
                                }
                                else if(y1.compareTo(y2)==0 && y2.compareTo(y3)==0 &&
y3.compareTo(y4)==0)
                                {
                                        IsXZ = true;
                                        StoredY =
Double.valueOf(nodes.item(i+1).getNodeValue());
                                        ppoints.add( new
PolygonPoint(Double.valueOf(nodes.item(i).getNodeValue()),

Double.valueOf(nodes.item(i+2).getNodeValue())));
                                }

                                else {
                                        IsXY= true;
                                        StoredZ =
Double.valueOf(nodes.item(i+2).getNodeValue());
                                        ppoints.add( new
PolygonPoint(Double.valueOf(nodes.item(i).getNodeValue()),

Double.valueOf(nodes.item(i+1).getNodeValue())));
                                }

                        }

                polygon = ( new Polygon(ppoints) );

                if (BuildingObject.Entity != "hole"){
                for (int j = 0; j< holes.getLength(); j+=3)
                {


                        if (IsYZ)

                        {
                                apoints.add( new
PolygonPoint(Double.valueOf(holes.item(j+1).getNodeValue()),

Double.valueOf(holes.item(j+2).getNodeValue())));
                        }
                        else if (IsXZ)
```

```
                        {
                                apoints.add( new
PolygonPoint(Double.valueOf(holes.item(j).getNodeValue()),

Double.valueOf(holes.item(j+2).getNodeValue())));
                        }

                        else

                                apoints.add( new
PolygonPoint(Double.valueOf(holes.item(j).getNodeValue()),

Double.valueOf(holes.item(j+1).getNodeValue())));

                        if((j+3) % 12 == 0 )
                        {

                                apertures = (new Polygon(apoints) );
                                polygon.addHole(apertures);
                                apoints.clear();
                                apertures.clearSteinerPoints();

                        }

                }
        }


                ///this will create sub simplexes/simplices
                Poly2Tri.triangulate( polygon );
                List<DelaunayTriangle> triangles = polygon.getTriangles();



               String[] tempCords = new String[6];

               AbstractList<String> temp = new LinkedList<String>();
               AbstractList<String> result = new LinkedList<String>();

               //////  Creating and writing new coordinates by adding depth to result
               for (int k = 0; k < triangles.size(); ++k)
                       if (IsYZ)
                       {

                               for(int l=0; l<3; l++){

                                       if (inw) {

                                               temp.add(Double.toString(StoredX+Depth));

                                               if ((Lcc) &&
(triangles.get(k).points[l].getXf()==y1 || triangles.get(k).points[l].getXf()==y2))

       temp.add(Float.toString(triangles.get(k).points[l].getXf()+2*Depth.floatValue()))
;
                                               else
                                                       if ((Rcc) &&
(triangles.get(k).points[l].getXf()==y3 || triangles.get(k).points[l].getXf()==y4))

       temp.add(Float.toString(triangles.get(k).points[l].getXf()-
2*Depth.floatValue()));
                                                       else

       temp.add(Float.toString(triangles.get(k).points[l].getXf()));

       temp.add(Float.toString(triangles.get(k).points[l].getYf()));

                                               //adds 2nd layer
                                               //Chamfer corners
```

```
                                                if (triangles.get(k).points[l].getXf()==y1
|| triangles.get(k).points[l].getXf()==y2){
                                                        temp.add(Double.toString(StoredX));
                                                        if (Rcp)
temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));
                                                        else

        temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                                } else
                                                        if
(triangles.get(k).points[l].getXf()==y3 || triangles.get(k).points[l].getXf()==y4){
                                                        temp.add(Double.toString(StoredX));
                                                //      if (Lcp)
temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));
                                                //      else

        temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                                } else

                                                {
                                                temp.add(Double.toString(StoredX));

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                                }

                                        } else {

                                                //Chamfer corners
                                                if (triangles.get(k).points[l].getXf()==y1
|| triangles.get(k).points[l].getXf()==y2){
                                                        temp.add(Double.toString(StoredX));

        temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                                } else
                                                        if
(triangles.get(k).points[l].getXf()==y3 || triangles.get(k).points[l].getXf()==y4){
                                                        temp.add(Double.toString(StoredX));

        temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                                } else {
                                        temp.add(Double.toString(StoredX));

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                                }

                                        //adds 2nd layer
                                        ///*
                                        temp.add(Double.toString(StoredX-Depth));
                                        if ((Lcp)&&(triangles.get(k).points[l].getXf()==y1
|| triangles.get(k).points[l].getXf()==y2))

        temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));
                                        else if
((Rcp)&&(triangles.get(k).points[l].getXf()==y3 ||
triangles.get(k).points[l].getXf()==y4))

        temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));
```

```
                                    else

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                    //*/
                                    }
                            }

                            /////traverse each prism and create 6 triangles of each
                            int q=0;
                            for (int m=0; m<=5; m++)
                            {

                                        for(int p=0; p<=8; p++)
                                        {result.add(temp.get(q%18));
                                        q++;

                                        }
                                        q=q-6;


                            }
                            for (int n=0; n<15; n+=6) {

        result.add(temp.get(n));result.add(temp.get(n+1));result.add(temp.get(n+2));
                            }
                            for (int n=3; n<18; n+=6) {

        result.add(temp.get(n));result.add(temp.get(n+1));result.add(temp.get(n+2));
                            }
                            temp.clear();

                        }
                        else if (IsXZ)
                        {
                                for(int l=0; l<3; l++)

                                if(inw)

                                {
                                if ((Rcp)&&(triangles.get(k).points[l].getXf()==x3
|| triangles.get(k).points[l].getXf()==x4))

        temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));
                                else if
((Lcp)&&(triangles.get(k).points[l].getXf()==x1 ||
triangles.get(k).points[l].getXf()==x2))

        temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));
                                        else


        temp.add(Float.toString(triangles.get(k).points[l].getXf()));
                                temp.add(Double.toString(StoredY-Depth));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                //adds 2nd layer
                                ///*
                                if (triangles.get(k).points[l].getXf()==x1 ||
triangles.get(k).points[l].getXf()==x2){

        temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));
                                        temp.add(Double.toString(StoredY));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                } else
                                        if (triangles.get(k).points[l].getXf()==x3
|| triangles.get(k).points[l].getXf()==x4){

        temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));
```

```
                                                        temp.add(Double.toString(StoredY));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                        } else{

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));
                                        temp.add(Double.toString(StoredY));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                        }
                                        //*/

                                } else

                                {
                                        if (triangles.get(k).points[l].getXf()==x1 ||
triangles.get(k).points[l].getXf()==x2){


        temp.add(Float.toString(triangles.get(k).points[l].getXf()-Depth.floatValue()));
                                                temp.add(Double.toString(StoredY));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));}

                                        if (triangles.get(k).points[l].getXf()==x3 ||
triangles.get(k).points[l].getXf()==x4){

        temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValue()));
                                                temp.add(Double.toString(StoredY));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));}
                                        //adds 2nd layer
                                        //*
                                        if (triangles.get(k).points[l].getXf()==x1 ||
triangles.get(k).points[l].getXf()==x2){

        if(Lcp)temp.add(Float.toString(triangles.get(k).points[l].getXf()-
Depth.floatValue()));
                                                else
if(Lcc)temp.add(Float.toString(triangles.get(k).points[l].getXf()-
2*Depth.floatValue()));
                                                else

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));
                                                temp.add(Double.toString(StoredY+Depth));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                        } else
                                                if (triangles.get(k).points[l].getXf()==x3
|| triangles.get(k).points[l].getXf()==x4){

        if(Rcp)temp.add(Float.toString(triangles.get(k).points[l].getXf()+Depth.floatValu
e()));
                                                        else
if(Rcc)temp.add(Float.toString(triangles.get(k).points[l].getXf()+2*Depth.floatValue()))
;
                                                        else

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));

        temp.add(Double.toString(StoredY+Depth));

        temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                        } else
                                                //*/
                                        {

        temp.add(Float.toString(triangles.get(k).points[l].getXf()));
                                        temp.add(Double.toString(StoredY));
```

```
temp.add(Float.toString(triangles.get(k).points[l].getYf()));


temp.add(Float.toString(triangles.get(k).points[l].getXf()));
                                temp.add(Double.toString(StoredY+Depth));

temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                }
                                //*/

                        }
                        int q=0;
                        for (int m=0; m<=5; m++)
                        {



                                for(int p=0; p<=8; p++)
                                {result.add(temp.get(q%18));
                                q++;

                                }
                                q=q-6;


                        }
                        for (int n=0; n<15; n+=6) {

result.add(temp.get(n));result.add(temp.get(n+1));result.add(temp.get(n+2));
                        }
                        for (int n=3; n<18; n+=6) {

result.add(temp.get(n));result.add(temp.get(n+1));result.add(temp.get(n+2));
                        }
                        temp.clear();

                }
                else if (IsXY)
                {


                        for(int l=0; l<3; l++){

temp.add(Float.toString(triangles.get(k).points[l].getXf()));

temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                temp.add(Double.toString(StoredZ+Depth));
                                //adds 2nd layer
                                ///*


temp.add(Float.toString(triangles.get(k).points[l].getXf()));

temp.add(Float.toString(triangles.get(k).points[l].getYf()));
                                temp.add(Double.toString(StoredZ));
                                //*/

                        }

                        /////traverse each prism and create 6 triangles of each
                        int q=0;
                        for (int m=0; m<=5; m++)
                        {

                                        for(int p=0; p<=8; p++)
                                        {result.add(temp.get(q%18));
                                        q++;

                                        }
                                        q=q-6;
```

```
                        }
                        for (int n=0; n<15; n+=6) {

        result.add(temp.get(n));result.add(temp.get(n+1));result.add(temp.get(n+2));
                        }
                        for (int n=3; n<18; n+=6) {

        result.add(temp.get(n));result.add(temp.get(n+1));result.add(temp.get(n+2));
                        }
                        temp.clear();


                }


               ///rounding of node values, for further calculation of normals
               for (int cc=0; cc<result.size(); cc++)
               {
                       double floatitem = Float.parseFloat(result.get(cc));
                        result.set(cc,
String.valueOf(Math.round(floatitem*100.0)/100.0));
               }

                return result;
         }

         public static AbstractList<String> buildRoof (List<Polygon> faces)
         {
                AbstractList<String> result = new LinkedList<String>();

                for (int i = 0; i< faces.size(); i++)
                {

                        ///this will create sub simplexes/simplices
                        Poly2Tri.triangulate( faces.get(i) );
                        List<DelaunayTriangle> triangles = faces.get(i).getTriangles();

                        //////  Creating and writing new coordinates by adding depth, to
result
                        for (int k = 0; k < triangles.size(); ++k)

                        {
                                for(int l=0; l<3; l++){


result.add(Double.toString(triangles.get(k).points[l].getX()));

result.add(Double.toString(triangles.get(k).points[l].getY()));

result.add(Double.toString(triangles.get(k).points[l].getZ()));

                                }
                        }

                        ///rounding of values, for further calculation of normals
               }
               for (int cc=0; cc<result.size(); cc++)
               {
                        double floatitem = Float.parseFloat(result.get(cc));
                        result.set(cc,
String.valueOf(Math.round(floatitem*100.0)/100.0));
               }
               return result;
         }


} //end of class
```