WEB-BASED PARAMETRIC MODELING FOR ARCHITECTURAL DESIGN AND OPTIMIZATION: CASE STUDIES

A Thesis

by

SEDA TUZUN CANADINC

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2019

Major Subject: Architecture

ABSTRACT

This thesis presents a new framework for Web-based parametric modeling for design collaboration, towards allowing multiple users to work on the shared Web-based model in the process of drafting, design, simulation, and optimization. The framework consists of a WebGL-based model Viewer (Autodesk Forge Viewer), two visual programming tools, Grasshopper and Dynamo, and two software prototypes developed for this thesis. The prototypes, one for Grasshopper, the other for Dynamo, provide the communication between the Viewer and the visual programming tools. The Web-based 3D model and design data can be viewed in the Viewer. The Web-based model is controlled and modified through the visual programming tools using the prototypes. The embodiment of the Web-based information technology, WebGL and networking, makes it possible for users to view the Web-based model, collaborate, and participate in modeling through Web browsers. The full-fledged visual programming environments, Grasshopper and Dynamo, enable users to interact with the parametric Web-based model; the plugins of the visual programming tools allow users to implement building energy performance (BEP) simulation and optimization while the Web-based model enables collaborative design exploration. Two case studies with three tests each were conducted on a simplified residential building model. In Case Study 1, two simulated users (actions done by the author) tested the parametric capabilities of the shared Web-based model using Grasshopper and Dynamo. The basic geometric transformations including scale, translate, and rotate were tested in Tests 1.1, 1.2, and 1.3 respectively. In the Case Study 2, the collaboration of two Grasshopper users on the shared Web-based model in the process of optimization for different building performance objectives -in terms of daylight, energy use, and roof coverage- was tested. Case Study 2 was conducted through three tests of optimization. (i) In Test 2.1, the objectives were maximum preferred daylight and minimum roof shape with shading. (ii) In Test 2.2, the objectives were minimum energy and minimum roof shape with shading. (iii) In Test 2.3, minimum energy use was the first objective, maximum preferred daylight and minimum roof shape with shading was the second objective.

The findings demonstrate that the framework successfully provides interoperable Web-based architectural models which could be controlled parametrically through different visual programming tools. The framework also constitutes an environment of collaboration between Grasshopper users during daylight, energy, and integrated daylight and energy optimization along with the optimization of the roof shape. In both collaboration cases, the users retrieve data from the Web-based model to update their visual programming files. Major technical limitations of the framework are also found and discussed. Future work includes automation of the data flow through Web to the local visual programming files and contributing to the interoperability solutions by incorporating other tools.

DEDICATION

To Dr. Demircan Canadinc

ACKNOWLEDGEMENTS

Foremost, I would like to express my very great appreciation to Dr. Wei Yan. His expertise, encouragement, continuing support, and guidance gave me the power and motivation throughout this study. I would like to thank my committee members, Dr. Liliana Beltran and Dr. Manish Dixit, for their time, insightful comments, and all of their valuable contributions.

I especially want to thank Bihan Wang and Yalong Pi; this thesis could not have been accomplished without their contributions. I am grateful to the Department of Architecture for financially supporting me and giving me scholarship and assistantship. I would like to extend my thanks to Ms. Judy Pruitt and Dr. Ray Pentecost for their kindness. I owe many thanks to the friends, faculty members, and staff at the Department of Architecture and the BIMSIM group with great people that I am happy being a part of it. I am grateful to Drs. Ibrahim and Sena Karaman and Yunus, and Hande Ozcan for their support, help, and friendship. I am very thankful to Somaye Seddighikhavidak for her kindness, friendship, and support.

My deepest gratitude goes to my mother, father, brother, and my aunt. They always believe in me, love me, and endlessly support me. Special thanks to my mother-in-law, father-in-law and sister-in-law for their love and support. Especially, I am grateful to my spouse Dr. Demircan Canadinc. I always feel lucky to have his endless support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee chaired by Professor Wei Yan and the committee includes Professor Liliana Beltran of the Department of Architecture and Professor Manish Dixit of the Department of Construction Science as members.

The scripts for the two plugins were provided by Professor Wei Yan, Yalong Pi, and Bihan Wang.

All other work conducted for the thesis was independently completed by the student.

Funding Sources

This study was supported in part by the Kelley Vrooman Endowed Scholarship from the Department of Architecture at Texas A&M University.

The development of the two software plugins used in this research for Grasshopper and Dynamo to communicate with the Web-based models was funded by the Autodesk Forge Research Grants 2016 and 2018.

NOMENCLATURE

ADO	Architectural Design Optimization
AIA	The American Institute for Architects
ASHRAE	The American Society of Heating, Refrigerating and Air-Conditioning Engineers
API	Application Programming Interface
BEM	Building Energy Modeling
BEP	Building Energy Performance
BIM	Building Information Modeling
Btu	British Thermal Unit
CBECS	Commercial Buildings Energy Consumption Survey
CIE	Commission Internationale de l'Eclairage
DOE	US Department of Energy
EUI	Energy Use Intensity
GUI	Graphical User Interface
HVAC	Heating, Ventilation, and Air Conditioning
IP	Inch Pound, Imperial Units
kBtu	Kilo British Thermal Unit
RAM	Random Access Memory
SI	International System of Units
sqft	Square Foot

U	Heat Transfer Coefficient U
UDI	Useful Daylight Illuminance
USGBC	US Green Building Council
WP-BIM	Web-based Parametric BIM

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

This thesis introduces a new framework for Web-based parametric modeling for architectural design collaboration and optimization through case studies. Within this framework, different users can collaborate from disparate places by sharing a Web-based model on Web browsers. The users get updates of a Web-based model in real time. The framework allows multiple users to work on the shared Web-based model in the process of drafting, design, simulation, and optimization.

The embodiment of the Web-based information technology (WebGL graphics, networking, and Web browsers) makes it possible for users to view, collaborate, and participate in parametric modeling through Web browsers. The other component is the design computing technology (visual programming) which enables users to interact with the parametric Web-based model. The framework consists of a WebGL-based model Viewer (Autodesk Forge Viewer), two visual programming tools, Grasshopper and Dynamo, and two software prototypes developed for this research. The prototypes, one for Grasshopper, the other for Dynamo, provide the communication between the Viewer and the visual programming tools. The Web-based model is modified through visual programming interfaces using the prototypes. The full-fledged visual programming environments enable building performance simulation and optimization while the Web-based model enables collaborative design exploration.

The Web-based model is a parametric model, so that different design options can be explored and generated by changing values of parameters. Besides, parametric relations allow users to run simulations and conduct optimizations. Visual programming provides users Graphical User Interface (GUI) instead of command line-based interface, and it extends the capabilities of the 3D modeling programs by allowing them to build custom parametric relationships. In addition, by integrating simulation plugins, visual programming enables users to simulate and optimize the architectural 3D models in its own environment eliminating the translation of the models to other software environments. With this framework, users can view, edit, simulate, and optimize a parametric Web-based model on Web browsers and visual programming user interfaces. The framework contributes to the interoperability by letting users to utilize different visual programming user interfaces during collaboration. A novel approach to design practice is promoted by emphasizing Web-based collaboration and employing building performance simulation and optimization in early steps of design process.

In order to test the framework, two case studies were conducted on a simplified residential building model inspired by Clavienrossier Architects' "Two in One House" project. The goal is to demonstrate the collaboration through the Web and the implementation of energy and daylight optimization. In the tests of case studies, two simulated users (actions done by the author) are included. For the convenience of discussion, in this thesis when mentioned the two simulated users, the word "simulated" are often omitted. In the Case Study 1, the collaboration of different visual programming tools and parametric capabilities of the Web-based model were tested. Two users tested the parametric capabilities of the shared Web-based model using Grasshopper and Dynamo. The basic geometric transformations including scale, translate, and rotate were tested in Tests 1.1, 1.2, and 1.3 respectively. The parametric relationships were built in each user's visual programming interface, Grasshopper or Dynamo. Both users have the control of exploring design options by changing the variables. The parametric changes were transferred to the Web browser with the help of two developed prototypes and viewed on the Web-based model simultaneously. Each user can get updates of the other user's design drafting results on the Web-based model and can retrieve data from the Web-based model.

In the Case Study 2, the collaboration of two Grasshopper users on the shared Web-based model in the process of optimization for different building performance objectives -in terms of daylight, energy use, and roof coverage- was tested. Case Study 2 was conducted through three tests of optimization. (i) In Test 2.1, the objectives were maximum preferred daylight and minimum roof shape with shading. (ii) In Test 2.2, the objectives were minimum energy and minimum roof shape with shading. (iii) In Test 2.3, minimum energy use was the first objective, maximum preferred daylight and minimum roof shape with shading was the second objective. The optimization dataflow was built on each user's visual programming interface, in this instance Grasshopper, and then the Web-based model was optimized through the program. Genetic algorithm was utilized in multi-objective optimization to generate a set of solutions called Pareto front. The Multi-objective evolutionary optimization plugin Octopus and building energy performance (BEP) simulation plugin Honeybee worked together in an iterative process of simulation and generation for the best solutions. After the users run daylight and energy simulation and optimization, chosen results were sent to the Web-based model.

Each user can view the results of other user’s optimization process on the updated Web-based model and can retrieve data from the model.

Overall, the findings of the Case Study 1 presented herein demonstrate that the proposed framework successfully provides multiple users’ collaboration on interoperable Web-based architectural models, which could be controlled parametrically through different visual programming tools, Grasshopper and Dynamo, on Web browsers on the transformations scale and translate. The findings of the second study presented the framework for collaboration between Grasshopper users through the Web-based model during daylight, energy, and integrated daylight and energy optimization while optimizing the roof shape with shading. In both collaboration case studies, the users can retrieve data from the Web-based model to update their visual programming files. While experimenting on the case studies, two interoperability issues were identified between the Web-based model and Grasshopper on how they responded to rotate and scale transformations. The first major issue is the reference point of transformations. All scale and rotation transformations reference the center of the bounding box of the Web-based model, whilst Grasshopper allows the users to define a reference point for these transformations. The twin model method was adopted to solve this issue. The twin model, which is the reproduction of the Web-based model in Rhino, served in calculating the displacement between the initial and the last positions. The second major interoperability issue is the difference of the directions of rotation between the Web-based model and Grasshopper. Even though the Web-based model’s and Grasshopper’s coordinate system follows the same right-hand rule during rotation, the positive Z-axis in

Grasshopper coincides with the positive Y-axis in the Viewer. This may cause inconsistency in the data transfer of the transformation matrix, which is compounded in Grasshopper, while assigning a rotation angle to the Web-based model. There are many technical challenges, especially interoperability problems, found and resolved for the project through the carefully designed experiments. However, due to the high complexity of the problem, limitations are found to be investigated in future work, for example, (1) geometry vertices of the Web-based model should be editable to enable more flexible parametric changes. (2) The automation of the data transfer from the Web-based model to the visual programming tools should be implemented. (3) The elimination of the need for the twin models in local visual programming environments, and enabling direct modeling, simulation, and optimization on the Web.

Limitations of the framework were found during the experiments, including: Limited control on scale transformation and editing of the vertices and the data transfer being one-way, from the visual programming tool to the Web-based model. First, the elements of the Web-based model could be scaled through visual programming tools, the scale only works uniformly. Secondly, the users cannot specify a vertex and control that point on the Web-based model, but control the whole element instead. Lastly, the data transfer requires follow-up and manual data input while updating visual programming tools according to the Web-based model. The framework provides a collaboration environment for users who have knowledge of Revit, Dynamo or Rhino, Grasshopper for design drafting process. Additional knowledge of building performance simulations and metrics, the knowledge for using the graphical user interfaces in visual programming tools to conduct simulations in simulation engines, and the knowledge to use the optimization tools are required to collaborate in the process of simulation optimization through this framework.

1.1. Research Problem

Since its invention in the 1960's, the Internet has gained importance day by day, and presently, it plays a major role in nearly every aspect of daily life. Internet technologies not only provide fast communication, but also enable data exchange in different formats. In addition, with the introduction of Cloud Computing technology, applications became accessible to users over the Internet and on demand (Buyya et al., 2009). A Web technology, WebGL allows users to interact with 2D and 3D graphics on Web browsers ("WebGL", n.d.). Although there are some viewer services that use Cloud and WebGL to visualize architectural models, design, drafting, simulation or optimization features are not a part of these models yet.

On the other hand, raising awareness in global warming, climate change, and the limitations of natural resources necessitate a sustainable approach in many fields, including architecture. BEP simulations are of great importance because buildings are the largest energy consumers, as 41% of U.S. primary energy is consumed by this sector (U.S. Department of Energy [DOE], 2011). Recently, more professionals in the building sector have specialized in BEP, and collaborated in design process with architects. The collaboration of specialists and architects is most valuable in the initial design stages since the significant design decisions which affect the building performance are made in that stage (Schade et al., 2011). With the development of visual programming tools and various plugins, now it is possible to integrate BEP simulation and optimization in the process of form finding. The users have the control of generating diverse design solutions through parameters. Yet, neither multi-user collaboration through the same visual programming tool, nor the collaboration of different visual programming tools are examined thoroughly.

In this research, a framework is proposed to provide a collaboration of different visual programming tools on Web browser. A prototype of the framework is developed and experimented to investigate how multiple users collaborate through Web-based model in search of possible solutions of design, simulation and optimization.

1.2. Research Objectives

The research presented herein seeks answers for the following two questions:

1. How to implement set up parametric relationships between the elements of a Web-based model and control it through different modeling and visual programming tools by different users while getting updates in real time?
2. How to implement two users to optimize the same parametric Web-based model through their own computers?

The main objective of this research is to test the Web-based parametric modeling by using the data flows in the visual programming tools, coding and WebGL (Web Graphics Library) technology.

Another important objective is to solve the interoperability issues by enabling users to practice with their own choice of tools on the shared Web-based model and to provide updates in real time. Lastly, this research aims to provide a solution for different users to execute different optimization processes on the same Web-based parametric model by sharing parameters. For all the above objectives, the research is also going to find limitations in the proposed framework.

1.3. Research Significance

The significance of the research lies in its demonstration of parametric Web-based modeling while dealing with interoperability issues. The research fills the gap of collaboration in parametric design, especially with different visual programming tools. The framework provides parametric Web-based modeling with two full-fledged visual programming tools and their third-party plugins. By providing the communication between the Web-based model and visual programming tools reside in local computers, the framework extends the capability of the Web technology beyond viewing by employing drafting, simulation, and optimization methods in the process of design.

2. LITERATURE REVIEW

The literature review contains five main sections: (1) Building Information Modeling; (2) Parametric Modeling; (3) Web-based Modeling; (4) Building Performance Simulation, and (5) Optimization. In the first section, the Building Information Modeling is investigated; because, Building Information Models can be converted into Web-based models and the Web-based model presented in this thesis is created by this conversion. In the second section, Parametric Modeling is examined, since one of the objectives of this thesis is to set up parametric relationships between the elements of the Web-based model which is lost during that conversion. In the third section, existing Web-based modeling tools are explored and analyzed in subsections; the capabilities and the limitations of these tools are identified. This section gives an overview of the recent Web-based modeling availabilities and clarifies the contribution of the thesis to the field. In the fourth section, Building Performance Simulation is interrogated in two subsections, Daylight and Energy. In both subsections, benefits, metrics, and tools to simulate building performance are examined. The related metrics and tools are used in the case studies. In the last section, optimization in architectural research is investigated; optimization tools for different software and purposes are analyzed. The appropriate tool is chosen for the case studies.

2.1. Building Information Modeling

Building Information Modeling (BIM) is distinguished as a new way of project drafting, modeling, constructing, and managing among other computer aided design tools by integrating information management to the project. The most significant feature of BIM is the generation, use, and extraction of information from design to demolition, through the building's life cycle. By digitally representing the physical and functional characteristics of a facility, BIM enables different stakeholders at different phases of the life cycle of a facility to collaborate by inserting, extracting, updating, or modifying information (National Institute of Building Sciences [NIBS], 2008; Eastman et al., 2011). The object-based nature of BIM's structure makes it possible to produce that information either by utilizing the generic and product specific properties of the element or by benefiting from the element's relations to others and the attributes the element has. For instance, modeling in 3D automatically creates plans, elevations and sections, or changing a location or size of a window simultaneously updates the form of the void on the wall. Since BIM emerges as an innovative way to design and manage projects (Azhar, 2011), its interoperability with other design and simulation tools gains importance.

BIM uses building objects and their relationships to express design intent; and the parametric modeling method establishes and manages relationships between building objects in a building model (Yan, 2014). The objects parametrically defined through relations and attributes facilitate automatic changes with any alteration in the objects' values (Eastman et al., 2011). According to Yan, the parameters defining objects and their relations can be built-in or user-specified such as mathematical formulations or computational algorithms (2017). External data can be fed into these relations by adopting Graphical User Interface (GUI) (Yan, 2017). Revit is a BIM authoring tool developed by Autodesk, Inc. It allows building parametric relations both inside its building elements or between these elements by utilizing reference lines, planes or dimensions. The definition of parametric relations can be further expanded with the integration of visual programming tool Dynamo.

Integrating Web technologies to create a Web-based model by using BIM would significantly contribute to design collaboration. BIM is object-based and its elements have the information embedded in these elements; therefore, Web-based models can display this information and allow users to retrieve data from Web. This information can be related to geometry, location, materials, etc.

2.2. Parametric Modeling

Parametric equation is a mathematical term, which employs independent variables called parameters in explicit functions while defining quantities (Stover & Weisstein, n.d.). By changing variables of parametric equations, it is possible to produce numerous solutions. Parametric equations are implemented in many fields, including architecture. The application of parametric equations in architecture creates the concept of parametric architecture. Although, parametric architecture has been used as a term for a long time (Davis, 2013), today, the term refers to digital parametric design. In 2008, Patrik Schumacher presented Parametricist Manifesto and defined the relationships between elements in a parametric composition as highly integrated so they cannot be easily decomposed into independent subsystems (Schumacher, 2008). This integrated dependency allows to easily explore form finding by proliferating design solutions.

Despite requiring more comprehensive understanding and skills in mathematics and computer science besides design, parametric design tools give more control and capability to designers (Woodbury, 2010). Hudson (2010) defines the implementation of parametric design in architectural practice in five tasks: Translation, Rationalization, Control, Generate and Test, and Share Information. According to him, the first task includes converting a design idea into a parametric model, which is called translation. The second task, rationalization is the application of known geometric principles and construction techniques in order to realize a project. The third task, Control, is considered by Hudson (2010) on multiple levels, in which the lowest level defines the fundamental geometry and the higher levels define further geometry by integrating fundamental geometry with other parameters. The fourth task, Generate and Task defines testing various criteria and generation of alternatives based on specialist input. The fifth task, sharing information, is the integration of design analysis by a specialist and production of construction information (Hudson, 2010). Implementation of parametric design produces a set of design solutions instead of a single output, especially in the exploration of form finding and visualizing. Similarly, parametric relations contribute to simulation and optimization processes with the help of computer algorithms in a relatively short time compared to an exhaustive search for the optimal solutions.

2.3. Web-based Modeling or Model Viewing Tools

Recently, incorporating Cloud services such as storing, viewing, editing, designing, and modeling to the architectural design process contributes to participatory collaboration and data exchange. Several Web-based tools have been introduced that support exchanging the parametric models, viewing the parametric model online and making parametric changes on a Web-based platform (Daher, 2017; Bakshi, 2017). A brief description of existing Web-based modeling tools, as well as an initial examination of each, is provided in subsections 2.3.1-2.3.5.

2.3.1. Lena

Lena by BIMobject® creates parametric models in Rhino and utilizes cloud service (BIMobject® Cloud) to view the model with parameters and other information (e.g. color or material) on the Web browser ("BIMscript®", n.d.). Lena focuses on creating BIM objects (such as products, e.g. toilet fixtures) with data and parameters embedded. Lena is built on top of Rhino and run by BIMscript which is a scripting language that represents parametric behaviors, calculations, product logic, level of detail, properties, and attributes ("BIMscript®", n.d.).

Lena presents a tool to set up parametric models and operates across Rhino, ArchiCAD, cloud, and BIM by creating objects in commonly used formats such as IFC, 3DS, DWG, and WebGL ("Is There a Place", 2017). However, it heavily relies on the users' knowledge and skills of scripting, and that becomes a barrier for designers in using Lena during drafting and modeling. Lena is created for the manufacturers to convert their products in their catalog into BIM objects. BIM objects contain manufacturer details and help to build up the data sheets which makes purchasing products easier. Lena helps to automatically provide information about the products while enabling parametrical changes, but it acts mainly as a product library supplier rather than a design tool.

2.3.2. Mobius Parametric Modeler

Developed by The Design Automation Lab at the Department of Architecture, National University of Singapore, Möbius Parametric Modeler is a tool for creating parametric models on Web browsers (“Möbius Parametric Modeler”, n.d.). Möbius Parametric Modeler is an application built on top of Möbius, which is an open-source, independent Web platform for building modeling apps that benefits from visual programming techniques, combining flowcharts with drag-and-drop, fill-in-the-blanks coding methods, creating the network of data flowcharts with nodes and wires. The difference of Möbius Parametric Modeler from the general visual programming tools is that in addition to combining the nodes with wires and composing the data flowcharts, the user also constructs the node itself (Janssen et al., 2016). The user generates nodes based on a node template.

Möbius Parametric Modeler interface consists of four main viewports that can be rearranged into any configuration by the user: output, flowchart, procedure, and parameters. Möbius Parametric Modeler adopts imperative programming which means customizing nodes by using a procedure viewport to define the set of codes. Specifying the nodes facilitates iterative building performance optimization process by including loops in both the procedure and the dataflow level (Janssen et al., 2016). However, the user needs to create the nodes from scratch in each time since the whole dataflow network is a single script till the memorization feature will be included as is specified in the future work (Janssen et al., 2016).

Möbius Parametric Modeler is a parametric modeling tool for a single user working on a Web browser which relies on coding and it becomes a tool for iterative loops and higher order functions (Janssen et al., 2016). The files cannot be converted to any other format. It is specified that additional viewers and function modules are being developed for BIM as a future work (Janssen et al., 2016). It is not a collaboration tool since an individual user scripts each file.

2.3.3. Spectacles

Developed by Thornton Tomasetti's CORE Studio, Spectacles is a plugin for Grasshopper to export Rhino/Grasshopper models (currently meshes and lines) and view them on the Web browser Spectacles is open source and utilizes three.js (Bhowes, 2015). Spectacles has a Grasshopper exporter and by using it, a Grasshopper file is exported as a .json file. Then this .json file can be uploaded to the Spectacles Web viewer (http://tt-acm.github.io/Spectacles.WebViewer/). In addition to Grasshopper, Spectacles also exports Revit files with a similar method. The Web viewer has orbit, pan, and zoom features. Clicking on an element displays the features and attributes of that element. The viewer can only control the settings of the scene such as background color, lighting, shadows, or the layers. It presents the embedded attributes of each element but gives no edit capability to users for manipulating them.

2.3.4. ShapeDiver

ShapeDiver is a plugin for Grasshopper to upload and display parametric models on the Web browser (“ShapeDiver”, 2018). ShapeDiver allows users to view the parametric models with their parameters on a Web browser and manipulate the parameters that were defined prior to the upload. ShapeDiver provides the possibility of embedding parametric models on a Website to present product configurators with sliders. It also enables exporting files in manufacture-ready formats such as .dxf, .3dm, .stl, .dwg, etc. The ShapeDiver Website provides API and UI controls for users to authorize their models. ShapeDiver provides material, color, and texture controllers for rendering the models (“ShapeDiver”, 2018).

ShapeDiver allows users to see different options whose boundaries are defined by the uploader. Once a parametric model is uploaded, it is on the Web and the range of its parameters’ values are fixed. There is no interaction between the original Grasshopper file and the uploaded parametric model. Instead, ShapeDiver is aimed at being a viewer to display parametric models, and provide users to change the values in those parameters.

All the aforementioned viewing and converting tools do not support working on the Web browsers on the same model by using different modeling tools and full-fledged parametric modeling tools directly through the Web. However, software interoperability and multidisciplinary collaboration is one of the key issues in the Architecture, Engineering and Construction (AEC) industry.

2.3.5. WP-BIM

WP-BIM is a framework that consists of a Web-based model and two visual programming tools. The Web-based model is generated by uploading a BIM model into a Web browser. The users are allowed to add parameters to building objects of a Web-based model. This Web-based model can be controlled with Grasshopper and Dynamo (Yan, 2017). The WP-BIM framework benefits from Autodesk Forge Viewer (the Viewer) for displaying BIM models with parameters on the Web browser. Besides, WP-BIM provides communication between the visual programming tools and the uploaded BIM model on the browser, which allows users to control the parameters from their computers with their own choice of visual programming tools. WP-BIM framework previously used Flux (https://flux.io) to provide the communication between a Web-based model and the visual programming tools, Dynamo and Grasshopper. However, Flux is no more available and another plugin developed by Yalong Pi, Bihan Wang, and Dr. Wei Yan could provide this communication in one-way from the visual programming tools to the Web-based model. The framework was experimented in parametric modeling by two simulated users controlling the same BIM element (roof) in different Web browsers (Google Chrome and Microsoft Edge) through different visual programming tools (Dynamo and Grasshopper). One of the users controls the translation and scale of the roof, while the other controls the rotation. One Grasshopper user can optimize the model automatically by using the Galapagos node which is a Genetic Algorithm tool. Objectives are covering the building and front area of the door while having a minimal area. After running the optimization, a set of generations of design options are created. The options can be viewed on the Web and synchronized with the Dynamo user in real time.

2.3.6. Discussions About the Web-based Modeling or Model Viewing Tools

Through the examination of existing Web-based design and viewing tools, it is observed that some of them act as a viewer with limited control on the Web-based models. All of them have the viewer controls such as pan, orbit and zoom; however, they have no control or limited control on manipulating the geometry and data. The viewer shows the geometry and data but provide no access to manipulate them as seen in Spectacles; or the viewer gives limited access to manipulate geometry and data as seen in ShapeDiver. This freedom to control the sliders on the Web browser is defined in the local model control file and once it is uploaded, the range of the sliders' values gets fixed since there is no communication between the Web-based model and the local model control file. Möbius and Lena rely on command-based programming, which requires a significant amount of time to learn them for designers.

Another limitation with Möbius and Lena is that their own programming limitations compared to visual programming tools such as Grasshopper, which has a comprehensive set of functions either built-in or through plugins. None of these tools present a solution for interoperability issues except WP-BIM framework. In addition, WP-BIM is also a prototype with limited capability to make the Web-based model a full parametric model. Finally, none of the tools except WP-BIM provide potentials for BEP simulation or optimization. The properties and capabilities of existing Web-based tools are summarized in Table 1.

Table 1. Comparison of existing Web-based modeling tools or model viewing tools

Tool Name	**Parametric Modeling**	**Inter-operability**	**Collaboration**	**Simulation**	**Optimization**
Lena	+	+	_	_	_
Mobius Parametric Modeler	+	_	_	_	_
Spectacles	_	+	_	_	_
Shapediver	limited	+	_	_	_
WP-BIM	limited	+	+	+	limited

2.4. Building Performance Simulation

Building Performance Simulation section consists of two subsections: Daylight Simulation and Energy Simulation. In each subsection, the benefits of the simulation, metrics, and tools are examined. The reasons for the chosen metrics and the tools are discussed.

2.4.1. Daylight Simulation

Proper daylighting in buildings has significant effects both on health of occupants and building energy performance. The studies show that the appropriate levels of daylight increase office workers' and students' performance, productivity, and concentration (Heschong Mahone Group, 2003a, 2003b). Daylit rooms make occupants feel more comfortable. The daylight, especially the morning sun, speeds up recovery time and reduce patients' average length of stay (ALOS) in hospitals (Choi et al., 2012). By stimulating circadian rhythm, daylight has direct positive impact on well-being (Edwards & Torcellini, 2002). However, daylight should be carefully designed to meet the occupants' needs. For instance, excess amount of light would cause glare which would negatively affect the performance and productivity of office workers or students. In addition, avoiding excess daylight with blinds would contribute to electrical lighting.

On the other hand, if daylighting becomes insufficient; additional electrical lighting is needed to meet the minimum illuminance levels, which increases the amount of energy consumption. However, excess amount of daylight would bring solar gain, which would increase the demand for cooling and add to electrical loads, as well. Therefore, daylighting should be planned rigorously in order to meet the requirements of the function of the building. For instance, healthcare facilities require more daylight than other buildings (USGBC, n.d.).

There are a number of useful daylight metrics and standards that guide architects to plan daylighting through daylight simulations. Illuminance is the total luminous flux incident on a surface, per unit area with unit lux (SI) or foot-candle (IP). Illuminating

Engineering Society (IES) provides illuminance levels according to building and space types. As reported by IES, recommended illuminance values for residence ranged between 50-500 lux (DiLaura et al., 2011). However, in houses many activities take place and this range does not fully provide the required illuminance levels for some activities such as reading or typing. Also, good lighting is subjective and without requisites, it is difficult to identify an illuminance range for houses. This brings a necessity to specify the activity and the space of the house, or to follow a universally accepted standard.

For this thesis, the house is accepted as a universally designed house, which requires minimum levels of illumination for older people and people with visual impairment. Universal design is a term that implies designing for occupants of all ages, sizes, and abilities, or shortly "design for all people", throughout the lifespan (NAHB National Research Center, 1996). Universal design principles are developed to answer the needs of a growing population of people with disabilities by making products, buildings, and exterior spaces accessible and adaptable (Mace et al., 1996). The buildings that receive effective daylighting provide good visibility. In order to make the buildings to provide visibility to all people, the daylighting should be designed to meet the requirements. If universal features are considered at the early stages of design, the cost of the final product would not exceed the cost of a traditional one (Mace et al., 1996). According to CIE 227:2017 Lighting for Older People and People with Visual Impairment in Buildings, a technical report by International Commission on Illumination

(CIE), aged and visually impaired people require illumination levels of at least 750 lux for task areas to do reading, writing, typing, etc. (Akashi et al., 2017).

Although minimum illuminance requirement for older people and people with visual impairment is identified in CIE 227:2017 standard, there is no upper limit provided in the report. Yet, excess daylight may result in glare and discomfort, which could obstruct the visibility. Therefore, another metric is required to set the upper limit for a good visibility for older people and people with visual impairment. To determine the optimum daylight range for this research, a synthesis was obtained by incorporating Useful Daylight Illumination (UDI) metric to universal design principles. UDI is chosen because it is a human-based metric, and it is constituted based on the surveys of occupant comfort. According to UDI, excess levels of illuminance create discomfort for occupants (Nabil & Mardaljevic, 2006). So that, the lower level is set according to the needs, and the upper limit is set for illuminance comfort levels. As a daylight availability metric, UDI defines useful levels of daylight illuminance based on realistic sky and sun conditions by aiding climate-based analyses of yearly daylight illuminance levels during occupancy hours (Nabil & Mardaljevic, 2006). According to UDI, the illumination levels below 500 lux are considered useful, which are ignored by Daylight Factor metric; a threshold is defined for excess illuminance levels that would result in occupant's discomfort (Nabil & Mardaljevic, 2006). The Daylight Factor metric was out of the scope because of its limits with working independent from the orientation and only functioning under the overcast sky condition (Nabil & Mardaljevic, 2005). According to Nabil and Mardaljevic (2006), the useful limits of UDI are determined between 100-

2000lux. However, when the illumination levels are below 500 lux, additional electrical lighting may be needed.

In order to provide a comfortable illuminance for aged and impaired vision occupants, the simulation is run with minimum and maximum constraints. The minimum constraint is determined by using the illuminance levels of CIE 227:2017 standard, and the maximum constraint comes from UDI levels. As a result, the new useful daylight illuminance for the universal design house is ranged between 750lux and 2000lux for this study.

RADIANCE, ADELINE, DOE, EnergyPlus, and DAYSIM are some of the prevalently used building performance simulation tools (Wong, 2017). Among them, RADIANCE, DOE, and EnergyPlus are command line-based engines and it is difficult to use them without programming knowledge. With the development of graphical user interfaces (GUI), they are available to architects. With the help of GUIs such as Honeybee and Diva, these engines can be accessed through either GUI or visual programming tool. RADIANCE is a command line-based lighting simulation engine, which helps predict illumination, visual quality and appearance of spaces, and evaluate new lighting and daylighting technologies by employing the light-backwards ray-tracing method (Ward, 1994; Fritz & McNeil, 2019). Honeybee is a GUI plugin to run daylight simulation in RADIANCE through Grasshopper (“Honeybee”, n.d.). Honeybee works together with another plugin, Ladybug, to import EnergyPlus Weather files (.EPW) into Grasshopper and visualize the results of daylight simulation (Roudsari & Pak, 2013).

2.4.2. Energy Simulation

Due to the rise of oil prices and energy crisis in the 1970s, energy efficiency became a major concern in many sectors including construction. Also, the limitations of the natural resources, global warming, and climate change place stress on energy savings and renewable energy sources. On the other hand, buildings are the largest consumers of energy as compared to other sectors, such as transportation or industry (U.S. Department of Energy [DOE], 2011). Therefore, energy efficiency became a significant concern when designing buildings, therefore implementing BEP simulations gained significant importance.

In order to make buildings more energy efficient, a major thing to do is to predict how much energy will be used by a building during its life-cycle. Energy performance simulation tools not only calculate BEP and HVAC requirements, but also output valuable data about thermal comfort and how to design properly according to needs of occupants. For the past fifty years, a large number of BES tools have been developed, and Crawley at al. (2005) reviewed twenty major BEP simulation programs in detail with up-to-date comparison of the features and capabilities. Maile et al. (2007) reviewed BEP simulation engines EnergyPlus and DOE2 and their user interfaces in the perspective of a life-cycle and interoperability. These two engines are operated by mathematical and thermodynamic algorithms, and input and analysis output data are command line based. Honeybee and Archsim (Reinhart et al., n.d.) are available GUIs for EnergyPlus to work in Grasshopper. EnergyPlus is an open-source whole-building energy modeling (BEM) engine developed by DOE, to model both energy consumption—for heating, cooling, ventilation, lighting and plug and process loads—and water use in buildings (“EnergyPlus”, n.d.). GUIs present model-based control, therefore geometry can be input in 3D, and the output can be visualized in 3D as well. Running simulation through GUI brings the whole process and the output to the same environment where geometry is generated. The results could be directly used in the design process or could be optimized within the same visual programming environment. Since the simulation engines only accept single planes for BEP modeling, the geometry is required to be built with single planes instead of solid modeling or surface modeling. For instance, walls are presented as single planes instead of having thickness or having two layers of planes.

Honeybee provides access to a ready-to-use library of building construction materials through EnergyPlus. The building construction materials in the library are classified according to the standards from ASHRAE and CBECS. By running an energy performance simulation, zone loads, HVAC systems and equipment, electrical systems, renewable energy systems, ventilation and multizone airflow, and infiltration data can be obtained. Energy simulation is run in two phases, base design, and an improved design. The first one predicts the energy requirement of a building, while the second provides how much the energy performance of the building is improved and the amount of energy saved in the total energy use after doing modifications.

Energy Use Intensity (EUI) is a metric to calculate energy efficiency in a building. EUI is obtained by dividing the total energy usage of a building in one-year by its total floor area. Units are kBTU/ft^2 yr (IP) or kWh/m^{2*}yr (SI). The EUI benchmarks are determined according to the functions of buildings. U.S. Energy Use Intensity Table (U.S. Energy Use Intensity, 2018) and U.S. National Average Site EUI Table by AIA 2030 Commitment Working Group (“Reference”, n.d.) present EUI targets to evaluate the performance of buildings.

2.5. Optimization

Optimization is a mathematical term later adopted by other fields, such as computer science, manufacturing industry and architecture, in order to represent the process of finding the optimal solution. Optimization is either calculating the maximum or the minimum value of an objective function within the given range of parameter values.

The development of the computational design tools provides opportunities to do architectural design optimization (ADO) in finding optimal design solutions in many ways, e.g. floorplan layout design (Michalek et al., 2002) or optimal circulation paths between rooms. Gero (1975) pointed out that, building services such as air-conditioning, vertical transportation or materials selection can be treated as a multi-factor optimization problem. Today, from structural design to envelope design, energy optimization to daylight optimization, optimization became a method of designing. According to a survey conducted between the practicing architects, there is a need of an interactive multi-objective optimization tool especially in the process of daylight, structure, and geometry optimization (Cichocka et al., 2017). Visual programming tools allow for building parametric relations and optimization while generating the solutions within the same tool, with geometry or numerical data. Two of the prevalent visual programming tools, Grasshopper and Dynamo, allow for optimization with the help of additional plugins (Rutten, 2010; Vierlinger & Bollinger, 2014; Rahmani et al., 2014, 2015a & 2015b). Optimo is a BIM-based multi-objective optimization tool that utilizes Dynamo for high performance building design and generates a set of solutions based on evolutionary principles (Rahmani et al., 2015a). In Grasshopper, there are many available optimization tools. Among those tools, Galapagos and Octopus are two of the major evolutionary optimization tools. Galapagos Evolutionary Solver is a single objective optimization plugin for Grasshopper that emulates theory of evolution by adopting Genetic Algorithm and provides a single solution (the best solution) for a fitness function (Rutten, 2010). The variables, like genes, are fed into Galapagos with an objective which is the fitness function. The user starts and ends the solver which creates generations of variable values that fit the fitness function. Those generations are crossed over in an iterative process to yield the best result while worse solutions are deleted. Galapagos helps achieve the best result based on a fitness function in the process of design or simulation. By combining multiple objectives into a single objective using weighted sum, multiple objectives can be optimized through Galapagos.

Octopus is a multi-objective optimization tool in Grasshopper that applies evolutionary principles to produce a range of best solutions with tradeoffs between the objectives (Vierlinger & Bollinger, 2014). According to Fonseca and Fleming (1995), in a multi-objective optimization, the solution is not a single one, but instead, a set of solutions known as Pareto-optimal set, that satisfy the multi-objectives. Each solution in a Pareto-optimal set is in an optimal state where no improvement in a solution can be achieved without degradation in at least one of the remaining solutions (Fonseca & Fleming, 1995). Octopus generates this set of solutions, also known as Pareto front, which are symbolically presented as cubes in a graphical solution space. The population size of each generation of solutions and the number of generations denote the iterative process of producing design solutions; the starting, pausing, and stopping the optimization process are controlled by the user (Vierlinger & Bollinger, 2014). The history of the whole process can be recorded, and the data can be exported as text. Each objective is defined as a fitness function, and as default, Octopus algorithm runs to achieve the minimum values for all fitness functions. Octopus iteratively generates design solutions according to these fitness functions within the boundaries of parametric inputs. These inputs are connected to Octopus and act similarto genes in evolution. Integration of optimization, which adopts Genetic Algorithms (GA), into parametric modeling generates and evaluates results in a looping process (Caldas & Norford, 2002). If Octopus is used jointly with a simulation tool, simulation is run for every design solution at each iteration for each design solution.

3. METHODOLOGY AND PROTOTYPING

In this section, the methods and the technology adopted for building and testing the framework and the prototypes are described in detail. The features of the Web-based modeling and the working principles of two prototypes, Grasshopper and Dynamo, are provided. Also, the information about the case study model is included.

The research methodology includes a comprehensive literature review on Web-based modeling tools and testing the framework of this thesis for selected case studies. The framework integrates a Web-based viewer with full-fledged visual programming tools, Dynamo and Grasshopper, which support complex parametric modeling features through their comprehensive built-in functions and further extended capabilities with algorithm libraries and many third-party plugins. The framework embodies all these functions and becomes a powerful tool.

To provide the dataflow between the plugins and the server, Socket.io is adopted. Socket.io is a JavaScript library enabling real-time, bi-directional communication between Web clients and servers (https://en.wikipedia.org/wiki/Socket.IO). To test the framework and visualize the Web-based model, a local server was created by utilizing socket.io. Any change in the values of the transformations (scale, translate, rotate) of individual building element is required to be followed by the parametric changes of the other related elements. These changes in values are transferred as a matrix from plugin through the server. Consequently, these changes are visualized simultaneously on Web-based models on different browsers. The plugins which provide data transfer between the server and the visual programming tools are both built upon templates. Dynamo plugin is created by employing Zero-Touch Node, which is a custom node written in C# and helps to benefit of the .NET framework, a solid IDE, debugging tools, and many libraries (Cominetti, 2017). Grasshopper plugin is created by employing Add-on template.

The framework employs Autodesk Forge Viewer (the Viewer) which utilizes JavaScript-based WebGL technology for viewing the models on Web. WebGL technology makes it possible to display, orbit, pan, zoom, etc., on the Web-based model by default. Autodesk Forge Viewer API and Model Derivative API are used during this translation process, and the BIM model generated in Revit is converted into .svf format to be displayed in the Viewer. In order to create a Web-based model, a BIM (Revit) model, is converted into a Web-based model by using Autodesk Forge Viewer (the Viewer) (model.autodesk.io).

Viewer API/Extension is utilized to generate custom nodes for both Grasshopper and Dynamo to communicate with and control the Web-based model through parameters for geometry transformations (including Translation, Rotation, and Scale). The prototype enables the communication between Grasshopper/Dynamo and the Viewer. By using a slider, an object index is specified. Each BIM element has its own index number. In order to control BIM elements, each element must have its own dataflow chart, and to obtain parametric changes, the parametric relationships should be built up between these elements. The parametric relationships in the BIM model are generally lost after it had been converted to a Web-based model. A main task of this research is to build up these parametric relationships in Grasshopper and Dynamo in order to control the Web-based building model parametrically.

Transformation in Grasshopper is defined by a transformation matrix, which defines rotation, scale, translation, shear, etc. These transformation operations in Grasshopper are driving the rotation, scale, and translation of objects in the Viewer. Any changes done by any user is reflected on the Web-based model and each user gets the updates in real time in each browser. The Web-based model has its own instruments to display data. Since the model is converted from a BIM (Revit) model, it is object-based and all the data of each element is embedded in the model. The Forge Viewer also allows 3D models in other formats such as 3dm or dwg to be converted into Web-based models and viewed on the Web browsers. However, these models are not BIM models; therefore, they only represent 3D model geometry, but not any information about the elements of the Web-based model, which would obstruct the further collaboration in the proposed framework. Still, users can retrieve data by using the measurement tool within the Viewer. Figure 1 displays sample Rhino and AutoCAD models viewed on the Forge Viewer. It can be noted that clicking on an element in the Viewer do not provide any data about that element. Therefore, BIM (Revit) was used to create Web-based model in this framework.

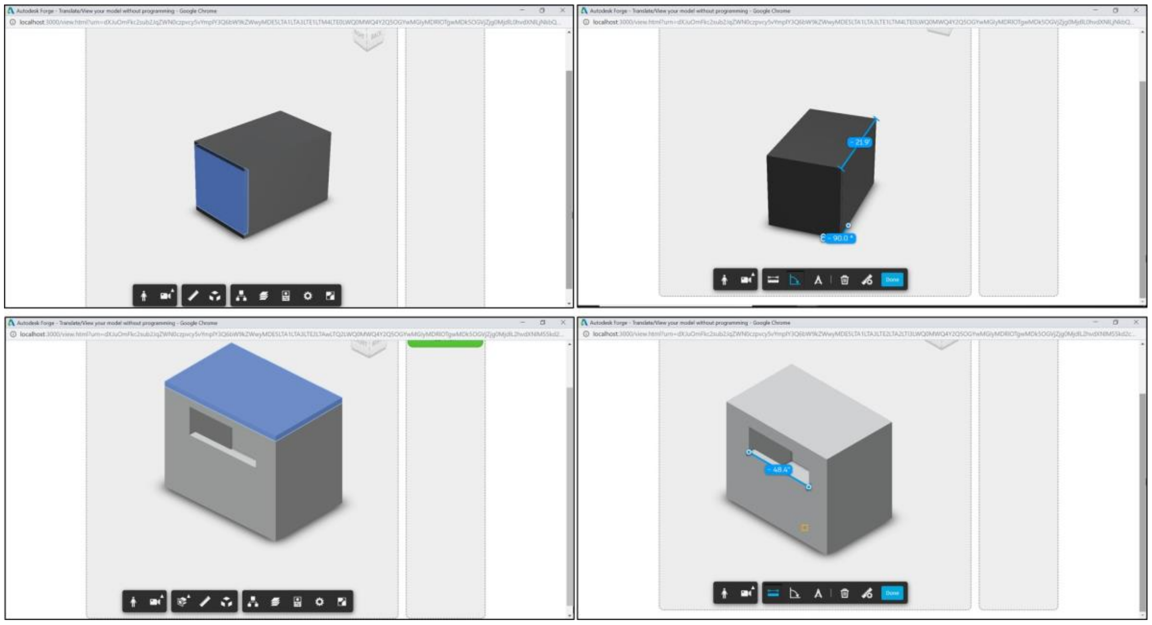


Figure 1. The Web-based model created with a Rhino model 3dm (top), the Web-based model created with an AutoCAD model dwg (bottom)

By clicking on an element and using the tools provided, data can be retrieved from the Web-based model (Figure 2 - 3). The Web-based model allows users to interact with the model and presents two additional features, *Explode* and *Section Box*, to visualize the relationships between the elements (Figure 4 -5).

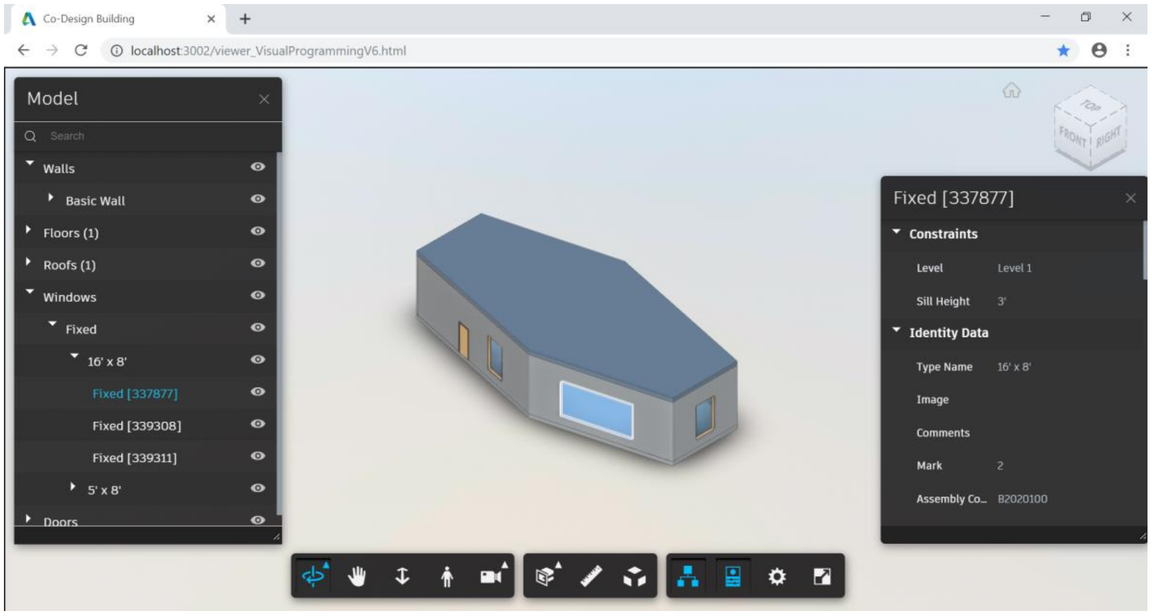


Figure 2. The Web-based model displaying properties of a window

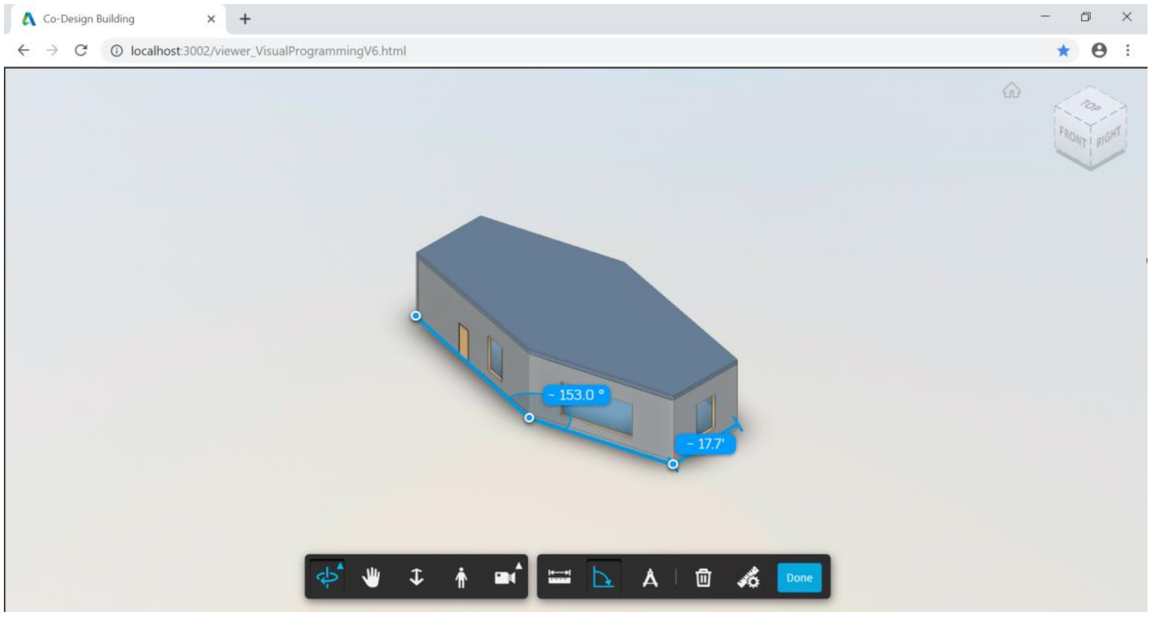


Figure 3. Dimension and angle retrieving from the Web-based model

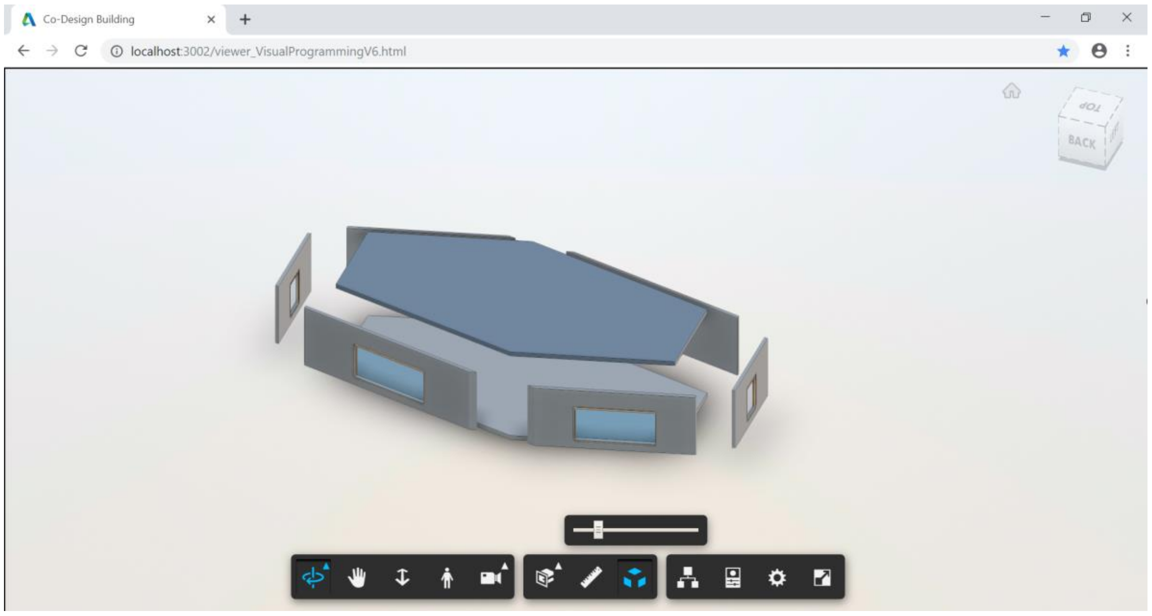


Figure 4. The Web-based model Explode feature

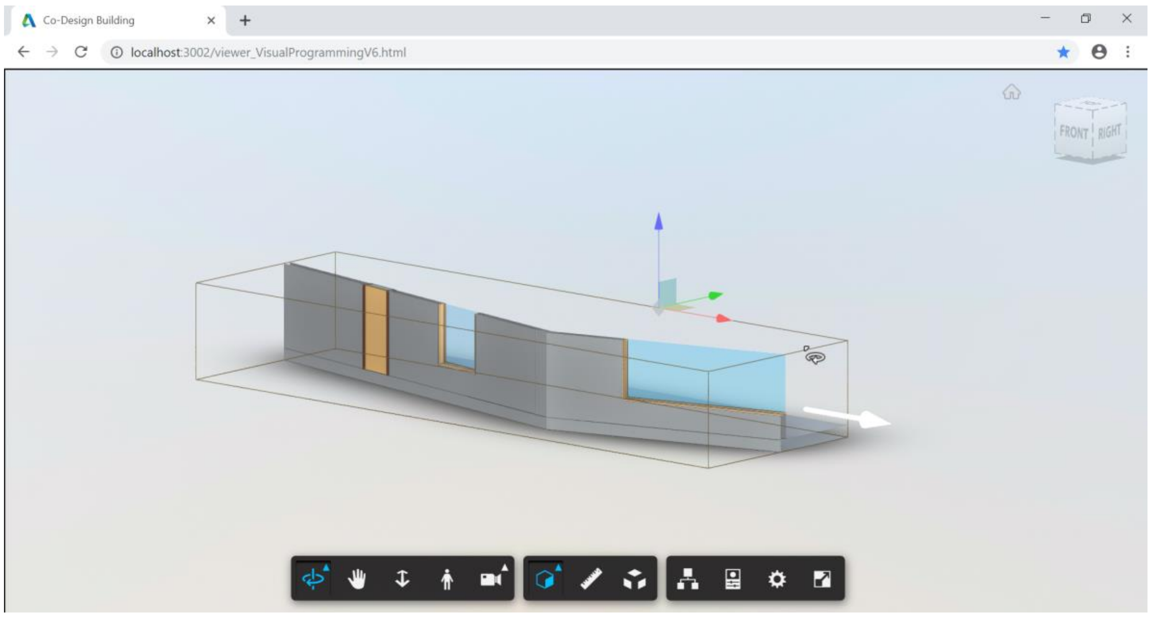


Figure 5. The Web-based model Section Box feature

As a case study, a simple house model is created inspired by Clavienrossier Architects' housing project "Two in One House" (Frei, 2013). This project is chosen because of its form; as its diamond-shaped geometry may offer a variety of diverse geometries in the optimization process. However, in order to reduce the simulation time, the building is simplified as a single zone with a flat roof.

The model of the house is built up in Revit and uploaded through the Viewer (model.autodesk.io) to the Web. In the upload process client id and client secret are used to get an access token from Viewer. Both id and secret are generated in the Viewer's Website following the generation of application (https://forge.autodesk.com/). Several models can be uploaded by using the same access token. An individual URN for the model is generated after each upload. By using socketserver.js, the model can be tested by running the Viewer with the parametric Web-based model from the Web server (for the testing purpose, a local Web server was used: http://localhost). The model is controlled from local Grasshopper and Dynamo programs. The users get updates in the Web-based model after each change. With any change in the location, scale or rotation of one element, other elements could parametrically adapt themselves and the model elements should act as a whole.

3.1. Prototype for Grasshopper

As previously mentioned, a Viewer plugin is developed for this research to provide data transfer between Grasshopper file and Web-based model. It is displayed on Grasshopper canvas as a node which has four inputs: Object Index, Transformation

Matrix, Boolean Toggle for Emit (the change), and Timer (Figure 6). The Object Index serves to individually select each element of the Web-based model for change, e.g. Object Index 4 refers to roof, Object Index 8 refers to floor, and etc. The Transformation Matrix is a combination of transformations: Scale, translate, rotate, etc. Emit serves to transfer the data of the Transformation Matrix to the Web-based model. A timer set to 1 second is connected to the Viewer node, which determines the intervals of data transfer to the Web-based model.

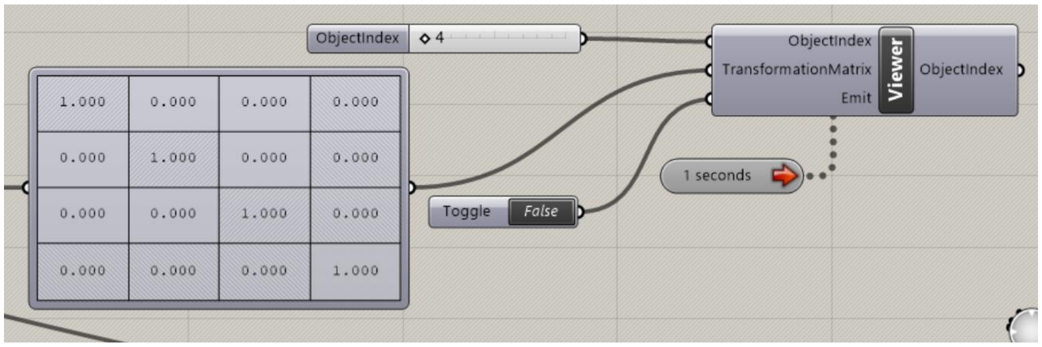


Figure 6. Grasshopper Viewer plugin, connected to the Object Index, Transformation Matrix, Boolean Toggle for Emit, and Timer

Each element of the Web-based model has its own unique Object Index number. A Viewer node for each wall, roof, and floor is required to be created. The same Viewer node can be used for both the windows (both frames and glass) and their host walls. The Object Index number for each element is identified by observing them through the Web-based model. Each Viewer node has its own Transformation Matrix; however, all of the Viewer nodes are controlled by a single Boolean Toggle for triggering data transfer to the Web-based model.

A twin model of the Web-based model is built in Grasshopper in order to guide the user in building parametric relationships and conducting simulations and optimizations. The original Revit file, which was previously uploaded to Web for creating the Web-based model, is the basis for the twin model. The file is converted to a .DWG file during exportation, so it could be directly opened in Rhino. The twin geometry of each element of the Web-based model is reconstructed by using Grasshopper. All elements are created individually as Breps on the Grasshopper canvas by referring to Rhino poly surfaces. The scale and translation parameters which are created for the Web-based model, are also utilized by the twin model. Even though both Grasshopper and the Web Viewer follow the same right-hand rule for coordinate systems, there is a rotation between the two reference frames, such that the positive Z-axis in Grasshopper coincides with the positive Y-axis in the Viewer. This may cause inconsistency in the data transfer of the transformation matrix, which is compounded in Grasshopper, while assigning a rotation angle to the Web-based model. In order to align the rotation in both coordinate systems, rotation angle is flipped (Figure 7).

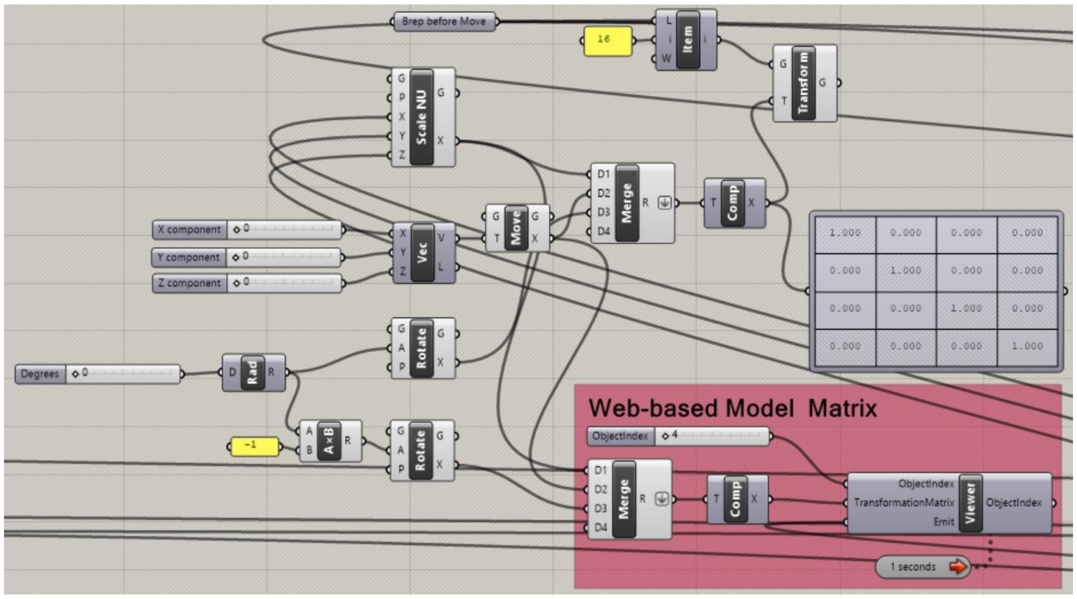


Figure 7. Typical data flow of an element in Grasshopper showing the Web-based model Transformation Matrix and the twin model Transformation Matrix

The goal is to create a twin of the Web-based model which acts and responds the same. Using a twin model allows users to mirror the changes in Web-based model elements and to retrieve data from the twin geometry. For instance, if a wall is rotated, twin wall provides the final location data. This data can be used to calculate the scale factor of other walls in order to meet that wall. The twin model is utilized for building and visualizing parametric relationships.

In order to make the twin model to respond the same with the Web-based model, the reference plane of scale should refer the same location. The geometric center of the Bounding Box constitutes the center of the Web-based model. All the transformations are operated with reference to this center. Therefore, the Grasshopper twin model is needed to be translated, since the origin point of Grasshopper is located on the northwest corner of the model.

Two types of parametric relationships are built within Grasshopper. First, each element is converted to become parametric with translate, scale, and rotate variables. Second, parametric relationships are built between elements to create a parametric architectural model. Different configurations of data flow are created for different types of parametric relationships to reduce the complexity. For instance, in order to show scale, a file is generated; another one is generated to show translate.

3.2. Prototype for Dynamo

Similar to the Grasshopper Viewer plugin, the Viewer plugin for Dynamo is developed for this research to provide data transfer between the Web-based model and the Dynamo file. Dynamo is a visual programming tool which can be integrated as a plugin to Revit or operated as a standalone version. In Dynamo Studio, the standalone version, a Viewer node is created for each of the elements, since Object Index Input only accepts one value at a time. Object Indices are matched with Web-based model elements by checking the model. The Dynamo Viewer node, similar to the Grasshopper Viewer node, has inputs for Object Index, Scale X-Y-Z, Translate X-Y-Z, Rotate X-Y-Z and Emit. Due to the lack of a built-in transformation matrix node in Dynamo, this prototype separately receives transformation values for each input of Scale, Translate, and Rotate as X-Y-Z values. Boolean Toggle fed into Emit serves to control the transfer of data from Dynamo to the Web-based model (Figure 8). In Dynamo, a twin model is not built, since the Web-based model is originally a Revit model.

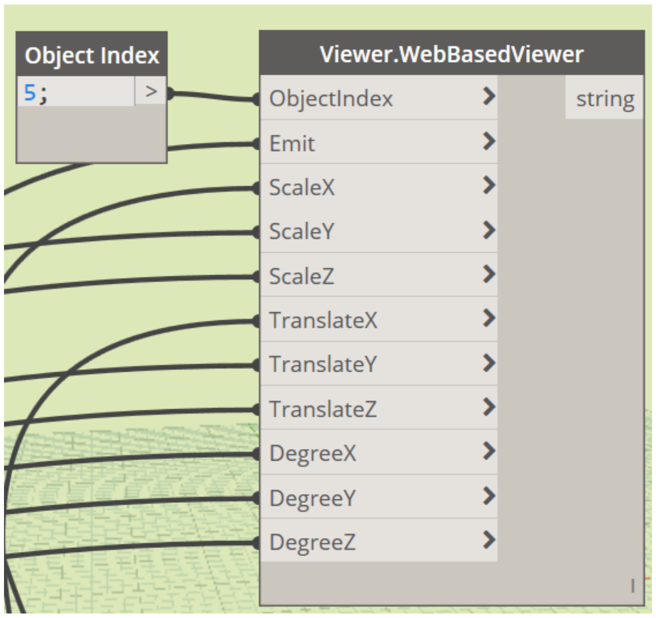


Figure 8. Dynamo Viewer plugin

4. EXPERIMENTS OF THE FRAMEWORK

In this section, two case studies, each consisting of three tests, are described in detail. In Case Study 1, the ways of collaboration between the Grasshopper and the Dynamo users through the model and creating parametric relationships between the elements of the Web-based model were explored. The parametric relationships were experimented on the basic transformations, scale, translate, and rotate in the subsections: Test 1.1, Test 1.2, and Test 1.3, respectively.

In Case Study 2, the ways of collaboration between two Grasshopper users on multi-objective optimization (the building performance and the roof shape optimization) through the Web-based model were explored in three different tests. In Test 2.1, the Web-based model was optimized for maximum preferred daylight and minimum roof shape with shading objectives. In Test 2.2, the Web-based model was optimized for minimum energy use and minimum roof shape with shading objectives. In Test 2.3, the Web-based model was optimized for two objectives: The first objective was the weighted sum of maximum preferred daylight and minimum roof shape with shading and the second objective was minimum energy use. Table 2 sums up the experiments of the framework over the tests of the case studies and the tools used.

Table 2. Experiments of the framework

Definition				# of users	Tools used
Case Study 1	Parametric changes	Test 1.1	Scale	2	Grasshopper & Dynamo
		Test 1.2	Translate	2	Grasshopper & Dynamo
		Test 1.3	Rotate	1	Grasshopper
Case Study 2	Building performance and roof shape optimization	Test 2.1	Max. Daylight & Min. Roof with shading	2	Grasshopper Honeybee Octopus
		Test 2.2	Min. Energy use & Min. Roof with shading	2	Grasshopper Honeybee Octopus
		Test 2.3	Max. Daylight and Min. Roof with shading & Min. Energy use	2	Grasshopper Honeybee Octopus

4.1. Case Study 1

In this case study, parametric changes are tested through the transformation factors, scale, translate, and rotate. These transformations are already defined in WebGL rendering technology. Therefore, the Web-based model can perform scale, translate, and rotate transformations, and they can be controlled by Grasshopper and Dynamo with the help of the Viewer plugins. In order to perform the parametric changes, the relationships are built by the user. For each transformation, particular data flow needs to be created in both Grasshopper and Dynamo. The parameters control the transformations, and the ranges of the parameters are determined by the users. The goal is exploring the ways of implementing the parametric relationships, testing the parametric capabilities of the framework for individual transformations, and observing if this serves parametric form propagation to be used in design drafting and editing.

Separate tests were run for each transformation: Test 1.1 scale, Test 1.2 translate, and Test 1.3 rotate. Two simulated users, using Grasshopper and Dynamo, were simulated for Test 1.1 and 1.2. A single user, using Grasshopper, was simulated for Test 1.3. Users could be architects, consultants, engineers, or clients. The framework consists of two users with their local control visual programming files, and a shared Web-based model. The Web-based model is visualized on the Web browsers. In order to simulate the Web page, local servers were utilized with the help of Node.js, and Socket.IO, and the page was visualized on Viewer provided by models.autodesk.io. The variables for transformation parameters are provided in Table 3. In Case Study 2, building performance optimization was conducted, and the orientation of the model was required to be provided for performance simulations. Therefore, the model was oriented towards the same direction as the inspiring house, aligning northwest and southeast direction. Since the building form was a distorted hexagon, and the alignment of the model was not in the cardinal directions, names are assigned to the walls in order to eliminate confusion. Figure 9 demonstrates the assigned names to the model elements.

Table 3. Parameter definition

Parameter	Value Range	Type of Number
Scale X Scale Y Scale Z	1.000 – 3.000	Decimal Numbers
Translate X Translate Y Translate Z	0ft – 30ft	Integers
Degree X Degree Y Degree Z	0° - 360°	Integers

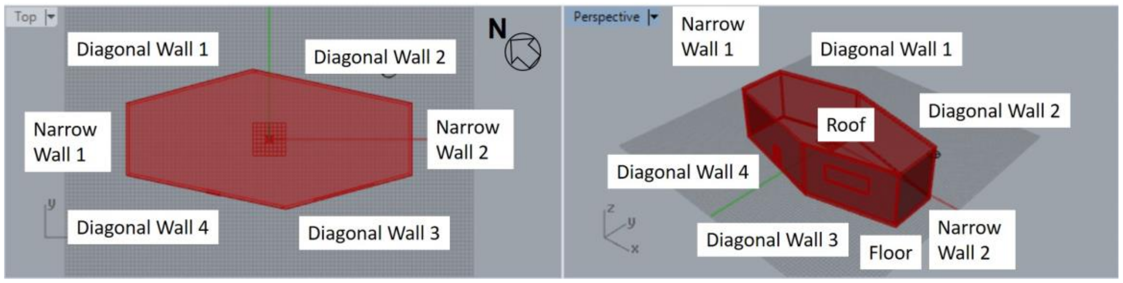


Figure 9. Model elements' names

4.1.1. Test 1.1 and Results

The goal in Test 1.1 is implementing collaboration and interoperability between Grasshopper and Dynamo through the Web-based model in the transformations: Scale in X, Y, and Z directions. Scale transformation was tested by simulating two users: Grasshopper and Dynamo users. According to the scenario, both users set up their visual programming files in their computers, and they shared the Web-based model in their Web browsers. The Grasshopper user defined transformation parameters for each of the model element, and built the parametric relations between these elements in Grasshopper. The parametric relations serve all the elements to be scaled in X, Y, and Z directions simultaneously. Although each object had its own matrix, all of the scale transformation values in the objects' matrices were controlled by single Scale X, Scale Y, and Scale Z parameters. Similarly, Emit input was controlled by a single Boolean toggle for controlling data transfer to the Web-based model (Figure 10).

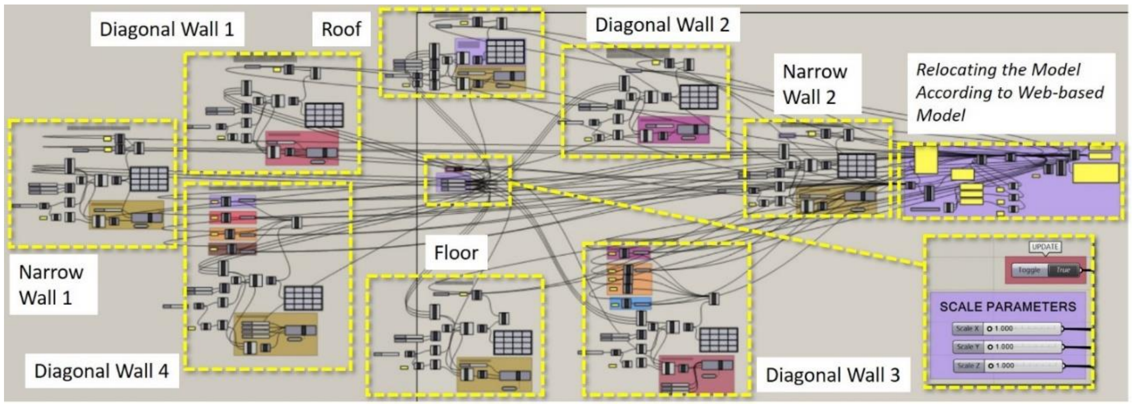


Figure 10. Parametric dataflow in Grasshopper for Scale X-Y-Z

On the other hand, the Dynamo user built the same parametric relations in Dynamo canvas by using Scale X, Scale Y, and Scale Z parameters. Similar to Grasshopper, a single Scale X parameter controlled all the Scale X transformations of objects. Likewise, all Scale Y or Scale Z transformations were controlled by single parameters. The data transfer was provided by a single Boolean toggle (Figure 11).

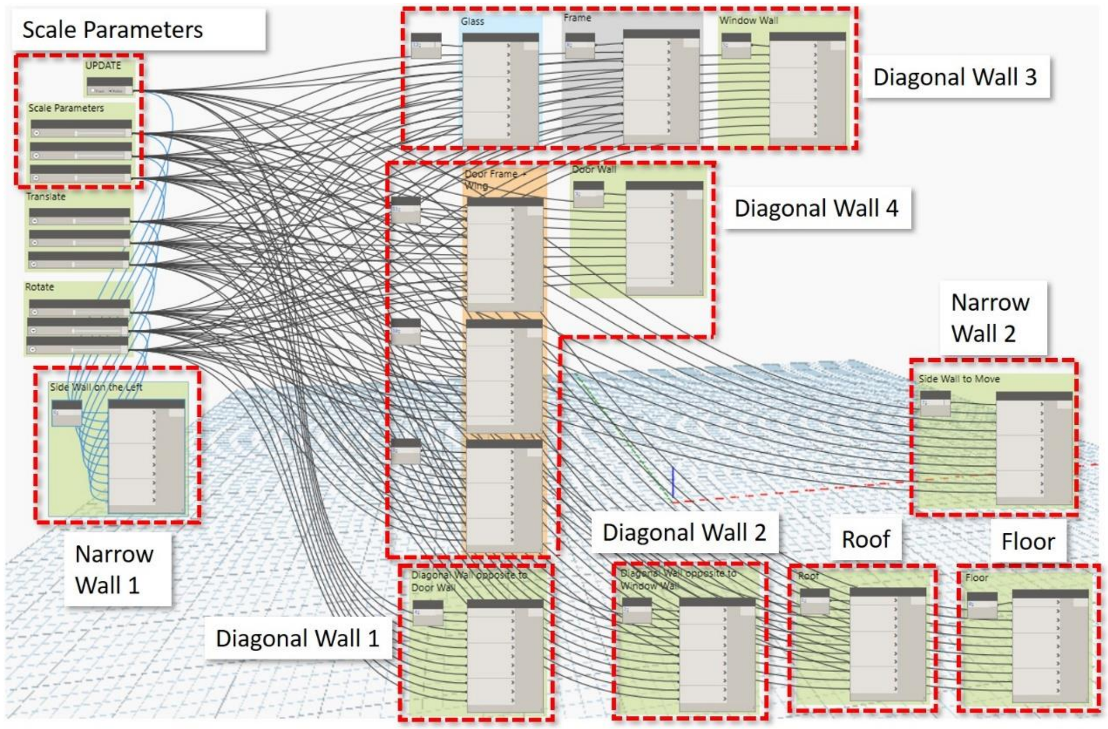


Figure 11. Parametric dataflow in Dynamo for Scale X-Y-Z

The Grasshopper user opened the Web-based model in one window and Rhino and Grasshopper in another window. Likewise, the Dynamo user opened the Web-based model in one window and Dynamo Studio in another window (Figure 12).

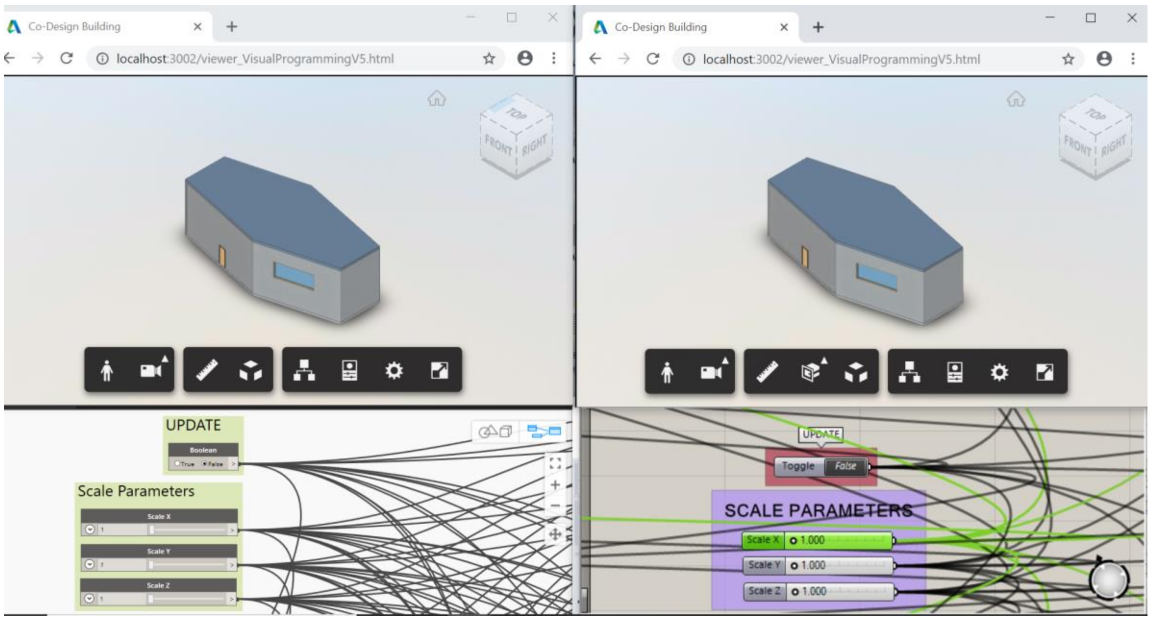


Figure 12. Test 1.1: Before Scale, Web-based model (top) and views of the Dynamo user (left) and the Grasshopper user (right)

Both users started to explore the overall form of the model by using scale parameters. First, the Grasshopper user changed the Scale X from 1 to 1.5. As a result, the walls, windows, door, roof, and floor stretched along the X-axis simultaneously, both in the Web-based model and Grasshopper twin model. The Dynamo user observed the parametric change in the Scale X in the Web-based model (Figure 13). Since the data transfer was through a local computer to the Web-based model, the Dynamo user needed to manually update the Dynamo file. Therefore, the user retrieved data from the Web-based model by using the measuring instruments embedded in the Web viewer. For this transformation, the Dynamo user measured the model along X axis; calculated the Scale X factor with the knowledge of original dimensions; and manually entered the value to the Scale X parameter in Dynamo to update the Revit model (Figure 14).

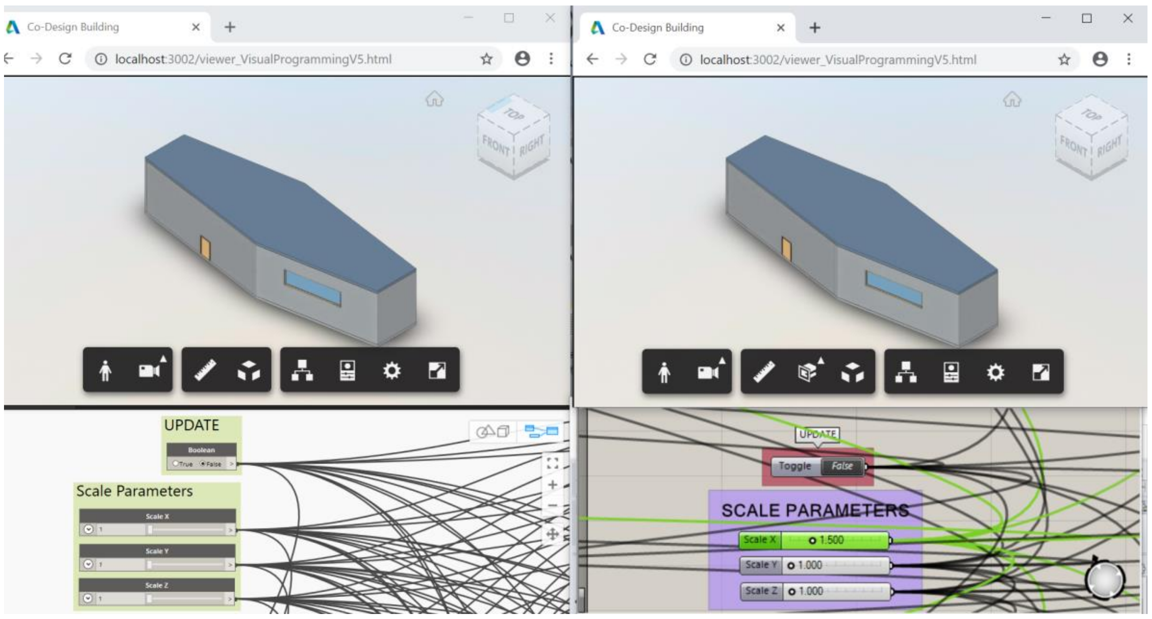


Figure 13. Test 1.1: After the Grasshopper user (right) changes Scale X = 1.5, both users observe the parametric changes in the Web-based model (top)

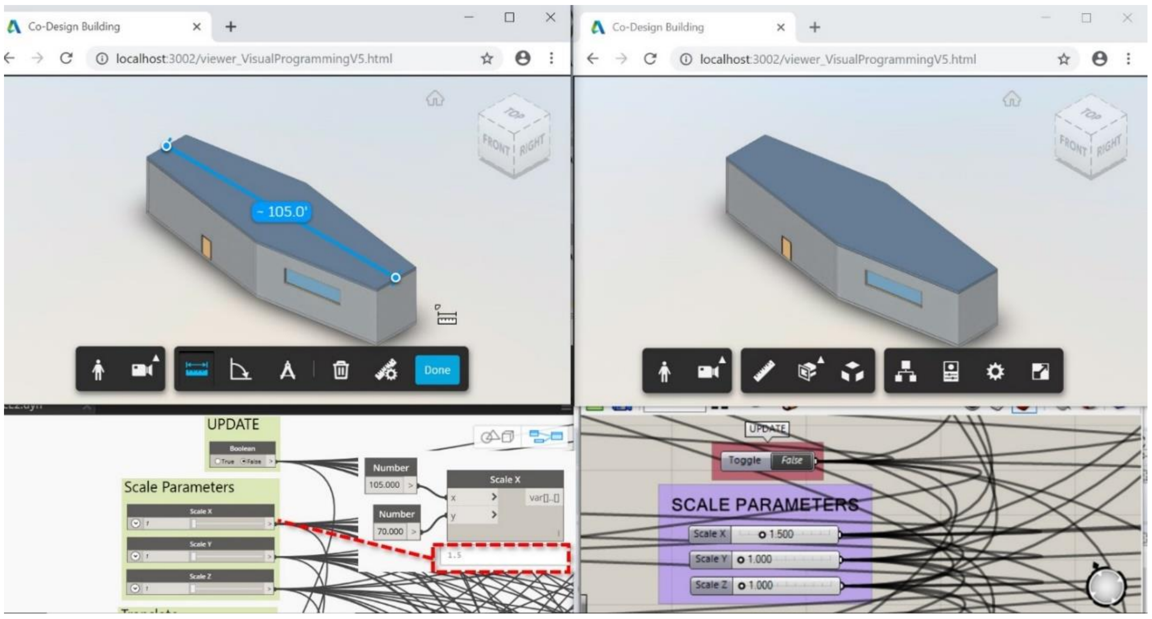


Figure 14. Test 1.1: The Dynamo user (left) retrieves data from the Web-based model and manually updates parameters in Dynamo to match the model

Afterwards, the Dynamo user changed the Scale Y value from 1 to 2 (Figure 15). Similarly, the Grasshopper user utilized the Web viewer's measure feature and manually entered the Scale Y value to the Grasshopper parameter to update the Rhino model (Figure 16). Lastly, the Grasshopper user repeated the same process for Scale Z by increasing the value from 1 to 2 (Figure 17). Consequently, the Dynamo user retrieved data from the Web-based model, and manually modified the Dynamo data flow.

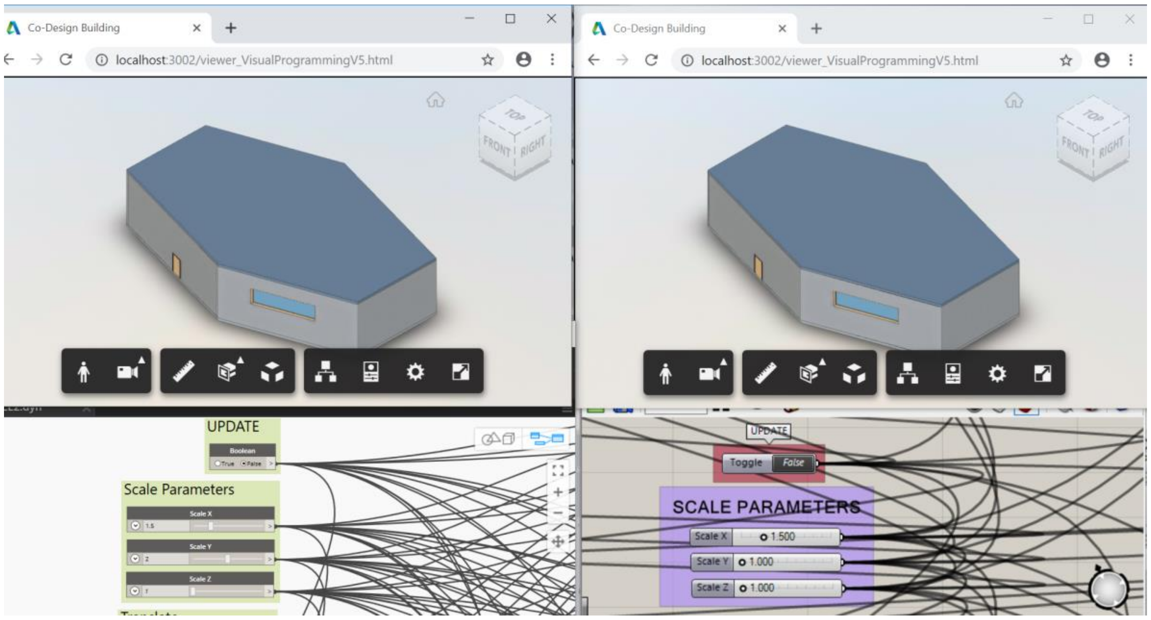


Figure 15. Test 1.1: After the Dynamo user (left) changes Scale Y = 2, both users observe the parametric changes in the Web-based model (top)

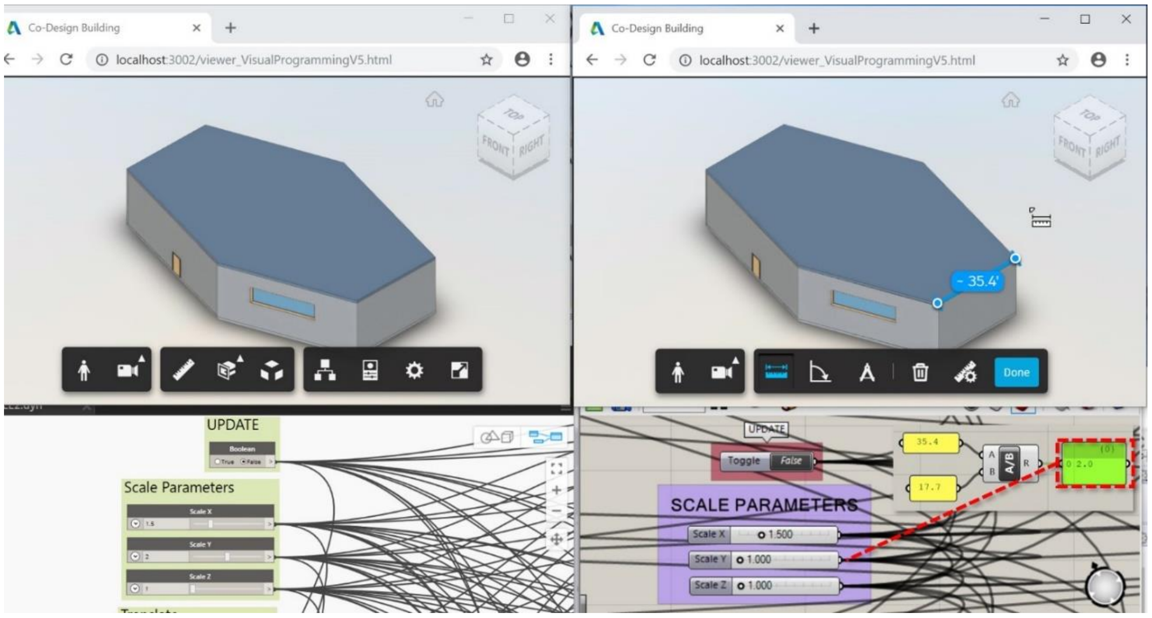


Figure 16. Test 1.1: The Grasshopper user (right) retrieves data from the Web-based model and manually updates parameters in Grasshopper to match the model

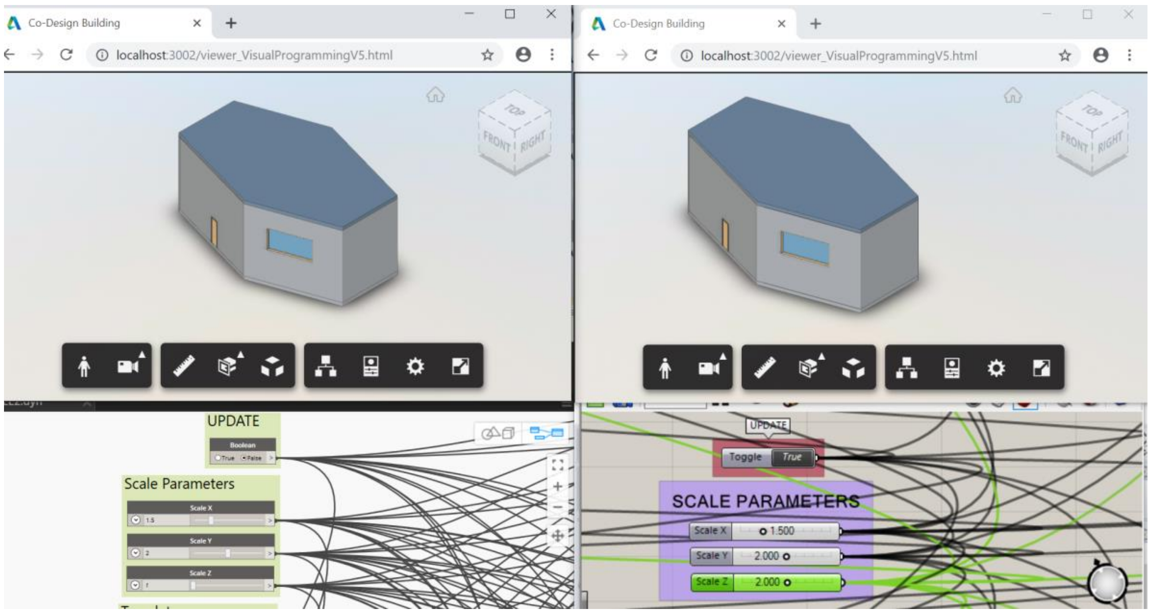


Figure 17. Test 1.1: After the Grasshopper user (right) changes Scale Z = 2, both users observe the parametric changes in the Web-based model (top)

All the steps were repeated backwards and by changing the order. The experiments showed that any change in Scale X, Scale Y, and Scale Z parameters by any user resulted in parametric change in both users' Web-based model. These findings demonstrate that the Web-based model is capable of parametrically responding to the value changes in scale parameters. Besides, the Web-based model provides a medium for users to collaborate while using different visual programming tools. However, there was a limitation in both Grasshopper and Dynamo; their number sliders of parameters do not allow to be updated by external data. Therefore, the updates in a Web-based model could not be transferred to the visual programming files. In order to collaborate, the users were required to measure the Web-based model, retrieve data from it, and manually update their visual programs. Additional coding can help to overcome the limitation of the data transfer.

4.1.2. Test 1.2 and Results

The goal in Test 1.2 is implementing collaboration and interoperability between Grasshopper and Dynamo through the Web-based model in Translation X. Translation was tested by simulating two users, namely the Grasshopper and the Dynamo users. The users made a copy of their visual programming files, previously used for scale transformation. Both data flow was modified to accommodate Translate. Users followed the same process as in scale transformation to access the Web-based model.

The translate data flow was set up in the same way by both users: Narrow Wall 1 was fixed and only Narrow Wall 2 was set to be translated. The Translation parameter was decided to be in X direction, and constrained between 0 and 30. Other related building elements, such as the window, roof, floor, connected walls were programmed to adapt their geometries to meet the translated Narrow Wall 2 – thus to behave as in a parametric model. Therefore, roof, floor, all diagonal walls with the door and the window had to extend in the positive direction of the X-axis. In order to provide this transformation, the value of Scale X of these elements was formulated depending on Translate X of Narrow Wall 2. The actual dimensions retrieved from the original Revit model were integrated into a formula of Scale X. Both users accessed the shared Web-based model in their localhost servers (Figure 18).

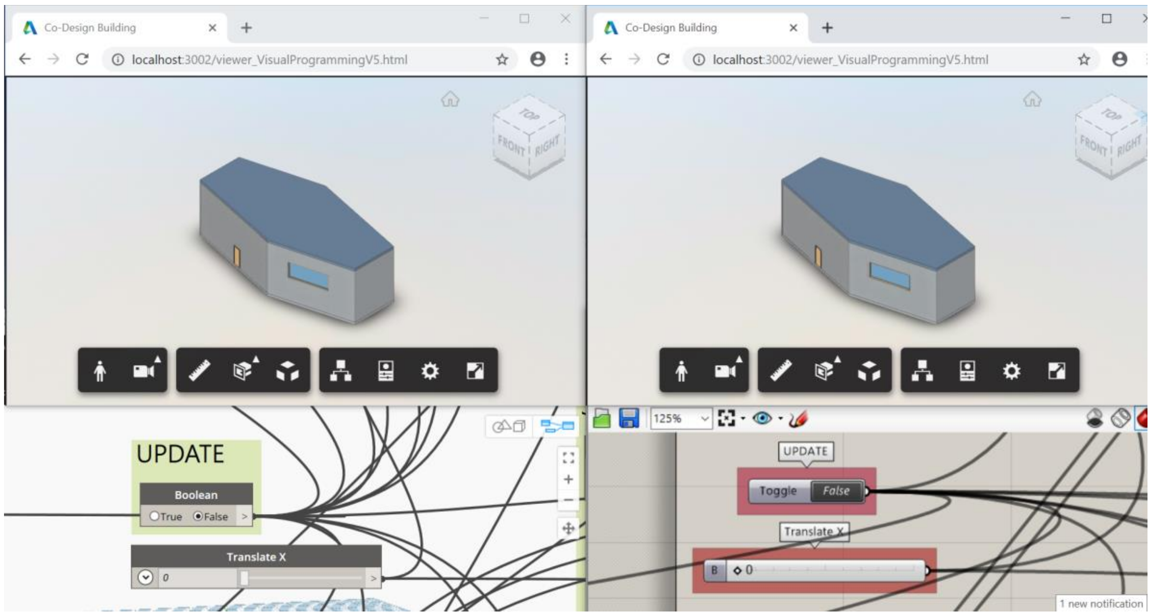


Figure 18. Test 1.2: Before Translation, Web-based model (top) and views of the Dynamo user (left) and the Grasshopper user (right)

Both users started to explore the overall geometry propagation based on translation of the Narrow Wall 2. First, the Dynamo user changed the Translate X value from 0ft to 15ft. The Narrow Wall 2 was translated; the roof, the floor, all the walls along with the door and the window were scaled in X direction and composed a closed geometry. All the parametric changes were observed on both of the users' Web-based models (Figure 19).

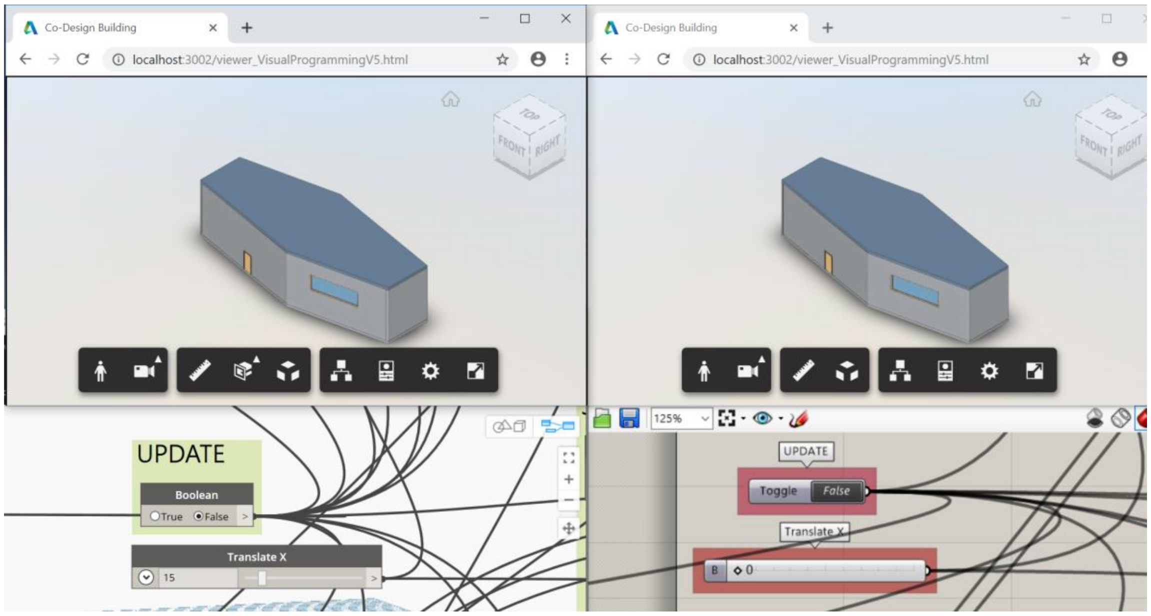


Figure 19. Test 1.2: After the Dynamo user (left) changes Translate X = 15ft, both users observe the parametric changes in the Web-based model (top)

Afterwards, the Grasshopper user identified the Translate factor of the parametric change done by the Dynamo user by measuring the Web-based model. The Grasshopper user calculated the translation factor and manually entered the data into Grasshopper to update the local twin model (Figure 20).

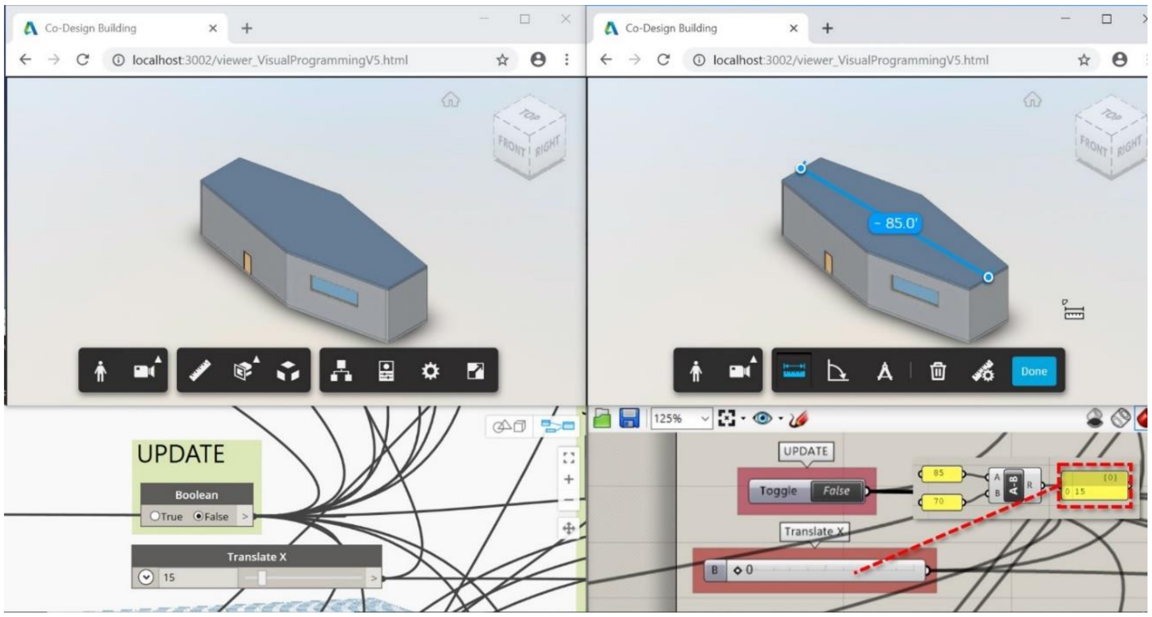


Figure 20. Test 1.2: The Grasshopper user (right) retrieves data from the Web-based model and manually updates parameters in Grasshopper to match the model

As the next step, the Grasshopper user moved the Narrow Wall 2 by increasing the Translate X value to 30ft. Consequently, other elements adapted their forms to meet the translated wall. The parametric change was observed in the Web-based models on both users' browsers (Figure 21). Next, the Dynamo user retrieved dimension data from the Web-based model, and after calculating the Translate X value, manually entered the translation value into the Translate X input (Figure 22).

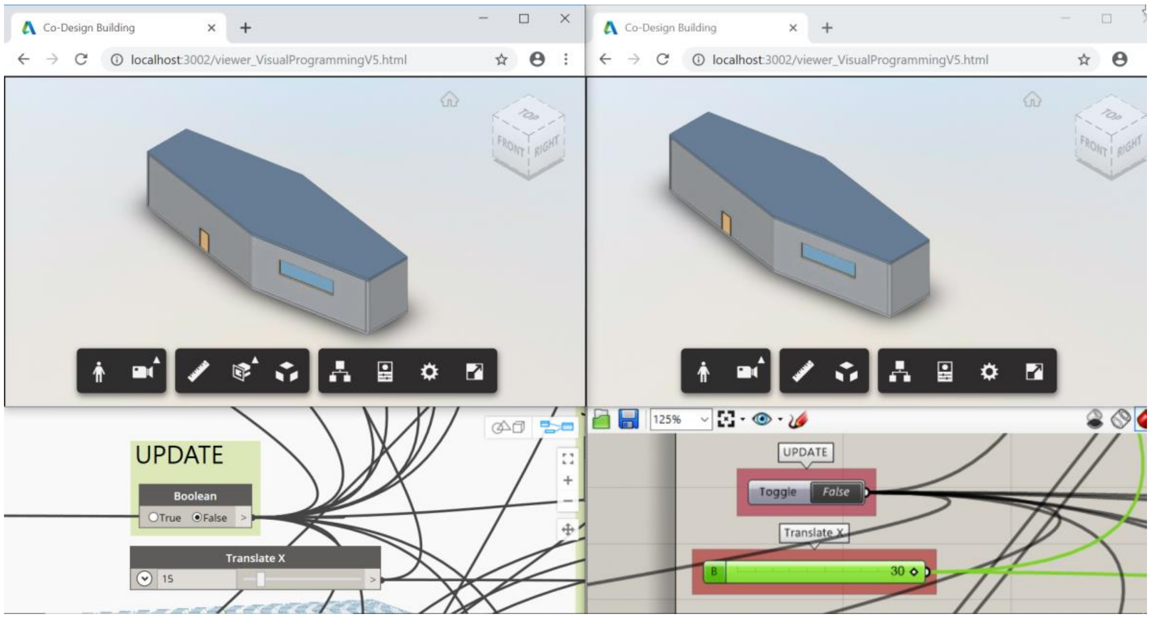


Figure 21. Test 1.2: After the Grasshopper user (right) changes Translate X = 30ft, both users observe the parametric changes in the Web-based model (top)

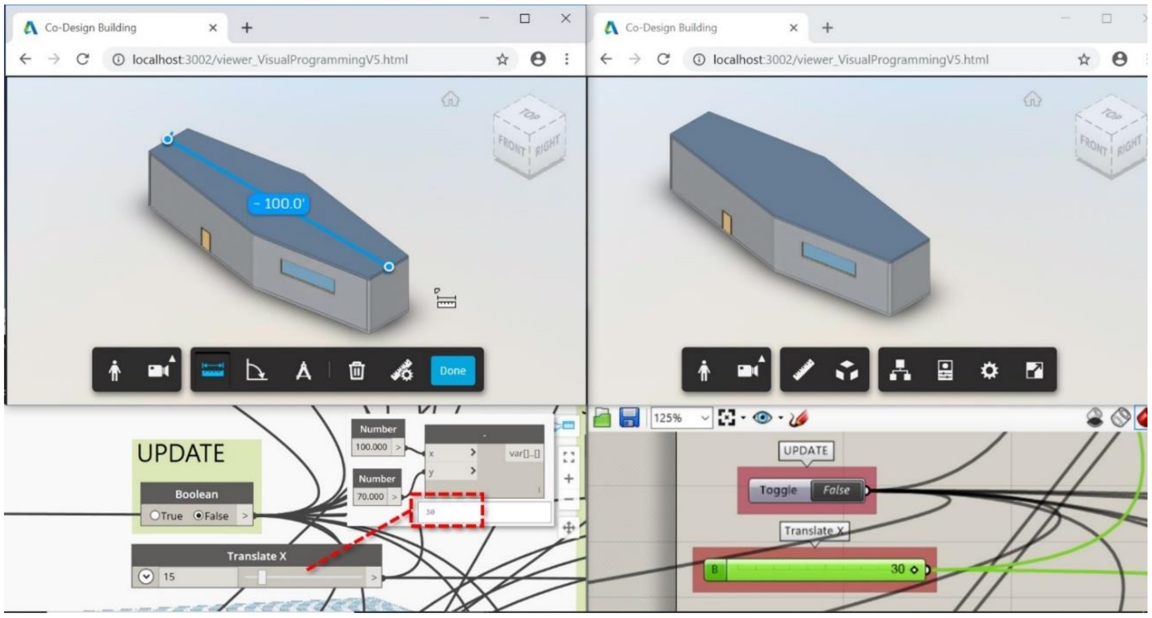


Figure 22. Test 1.2: The Dynamo user (left) retrieves data from the Web-based model and manually updates parameters in Dynamo to match the model

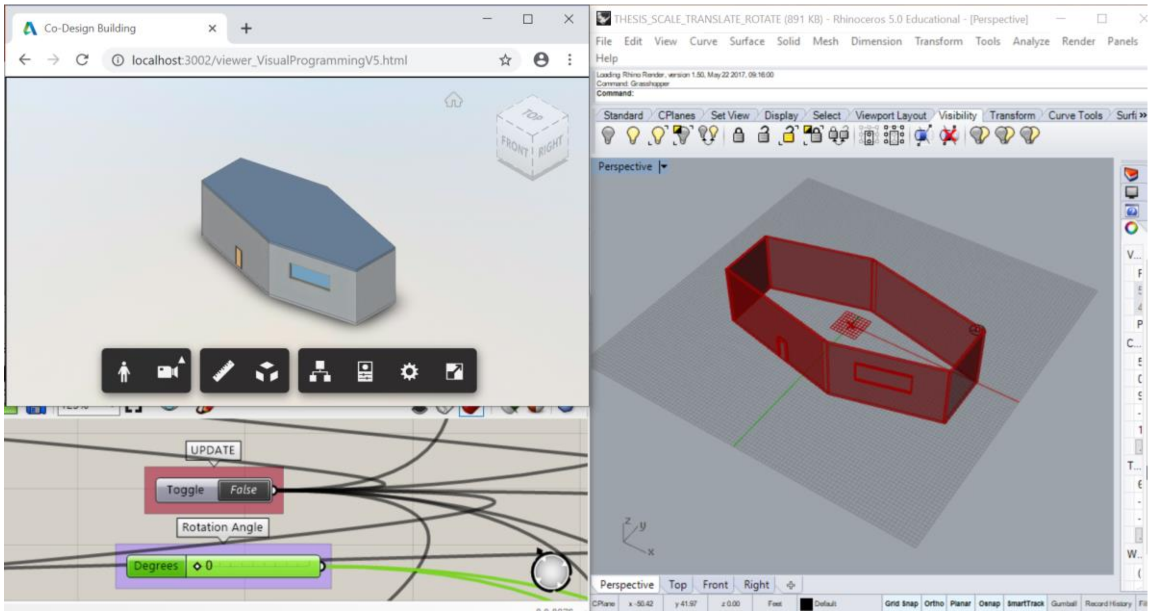
All the steps were repeated backwards and by changing the order. The experiments showed that, any change in Translate X parameter by any user, resulted in parametric change in both users' Web-based models automatically (but the changes in local twin model are made manually). By presenting visualization as well as data to retrieve, the Web-based model provided a medium for users to collaborate while using different visual programming tools. Because of the limitation in Grasshopper and Dynamo for the use of their sliders, the number sliders of parameters could not be updated by the external data received from the Web-based model. Therefore, the data was required to be retrieved from the Web-based model by using the instruments of the Web-based model, and manually input into the visual programming file by the user. Additional scripting can help to overcome this limitation by providing reciprocal data transfer.

4.1.3. Test 1.3 and Results

Due to the complexity of parametric rotation, Rotation was tested by simulating only a single Grasshopper user currently. The Grasshopper user created a copy of the Scale transformation file and modified the data flow to accommodate Rotate. For Rotate, Diagonal Wall 3 was chosen to rotate around the Z-axis where this wall meets Narrow Wall 2. The parametric relationships were defined for Diagonal Wall 4 in order to make this wall to respond the rotation by adapting its geometry. The door on the Diagonal Wall 4 and the window on the Diagonal Wall 3 were also modified to rotate simultaneously with their host walls. Other geometries were fixed. Also due to the complexity, a twin model was utilized to calculate the scale and translation factors of an adjacent wall to match the rotated one.

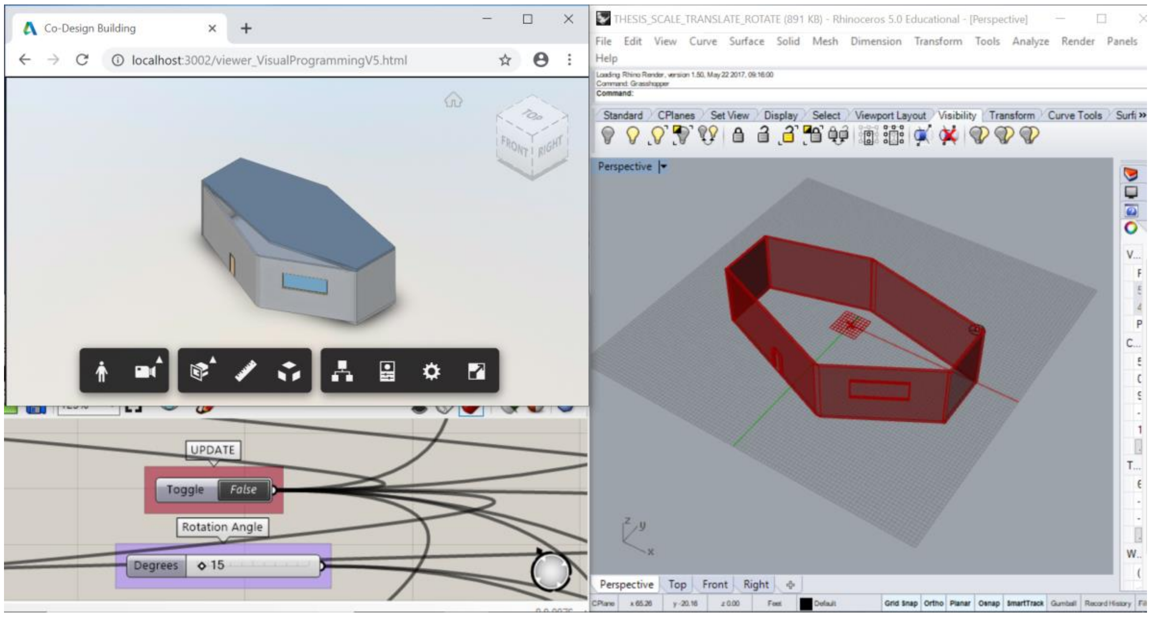
As previously mentioned, the center of the bounding box of the model constitutes the center point of the Web-based model. All the rotation transformations made on the Web-based model reference to the Z-axis with this center as the origin. After each rotation of Diagonal Wall 3, the twin geometry of Diagonal Wall 3 was translated to the location where the geometry was expected to be. By translation, not only the twin model and the Web-based model were synchronized, but also the data to calculate the rotation angle and the scale factor of Diagonal Wall 4 was obtained.

In order to test the transformation, the Grasshopper user connected to the Web-based model through the localhost server, and opened Rhino, Grasshopper file that included the data flow designed for Rotate, and the twin model of the Web-based model (Figure 23). Then, the Grasshopper user changed Rotation Angle of Diagonal Wall 3 from 0° to 15°. Consequently, Diagonal Wall 3 rotated and Diagonal Wall 4 rotated and scaled on both the twin model and the Web-based model (Figure 24).



Co-Design Building
localhost:3002/viewer_VisualProgrammingV5.html
THESIS_SCALE_TRANSLATE_ROTATE (891 KB) - Rhinoceros 5.0 Educational - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Panels Help
Standard CPlanes Set View Display Select Viewport Layout Visibility Transform Curve Tools
Perspective
UPDATE
Toggle False
Rotation Angle
Degrees 0
Perspective Top Front Right

Figure 23. Test 1.3: Before Rotation, Web-based model (top left) and views of the Rhino (right) and the Grasshopper (bottom left)



Co-Design Building
localhost:3002/viewer_VisualProgrammingV5.html
THESIS_SCALE_TRANSLATE_ROTATE (891 KB) - Rhinoceros 5.0 Educational - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Panels Help
Standard CPlanes Set View Display Select Viewport Layout Visibility Transform Curve Tools
Perspective
UPDATE
Toggle False
Rotation Angle
Degrees 15
Perspective Top Front Right

Figure 24. Test 1.3: After the Grasshopper user (bottom left) changes Rotation Z = 15°, the user observes the parametric changes in the Web-based model (top left) and Rhino (right)

Parametric responses of roof and floor were out of scope for this step due to the complexity, since those calculations would require a modified Grasshopper Viewer plugin. Diagonal Wall 3 and Diagonal Wall 4 did not meet precisely at their joint, which would also require more time to solve. Also, Diagonal Wall 4 moved away from the location where it meets Narrow Wall 1 as the Rotation Angle increased (Figure 25). Dynamo user collaboration was decided to be a future work, since that would also require extra coding for the plugin.

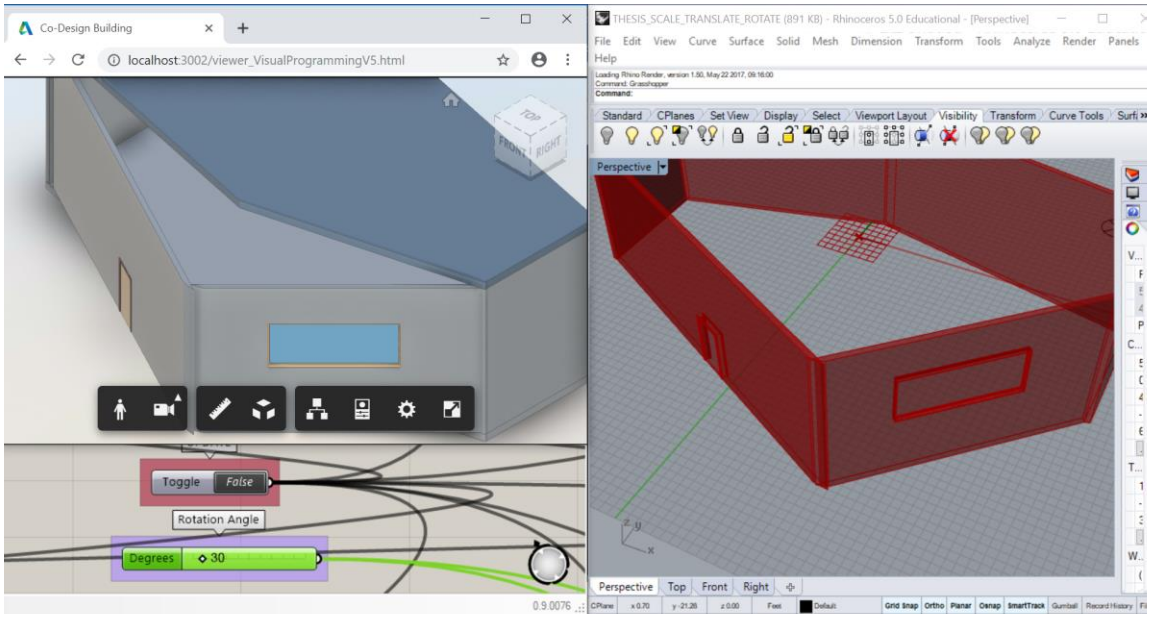

Co-Design Building
localhost:3002/viewer_VisualProgrammingV5.html
Toggle
False
Rotation Angle
Degrees
30
THESIS_SCALE_TRANSLATE_ROTATE (891 KB) - Rhinoceros 5.0 Educational - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Panels Help
Standard CPlanes Set View Display Select Viewport Layout Visibility Transform Curve Tools
Perspective
Perspective Top Front Right

Figure 25. Test 1.3: After the Grasshopper user (bottom left) changes Rotation Z = 30°, the user observes the limitations of the parametric changes in the Web-based model (top left) and Rhino (right)

4.1.4. Discussion

In Case Study 1, the Web-based model was examined through Scale, Translate, and Rotate transformations. The Scale and Translate tests provided a collaborative environment between the Grasshopper and Dynamo users. Both users were able to control the Web-based model and got the updates on the Web page. However, the number sliders of both Grasshopper and Dynamo have a limitation; they do not allow to be updated by external data which came from the Web-based model. Therefore, the data transfer was one-way, from the visual programming tools to the Web-based model; the updates on the Web-based model could not be automatically viewed on the visual programming tools. Instead, the users were required to measure the Web-based model, retrieve data by using the Viewer's own instruments, and manually update their visual programs.

The elements were not parametric by default, but the users were required to build the parametric relationships. The elements did not adapt their geometries at the joints to provide a precise union. Also, there was no access to control geometry vertices, so that, the sides of the walls did not provide a clean joint after scaling in Test 1.2. Similar problem was observed in Test 1.3, as the rotated walls did not meet at a clean joint. Also, the roof and the floor were not able to parametrically respond to the rotation of the walls because non-uniform scaling is not available. The solution to the both problems is the same, which requires users to be able to control the geometries through the vertices, which is not enabled by the current Grasshopper and Dynamo Viewer plugins. All of the above-mentioned limitations could be overcome by using additional scripting.

4.2. Case Study 2

In this case study, two simulated users, both using Rhino, Grasshopper and sharing the same Web-based model, collaborated on an optimization process. The goal is to examine the ways of collaboration between the users using the Web-based model framework during different types of optimization. For this case study, the roof of the building was defined as parametric, while the rest of the elements remained static; and the parameters of the roof proliferated the intended roof shape and building performance results in an iterative simulation and optimization process. Three optimization tests were defined: Test 2.1, Test 2.2, and Test 2.3. In Test 2.1 the objectives were maximum preferred daylight, and minimum roof area that creates a shading in the south. In Test 2.2, the objectives were minimum energy consumption and minimum roof area with a shading in the south. In Test 2.3, even though there were initially three objectives in the integrated optimization, two of them were combined by using the weighted sum method. The maximum preferred daylight and minimum roof area with a shading in the south constituted one objective, and the minimum energy consumption was the other objective in Test 2.3.

For this case study, a new Web-based model was used. Since daylight simulation and optimization would be run, the building would require more windows for a better daylight distribution. Two types of windows were added: 16’x8’ windows were located on diagonal walls, except the one that had a door; 5’x8’ windows were located on narrow side walls, and the wall with the door. A twin model of the Web-based model was created in both users' local computers by using Rhino and Grasshopper. The twin model would constitute the input geometry for conducting simulation and optimization; the simulation and optimization results could be visualized with the help of twin geometry. However, this twin geometry is different than the previous twin geometry used in Case Study 1. As previously mentioned, the simulation engines only work with surfaces. The twin geometry should be composed of single planes in order to be used as input geometry. Thus, a new model is built in Rhino with reference to the model, which was previously imported from Revit. The Rhino model is carried out to Grasshopper by referencing each surface as Breps and converted into Honeybee surfaces and zone in order to be used in both daylight and energy simulation and optimization.

For setting and running daylight and energy simulation, Honeybee was used in combination with Ladybug. Ladybug served to import weather data and visualize the simulation results. Honeybee provided running daylight simulation in RADIANCE and energy simulation in EnergyPlus. Since the Web-based model is a parametric model, the daylight model is also a parametric model; the roof element has the parameters for scale and rotation values. The roof, floor, and each wall and window were created individually as Honeybee surfaces so that they are identical to the Web-based model. Honeybee provides access to use materials both from RADIANCE and EnergyPlus libraries. Also, it is possible to define a custom-made Radiance material. For the daylight simulation, glass was defined upon Radiance glass material; wall, floor, and roof materials were chosen from the RADIANCE library. For wall, floor, and roof materials, two types were specified in RADIANCE library as Exterior and Interior; yet, both of them had the same values. Exterior wall, floor, and roof reflectance values are 50%, 20%, and 80%, which are denoted as 0.5, 0.2, and 0.8 for RADIANCE respectively. Visible transmittance was specified for the glass material. Table 4 shows the details of RADIANCE materials used.

Table 4. RADIANCE materials

Model Element	**RADIANCE Material**	**Material Type**	**Values**
Wall	Exterior_Wall	Plastic	Reflectance: 0.5 Specularity: 0 Roughness: 0
Floor	Exterior_Floor	Plastic	Reflectance: 0.2 Specularity: 0 Roughness: 0
Roof	Exterior_Roof	Plastic	Reflectance: 0.8 Specularity: 0 Roughness: 0
Glass	Radiance Glass Material	Glass	Transmittance: 0.65

For the energy simulation, EnergyPlus materials were chosen to specify the optimum thermal properties of the elements. EnergyPlus presents a list of building construction elements classified according to the standards ASHRAE and CBECS with varying U-values. The building materials are categorized according to climate zones. Since the inspiring project for the Web-based model is located in Geneva, Switzerland, Geneva is considered as the location of the Web-based model. According to U.S. Department of Energy [DOE] (2015), climate zone for Geneva, Switzerland is 4A, mixed humid. Table 5 shows the selected materials for wall, window, roof, and floor from the related climate zone materials regarding ASHRAE 90.1-2010.

Table 5. EnergyPlus building construction materials

Model Element	**EnergyPlus Building Construction Material**	**U-Value (Btu/h·ft^2·°F)**
Wall	ASHRAE 90.1-2010 EXTWALL WOODFRAME CLIMATEZONE 1-4	0.1
Window	ASHRAE 90.1-2010 EXTWINDOW NONMETAL CLIMATEZONE 4	0.4
Roof	ASHRAE 90.1-2010 EXTROOF IEAD CLIMATEZONE 2-8	0.05
Floor	EXTERIOR FLOOR	0.1

Both users had the same data flow for daylight and energy simulation and optimization in order to conduct optimization and retrieve data. Octopus, a Multi-Objective Evolutionary Optimization plugin for Grasshopper was utilized, since all three experiments employed multiple objectives. Octopus calculates the best set of solutions known as Pareto front, within the limits of genomes (parameters) according to fitness functions. In a Pareto front, no improvement in a solution can be made without degradation in at least one of the remaining solutions (Fonseca & Fleming, 1995). The ranges of the parameters' values remained the same as in the Case Study 1 (Table 3). The Scale X, the Scale Y, and the Rotation Z parameters of the roof constituted the genomes of Octopus for all tests. The fitness functions were defined in accordance with the optimization. In the daylight optimization, the preferred daylight illuminance values range between 750 lux and 2000 lux. Honeybee creates sensors to measure the illuminance levels. The aim is to maximize the number of sensors that receive illuminance between 750 lux and 2000 lux for daylight optimization.

A requirement for both the daylight and energy optimization is the minimum roof area while creating a shading for the south (in the meantime covering the entire building floorplan). To create a shading, a point located 3ft southeast from the south corner of the model at the roof level is considered to be covered by the roof. In order to formulate this, Point In Curve node in Grasshopper was utilized. Point In Curve node tests if a point is in a curve or not by assigning the coverage indices: 0 for outside, 1 for coincident, and 2 for inside. In order to cover the room area and the chosen point at the roof level in the south, all corner points of the room and the chosen point should be inside the curve of the roof polyline. Since there were 6 corners for roof and an additional point for the shading, the total 7 points yield a maximum sum of coverage indices 14. If the Coverage Index is equal to 14, the roof covers both the room and the point for shading. Therefore, in formulating the roof shape requirement, the Coverage Index had to be maximized while the roof area had to be minimized. The roof area was represented by a multiplication of the Scale X and Scale Y parameters. The objectives are formulated in detail in subsections 4.2.2, 4.2.3 and 4.2.4.

4.2.1. Test 2.1 and Results

Test 2.1 was conducted to investigate the ways of how two Grasshopper users could collaborate through the Web-based model while optimizing the roof shape and achieving the preferred daylight intake. Daylight simulation data flow in the twin model was created with Ladybug and Honeybee in Grasshopper based on the definition of the geometric data flow. The daylight simulation type was grid-based, and a work plane was defined at the height of 2.5ft for the grid. The sensors were set 5ft apart to minimize the simulation time. According to U.S. DOE (2015), climate zone for Geneva, Switzerland is 4A, mixed-humid. When the annual dry bulb temperature (Figure 26) and monthly diurnal average (Figure 27) graphics are examined, it is concluded that the dominant energy loads for Geneva are the heating loads. According to the graphs, the temperatures are below the comfort zone for most of the year. Thus, the month of December was chosen for both energy and daylight simulations. December 21st was tested with a standard CIE sky, sunny with sun, for 12 hours. CHE_Geneva.067000_IWEC.epw weather file was utilized.

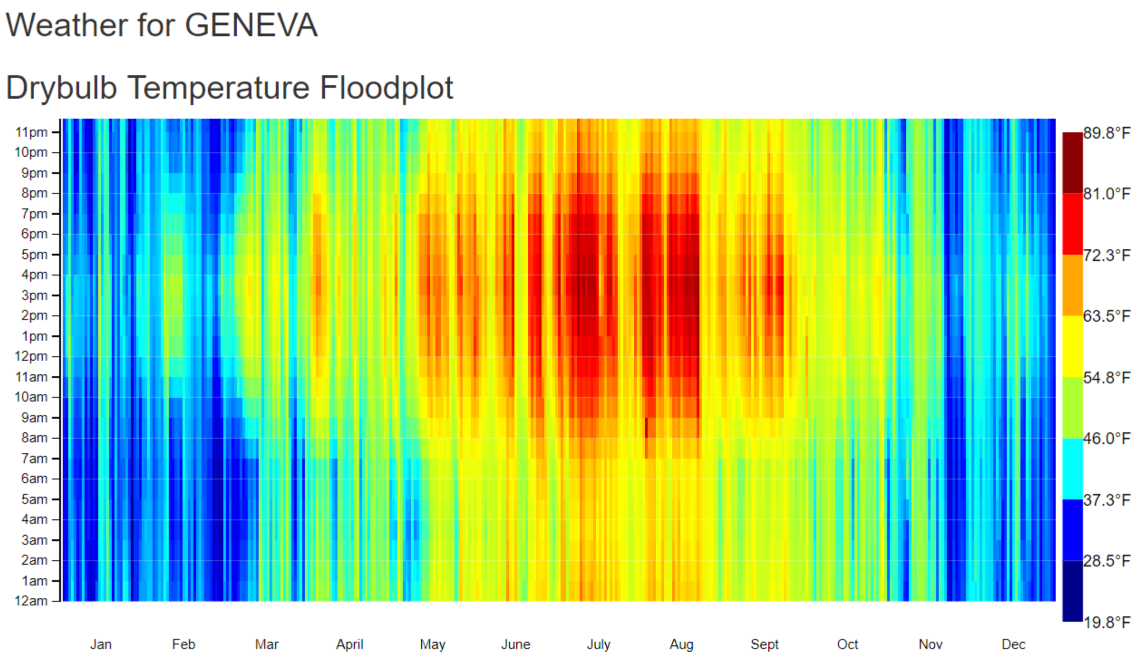
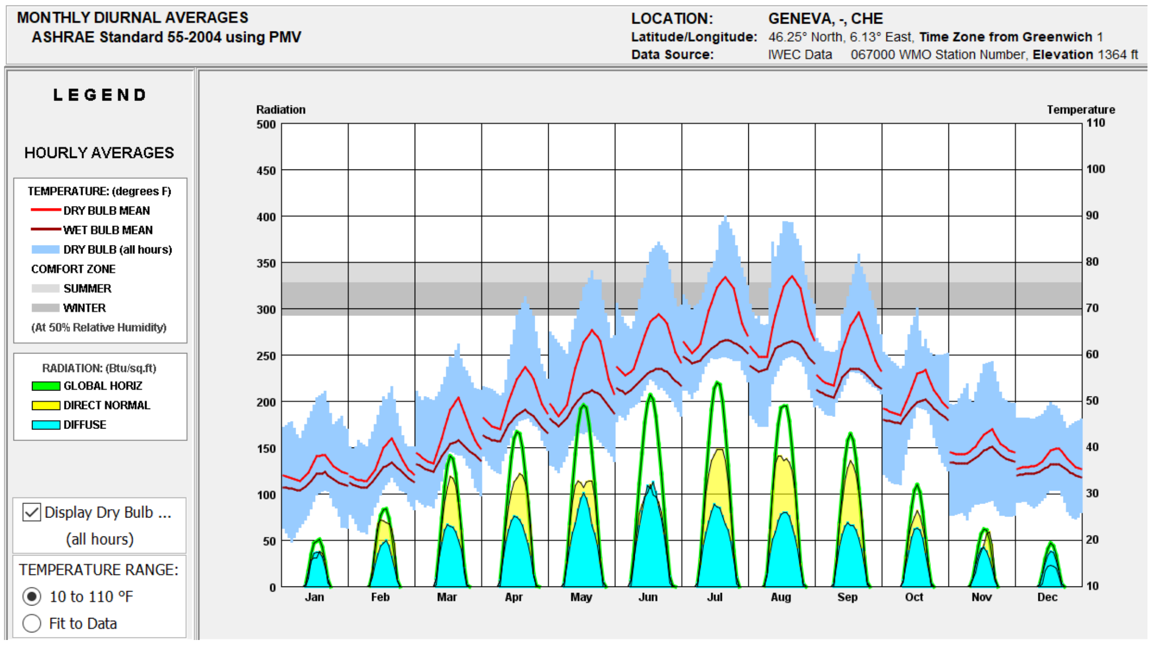


Figure 26. Annual dry bulb temperature for Geneva (Analyzed in epwvis)



MONTHLY DIURNAL AVERAGES
ASHRAE Standard 55-2004 using PMV

LOCATION: GENEVA, -, CHE
Latitude/Longitude: 46.25° North, 6.13° East, Time Zone from Greenwich 1
Data Source: IWEC Data 067000 WMO Station Number, Elevation 1364 ft

Figure 27. Monthly diurnal averages (Analyzed in Climate Consultant 6.0)

RADIANCE materials and reflectance values were determined for walls, roof and floor; visual transmittance value was determined for glass. RADIANCE materials and their specifications were previously mentioned in Table 4. Although, the daylight simulation and optimization were tested for a chosen day and time, it is possible to conduct the simulation for another day or a period of time, e.g annual daylight simulation, coupled with another sky type. Likewise, another metric could be utilized instead of using the metrics specified for this thesis. The chosen metrics, daylight simulation date and time, the function and the daylight requirement of the Web-based model are examples which could be replaced in accordance with the design of the project and the intended daylight simulation goals.

The model was oriented towards the same direction as the inspiring house, aligning northwest and southeast direction. Building program was set to midrise apartment. Illuminance was simulated for the given date and time. In this case study, the illuminance levels on the work plane were tested since the aimed levels were expected to serve the activities such as writing, reading, and typing. The floor level could also be tested in the same way. After obtaining the list of illuminance values, the list was filtered to calculate the number of sensors that received the illuminance levels in the 750-2000 lux range. The daylight simulation can be tested for various dates, times, sky conditions, weather files and orientations. Similarly, it can be tested for different forms, and since the Web-based model is a parametric model, the scale, translation and rotation of each element can be changed through the parameters.

The number of all sensors for daylight was 104, since Honeybee automatically created and distributed them on the work plane with 5ft apart. However, the number of sensors that received the preferred daylight range was dependent on the shape of the roof and other parameters such as the room shape and windows. The number of filtered sensors was multiplied by -1 in order to formulate the number of sensors into a fitness function, mainly because, Octopus generates solutions for minimizing objectives. Since Octopus is a multi-objective optimization tool, a set of design solutions were generated at the end of the optimization process that satisfy the objectives. This set of solutions, called Pareto front, provides an optimal condition that none of the solutions can be improved without degradation in at least one of the remaining solutions (Fonseca & Fleming, 1995).

For the second objective, the roof was aimed to cover the room area, as well as the point at the roof level in the south. Besides, the roof area was required to be minimized. All these objectives were combined in a single formula to produce a fitness function. Scale X was denoted by x, Scale Y was denoted by y, and the multiplication of Scale X and Scale Y should be minimum to minimize the roof area. The Coverage Index was denoted by z, ensured to cover both the roof area and the point at the roof level in the south. Both the number of sensors that receive the preferred daylight and the Coverage Index were required to be maximized and converted into fitness functions which seek for the minimum. Thus, the number of sensors and the Coverage Index were both multiplied by -1.

The objectives were defined with two equations; the first one referred to the daylighting, the second one referred the roof shape:

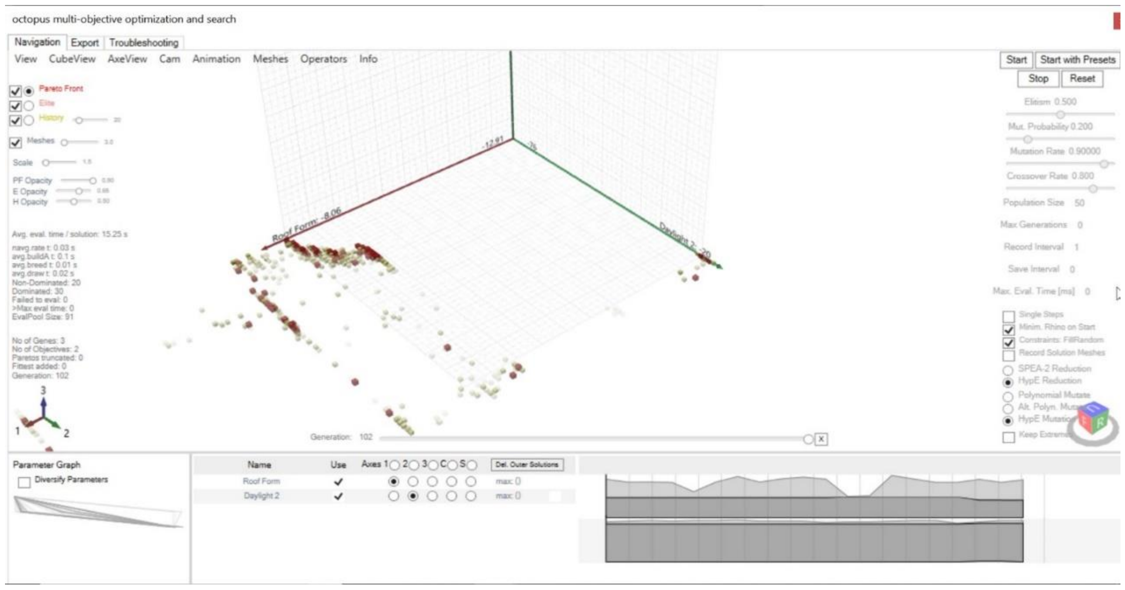
Fitness Function 1: min $(-u)$

Fitness Function 2: min $(x \cdot y - c)$

where *u* is the number of the sensors that received 750-2000 lux, *x* denotes the Scale X, *y* represents the Scale Y, and *c* stands for the Coverage Index. The Scale X, Scale Y and Rotation Angle parameters of the roof constituted the genomes for Octopus – the Genetic Algorithm (GA) tool in Grasshopper. Parameter value ranges were previously mentioned in Table 3.

In search of the optimum roof shape and daylight, Honeybee was run to simulate the daylight for all solutions generated by Octopus. In this iterative process, Octopus generated different design options, and after each generation, the solutions got closer to the optimum solutions as a trend. The design solutions were displayed in the design space; each solution could be regenerated by reinstating the solution. The computer used in the optimization process has Intel® Core™ i7-7500U CPU @ 2.70 GHz 2.90 GHz processor, 8.00 GB RAM installed memory, and Windows 10 operating system. The population was set to 50 which can provide diverse number of solutions after each generation. Simple Genetic Algorithms (sGA) have been terminated under three conditions: (1) When the number of generations reached an upper limit; (2) when the number of evaluations of the fitness function reached an upper limit; (3) when there is an extremely low chance of achieving significant changes in the next generations (Safe et al., 2004). In this instance, the user frequently checked the solution set in Octopus graph.

After Octopus had proliferated generations more than 100, the solution set was converged, and the user observed that the chance of obtaining changes in the next generations was extremely low. Therefore, the user stopped the optimization process. GA was run for 102 generations which took almost 18 hours. It should be noted that the optimization can be paused when needed, yet the file is saved each time since Octopus allows for resuming optimization. After 102 generations of running GA, 14 Pareto front (overall best) solutions were obtained (Figure 28). The Pareto-optimal solution set for roof shape ranged between -8.06 and -12.91 while the daylight solutions ranged between -20 and -76. The red cubes represent Pareto-optimal solutions, and each solution was numbered in order to map the geometric outcomes on the graph (Figure 29).



Avg. eval. time / solution: 15.25 s
navg.rate t: 0.03 s
avg.buildA t: 0.1 s
avg.breed t: 0.01 s
avg.draw t: 0.02 s
Non-Dominated: 20
Dominated: 30
Failed to eval: 0
>Max eval time: 0
EvalPool Size: 91

No of Genes: 3
No of Objectives: 2
Paretos truncated: 0
Fittest added: 0
Generation: 102

Generation: 102

Elitism 0.500
Mut. Probability 0.200
Mutation Rate 0.90000
Crossover Rate 0.800
Population Size 50
Max Generations 0
Record Interval 1
Save Interval 0
Max. Eval. Time [ms] 0

- [ ] Single Steps
- [x] Minim. Rhino on Start
- [x] Constraints: FillRandom
- [ ] Record Solution Meshes

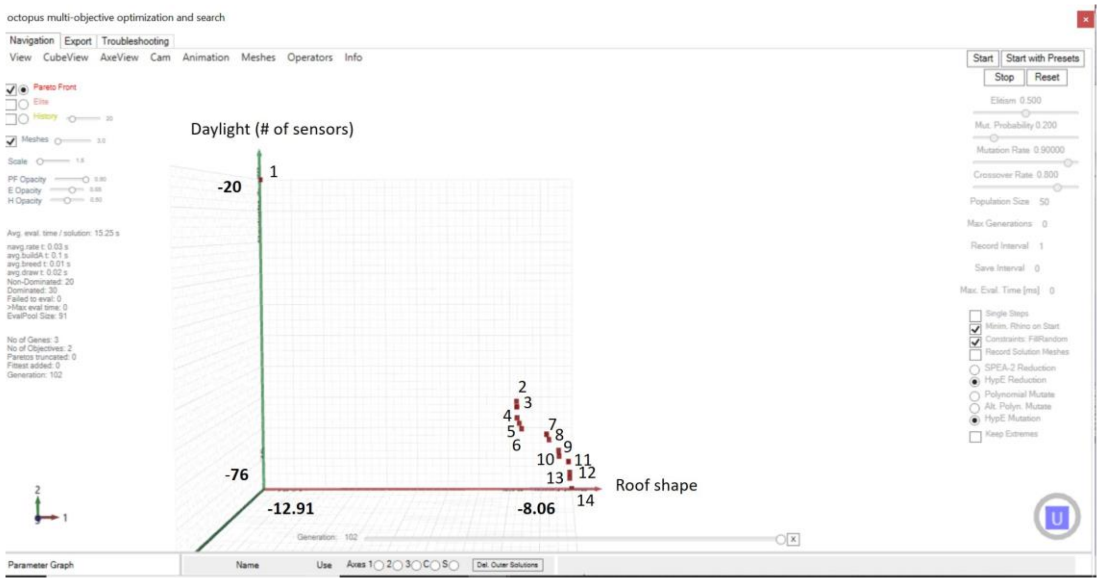
○ SPEA-2 Reduction
● HypE Reduction

○ Polynomial Mutate
○ Alt. Polyn. Muta
● HypE Mutatio

- [ ] Keep Extreme

Name	Use	Axes 1	2	3	C	S	
Roof Form	✓	●	○	○	○	○	max: 0
Daylight 2	✓	○	●	○	○	○	max: 0

Figure 28. Test 2.1: Search space for daylight and roof shape optimization with all solutions after 102 generations



```
Avg. eval. time / solution: 15.25 s

navg.rate t: 0.03 s
avg.buildA t: 0.1 s
avg.breed t: 0.01 s
avg.draw t: 0.02 s
Non-Dominated: 20
Dominated: 30
Failed to eval: 0
>Max eval time: 0
EvalPool Size: 91

No of Genes: 3
No of Objectives: 2
Paretos truncated: 0
Fittest added: 0
Generation: 102
```

```
Elitism 0.500
Mut. Probability 0.200
Mutation Rate 0.90000
Crossover Rate 0.800
Population Size 50
Max Generations 0
Record Interval 1
Save Interval 0
Max. Eval. Time [ms] 0
[✓] Minim. Rhino on Start
[✓] Constraints: FillRandom
(●) HypE Reduction
(●) HypE Mutation
```

Figure 29. Test 2.1: Search space and Pareto front for daylight and roof shape optimization

All 14 solutions in the Pareto front were individually visualized in Rhino to find the best options (Figure 30). The solution #1 presented the best values for the roof shape, where the solution #14 presented the best values for daylight. However, except the solution #1, none of the other solutions provided an appropriate room coverage.

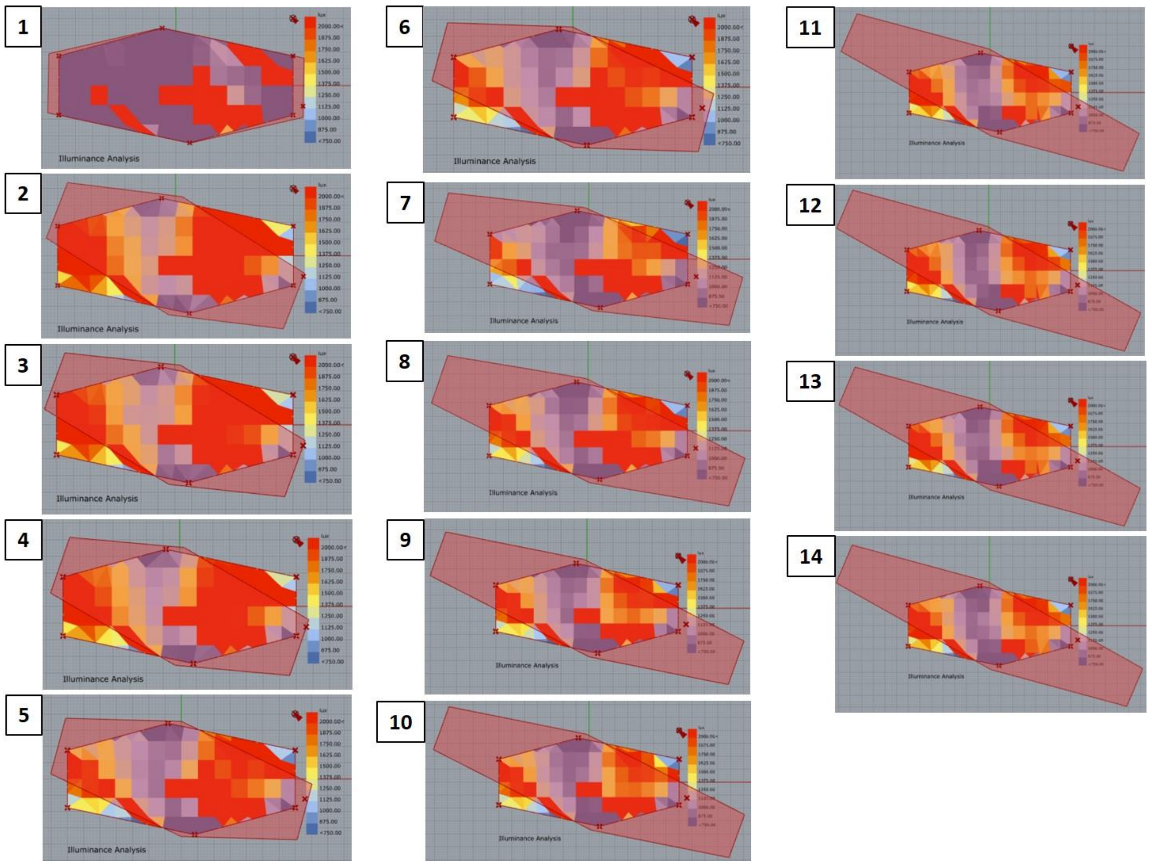


Figure 30. Test 2.1: Pareto front solutions visualized in the building floorplan view

The data from Pareto front was transferred to a table for a better understanding of results and further use (Table 6). Scale values, rotation angle, coverage index, number of sensors, and results of objective 2 formula were included. Only the solution #1 provided 14 for Coverage Index, which represents the coverage of the room and the point. Also, this solution provided the third minimum roof area. However, the number of sensors with the aimed illuminance range was also the lowest. The best daylight results belonged to solution #14 with 74 sensors. Maximum roof area was also given by solution#14, but the roof did not cover 2 of the room corners.

Table 6. Daylight optimization Pareto front data

Solutions #	**Scale X**	**Scale Y**	**Rotation Angle**	**Coverage Index**	**Number of Sensors in satisfactory range**	**(x*y)-z**
1	1.090	1.000	359	14	19	-12.91
2	1.077	1.000	339	10	56	-8.92
3	1.081	1.000	340	10	60	-8.92
4	1.084	1.000	341	10	60	-8.92
5	1.120	1.000	345	10	62	-8.88
6	1.156	1.000	345	10	63	-8.84
7	1.547	1.000	344	10	64	-8.45
8	1.586	1.000	341	10	65	-8.41
9	1.738	1.000	340	10	69	-8.26
10	1.745	1.000	340	10	70	-8.26
11	1.891	1.000	337	10	68	-8.11
12	1.912	1.000	337	10	72	-8.09
13	1.913	1.000	337	10	73	-8.09
14	1.942	1.000	337	10	74	-8.06

All the simulation and optimization process were completed on one user’s local computer, using Rhino, Grasshopper, Ladybug, Honeybee, and Octopus. Once the process was over, the Grasshopper user decided which solution was the overall best and transferred the chosen geometry into the Web-based model. In this case, the user chose the solution #1 as overall best and visualized it on the Web (Figure 31), while the other Grasshopper user observed it through her/his own Web-based model, and retrieved data from it. Since the data is transferred from local control file to the Web-based model, one user actively participates in the optimization process; while the other user passively observes the results on the shared Web-based model, and retrieves data from it by using the instruments embedded in the Web-based model.

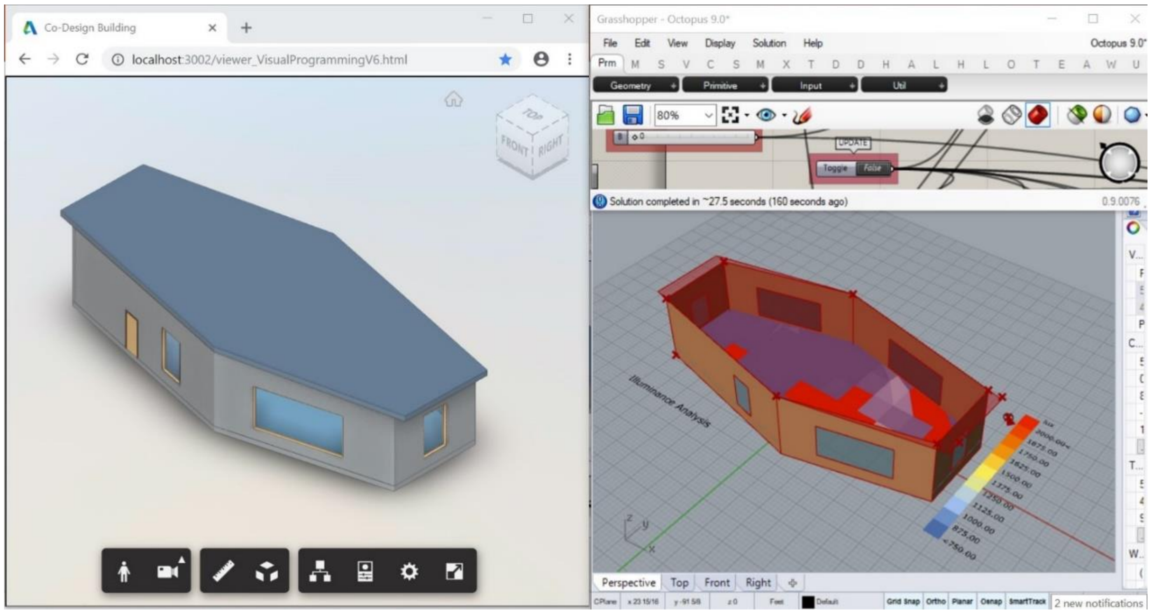

Co-Design Building
localhost:3002/viewer_VisualProgrammingV6.html
Grasshopper - Octopus 9.0*
File Edit View Display Solution Help
Geometry Primitive Input Util
80%
UPDATE
Toggle False
Solution completed in ~27.5 seconds (160 seconds ago)
Illuminance Analysis
Perspective Top Front Right
2 new notifications

Figure 31. Test 2.1: Optimum daylight and roof shape solution transferred from the local twin model (right) to the Web-based model (left)

4.2.2. Test 2.2 and Results

Test 2.2 was conducted to investigate the ways of how two Grasshopper users could collaborate through the Web-based model while optimizing the roof shape creating a shading and minimizing the energy usage. The goal of the energy optimization is to make the building more energy efficient and to use less energy as possible.

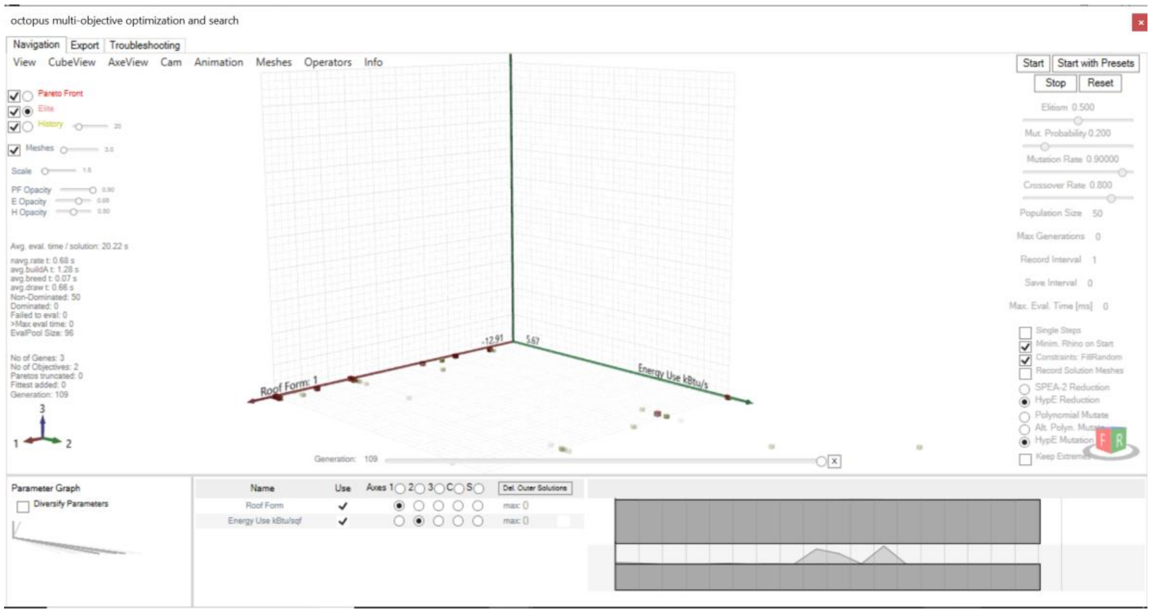
The process of setting up energy optimization is very similar to daylight optimization in terms of methodology and the tools that are used. The single surface twin model which was previously converted into a Honeybee model was used in the energy simulation. Ladybug was used to import the weather file and Honeybee was used for the energy simulation. The energy use results provided by Honeybee were used as one of the inputs for Octopus in the multi-objective optimization. Since Honeybee and Octopus were connected, energy simulation was run for each single design solution iteratively. Since the dominant loads were the heating loads for the project location, these loads were calculated for the month of December. However, any energy related optimization can be conducted with this framework, such as cooling loads, electrical loads, lighting loads, solar gain, airflow volume, and etc. Also, energy optimization calculations can be conducted for any time period, like for another month, or annually. Honeybee calculated the loads in kWh, so the total amount of energy usage was converted into kBtu and divided to the total floor area (sqft) in order to find the (dominant) heating EUI for December.

There were two objectives for optimization in Octopus: Minimization of the energy use and minimization of the roof shape while providing shading on the south. The two fitness functions are:

Fitness Function 1: min (u)

Fitness Function 2: min $(x \cdot y - c)$

where *u* is the (dominant) heating loads of December in kBtu/sqft, and *x*, *y* and *c* stand for Scale X, Scale Y, and the Coverage Index, respectively. The Coverage Index was multiplied by -1 in order to achieve a maximum value because Octopus minimizes the function. The Scale X, Scale Y, and Rotation Angle parameters of the roof constituted the genomes for Octopus. The computer used in the optimization process has Intel® Core™ i7-7500U CPU @ 2.70 GHz 2.90 GHz processor, 8.00 GB RAM installed memory, and Windows 10 operating system. The population was set to 50, and Octopus was stopped by the user after the solution set was converged. After109 generations of running GA which took almost 19 hours, the chance of achieving significant chances in the next generations got excessively low. The simulation was paused when necessary, but the file was saved then, and Octopus resumed optimization after the breaks. After 109 generations, 5 best solutions were generated as Pareto front. All solutions are displayed in Figure 32, while Pareto front is provided in Figure 33. The Pareto-optimal solutions for Roof Shape ranged between 1 and -12.91 while the energy solutions converged to 5.67 kBtu/sqft. The geometric outcomes of the solutions are shown in Figure 34. According to the geometric visualization, only solution #1 provided shading, only solutions #2 and #3 covered the room area. The solution #4 did not cover the room area and solution #5 did not even cover any of the required points.



[✓] Pareto Front
[✓] Elite
[✓] History 20
[✓] Meshes 3.0
Scale 1.5
PF Opacity 0.90
E Opacity 0.65
H Opacity 0.50

Avg. eval. time / solution: 20.22 s

navg.rate t: 0.68 s
avg.buildA t: 1.28 s
avg.breed t: 0.07 s
avg.draw t: 0.66 s
Non-Dominated: 50
Dominated: 0
Failed to eval: 0
>Max eval time: 0
EvalPool Size: 96

No of Genes: 3
No of Objectives: 2
Paretos truncated: 0
Fittest added: 0
Generation: 109

Generation: 109

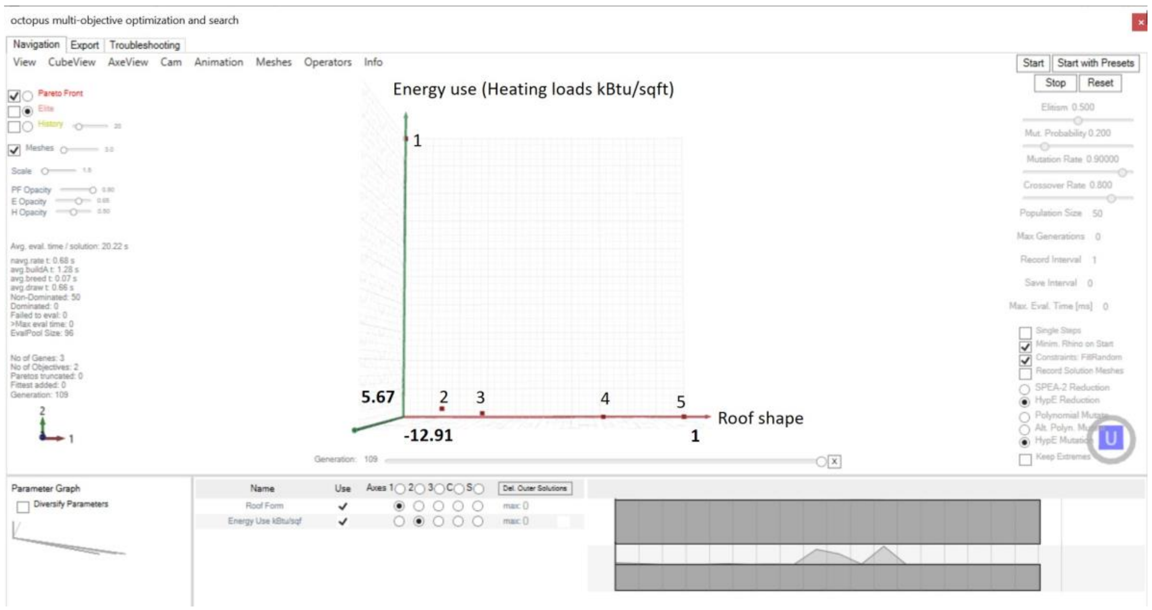
Elitism 0.500
Mut. Probability 0.200
Mutation Rate 0.90000
Crossover Rate 0.800
Population Size 50
Max Generations 0
Record Interval 1
Save Interval 0
Max. Eval. Time [ms] 0

[ ] Single Steps
[✓] Minim. Rhino on Start
[✓] Constraints: FillRandom
[ ] Record Solution Meshes
SPEA-2 Reduction [✓ HypE Reduction]
Polynomial Mutate / Alt. Polyn. Mutate / [✓ HypE Mutation]
[ ] Keep Extremes

[ ] Diversify Parameters

Name	Use	Axes 1	2	3	C	S	
Roof Form	✓	●	○	○	○	○	max: 0
Energy Use kBtu/sqf	✓	○	●	○	○	○	max: 0

Figure 32. Test 2.2: Search space for energy and roof shape optimization with all solutions after 109 generations



Avg. eval. time / solution: 20.22 s

navg.rate t: 0.68 s
avg.buildA t: 1.28 s
avg.breed t: 0.07 s
avg.draw t: 0.66 s
Non-Dominated: 50
Dominated: 0
Failed to eval: 0
>Max eval time: 0
EvalPool Size: 96

No of Genes: 3
No of Objectives: 2
Paretos truncated: 0
Fittest added: 0
Generation: 109

Generation: 109

Elitism 0.500
Mut. Probability 0.200
Mutation Rate 0.90000
Crossover Rate 0.800
Population Size 50
Max Generations 0
Record Interval 1
Save Interval 0
Max. Eval. Time [ms] 0

☐ Single Steps
☑ Minim. Rhino on Start
☑ Constraints: FillRandom
☐ Record Solution Meshes
○ SPEA-2 Reduction
● HypE Reduction
○ Polynomial Mutation
○ Alt. Polyn. Mutation
● HypE Mutation
☐ Keep Extremes

☐ Diversify Parameters

Name	Use	Axes 1	2	3	C	S	
Roof Form	✓	●	○	○	○	○	max: 0
Energy Use kBtu/sqf	✓	○	●	○	○	○	max: 0

Figure 33. Test 2.2: Search space and Pareto front for energy and roof shape optimization

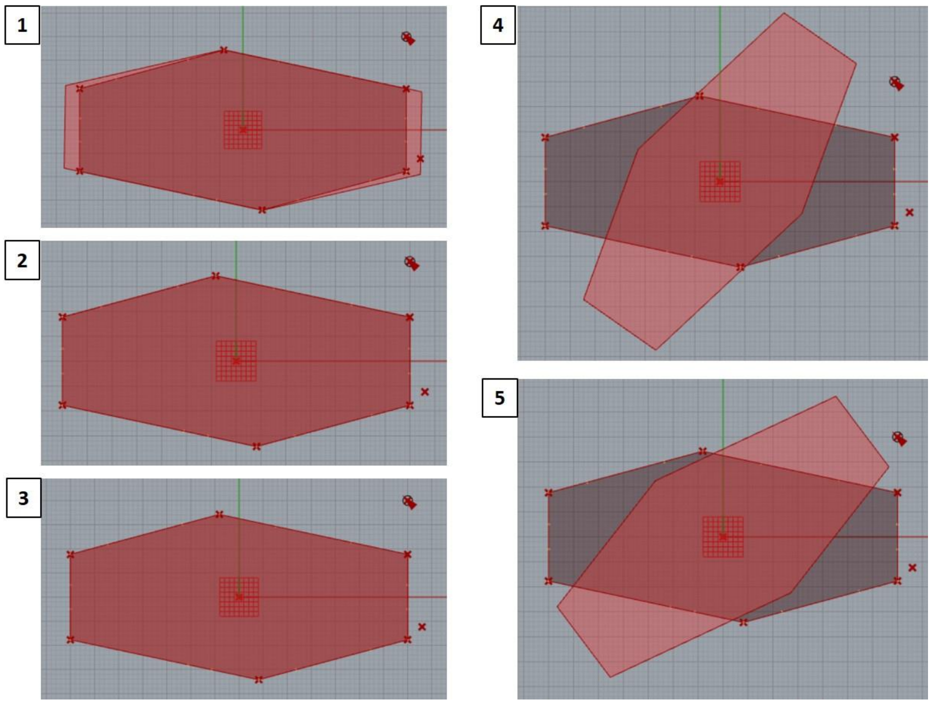


Figure 34. Test 2.2: Pareto front solutions visualized in the building floorplan view

The numeric outcomes of this analysis are listed in Table 7. Accordingly, the best energy use results were provided by the solutions #4 and the #5 with 5.671 kBtu/sqft. Minimum roof area was also produced by the solutions #4 and #5. Even though the initial orientation of the roof was defined at 0˚ angles in daylight optimization, for the energy optimization, the initial orientation of the roof was defined by 180˚. The solution #1 was chosen as the overall best result, and the corresponding data was transferred to the Web-based model. The Web-based models were updated and displayed the optimum result in both Grasshopper user's Web browser (Figure 35).

Table 7. Energy and Roof Shape optimization Pareto front data

Solutions	**Scale X**	**Scale Y**	**Rotation Angle**	**Coverage Index**	**Energy Use kBtu/sqft**	**x*y-z**
1	1.091	1.000	179	14	5.79	-12.91
2	1.001	1.001	180	12	5.68	-11.00
3	1.001	1.000	180	10	5.67	-9.00
4	1.000	1.000	235	4	5.67	-3
5	1.000	1.000	217	0	5.67	1

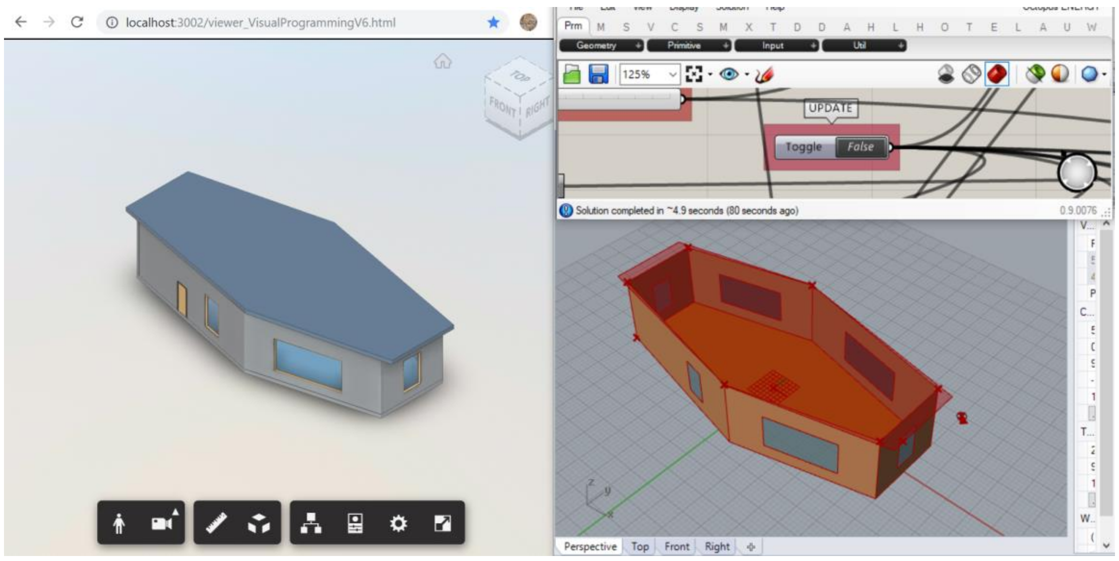

localhost:3002/viewer_VisualProgrammingV6.html
UPDATE
Toggle
False
Solution completed in ~4.9 seconds (80 seconds ago)
Perspective
Top
Front
Right

Figure 35. Test 2.2: Optimum energy and roof shape solution transferred from the local twin model (right) to the Web-based model (left)

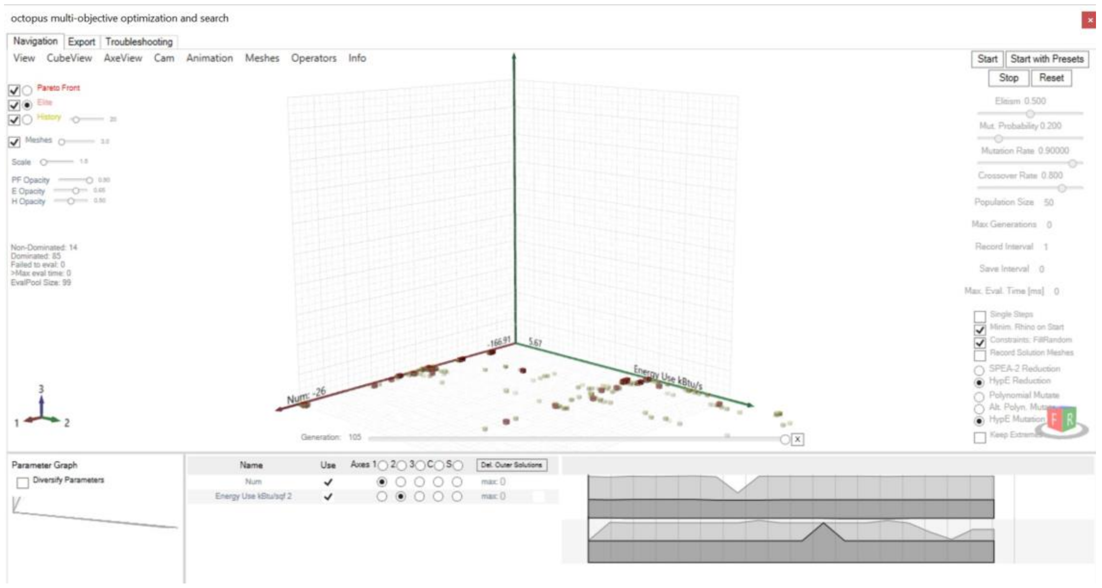
4.2.3. Test 2.3 and Results

Test 2.3 was conducted to examine how two Grasshopper users could collaborate through the Web-based model while executing multi-objective optimization with two objectives: (1) Energy use minimization for December (kBtu/sqft), and (2) preferred daylight (750-2000 lux) maximization and roof shape minimization while creating shading. The method, the tools, the simulation inputs, and the parameters (genomes) were the same as those of the previous tests. The only difference was the combination of the daylight objective with the roof shape objective in a weighted sum. The roof shape optimization was conducted through the use of the Coverage Index and Scale X and Scale Y variables of the roof. The Coverage Index referred the sum of the corner points and a selected point in the south at the roof level to be covered by the roof. If all the points were covered by the roof shape, 2 points were assigned for each point which yielded a sum of 14. The Coverage Index should be maximized in order to cover the room area. Therefore, by assigning the weight (10), the impact of the Coverage Index was increased, and the possibility to obtain solutions with covered roof were also increased. Since the second objective was achieving minimum roof shape, the Scale X and Y variables of the roof shape should be minimized. The two fitness functions were:

Fitness Function 1: min (t)

Fitness Function 2: min $(x \cdot y - u - 10 \cdot c)$

where *t* is the energy use in kBtu/sqft, u represents the number of sensors that received 750-2000 lux, and *x*, *y*, and *c* stand for Scale X, Scale Y, and the Coverage Index, respectively. The Coverage Index and the number of sensors output were both multiplied by -1 since Octopus optimizes through minimization of the function; so that, the roof coverage and the number of sensors within the ideal illumination range were maximized. The objective was to obtain the roof with shading and maximum sensors to receive the appropriate daylight. The population of the Genetic Algorithm using Octopus was set to 50 in this instance and the optimization was run on two different computers one after another (although one computer can also be sufficient to run the experiments). The optimization was started on a desktop with an Intel® Core™ i7-7820X CPU @ 3.60 GHz processor, 32.00 GB RAM installed memory, and Windows 10 operating system; and at the end of 10 hours, it produced 58 generations of solutions. The optimization was resumed on a laptop with an Intel® Core™ i7-7500U CPU @ 2.70 GHz processor, 8.00 GB RAM installed memory, and Windows 10 operating system. At the end of 13 hours, the optimization was ended by the user because the chance of achieving an improvement in the solution set in the next generations was excessively low. The number of produced generations increased to 105 (Figure 36). After 105 generations of running GA, 7 Pareto front (overall best) solutions were obtained; the solutions were visualized in Rhino for an evaluation of shape (Figure 37 - 38).



Meshes 3.0
Scale 1.5
PF Opacity 0.90
E Opacity 0.65
H Opacity 0.50

Non-Dominated: 14
Dominated: 85
Failed to eval: 0
>Max eval time: 0
EvalPool Size: 99

Elitism 0.500
Mut. Probability 0.200
Mutation Rate 0.90000
Crossover Rate 0.800
Population Size 50
Max Generations 0
Record Interval 1
Save Interval 0
Max. Eval. Time [ms] 0

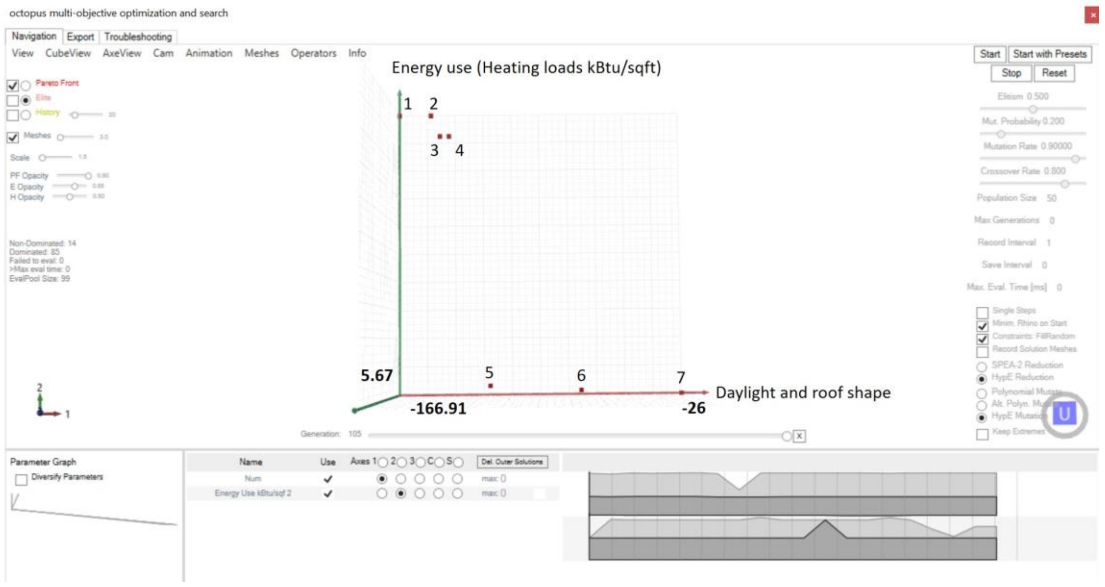
☐ Single Steps
☑ Minim. Rhino on Start
☑ Constraints: FillRandom
☐ Record Solution Meshes
○ SPEA-2 Reduction
● HypE Reduction
○ Polynomial Mutate
○ Alt. Polyn. Mutate
● HypE Mutation
☐ Keep Extremes

Generation: 105

☐ Diversify Parameters

Name	Use	Axes 1	2	3	C	S	
Num	✓	●	○	○	○	○	max: 0
Energy Use kBtu/sqf 2	✓	○	●	○	○	○	max: 0

Figure 36. Test 2.3: Search space for energy, daylight and roof shape optimization with all solutions after 105 generations



[✓] Pareto Front
[●] Elite
[ ] History 20
[✓] Meshes 3.0
Scale 1.5

Non-Dominated: 14
Dominated: 85
Failed to eval: 0
>Max eval time: 0
EvalPool Size: 99

Elitism 0.500
Mut. Probability 0.200
Mutation Rate 0.90000
Crossover Rate 0.800
Population Size 50
Max Generations 0
Record Interval 1
Save Interval 0
Max. Eval. Time [ms] 0

[ ] Single Steps
[✓] Minim. Rhino on Start
[✓] Constraints: FillRandom
[ ] Record Solution Meshes
SPEA-2 Reduction [✓ HypE Reduction]
Polynomial Mutation / Alt. Polyn. Mutation [✓ HypE Mutation]
[ ] Keep Extremes

Generation: 105

[ ] Diversify Parameters

Name	Use	Axes 1	2	3	C	S	
Num	✓	●	○	○	○	○	max: 0
Energy Use kBtu/sqf 2	✓	○	●	○	○	○	max: 0

Figure 37. Test 2.3: Search space and Pareto front for energy, daylight and roof shape optimization

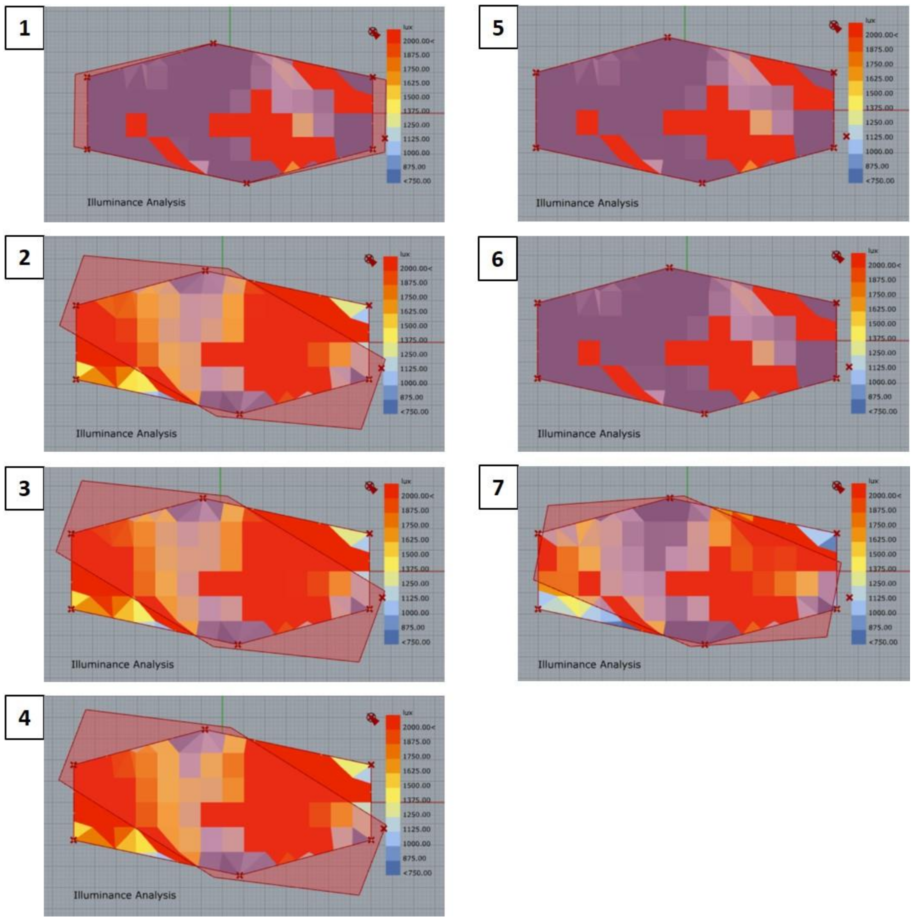


Figure 38. Test 2.3: Pareto front solutions visualized in the building floorplan view

The numeric data of the optimization solutions are presented in Table 8. Only the solution #1 provided the minimum roof area with shading. The solutions #5 and #6 covered the room area but did not cover the selected sample outdoor point (as a design requirement) for shading. The rest of the solutions #2, #3, #4, and #7 did not cover the room area; however, the solutions #2, #3, and #4 covered the point for the shading. As seen in Table 8, the best energy and daylight optimization results were produced by the solution #7, which has the worst room coverage. The 3 of the solutions (#1, #5, and #6) had the minimum preferred daylight. In all of the solutions, the findings show that the initial roof position was defined with an angle of 0º. The Grasshopper user chose the best result and transferred it to the Web-based model. In this instance, the solution #1 was selected as the overall best and viewed on the Web browser (Figure 39). The other Grasshopper user received the update and retrieved the data from the Web-based model.

Table 8. Energy, Daylight and Roof Shape optimization Pareto front data

Solutions	Scale X	Scale Y	Rotation Angle	Coverage Index	Energy Use kBtu/sqft	# of sensors
1	1.089	1.000	359	14	5.79	27
2	1.089	1.000	341	10	5.79	59
3	1.081	1.000	340	10	5.78	58
4	1.081	1.000	339	10	5.78	55
5	1.001	1.001	360	12	5.68	27
6	1.001	1.000	360	10	5.67	27
7	1.000	1.000	349	4	5.67	64

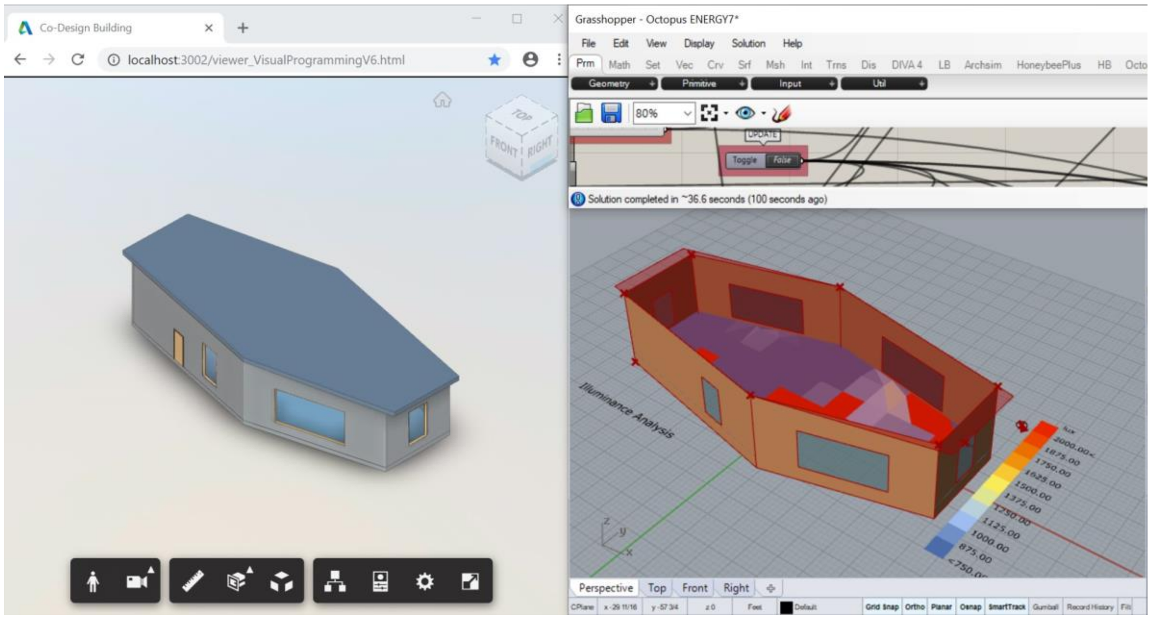

Co-Design Building
localhost:3002/viewer_VisualProgrammingV6.html
Grasshopper - Octopus ENERGY7*
File Edit View Display Solution Help
Solution completed in ~36.6 seconds (100 seconds ago)
Illuminance Analysis
2000.00<
1875.00
1750.00
1625.00
1500.00
1375.00
1250.00
1125.00
1000.00
875.00
<750.00
Perspective Top Front Right

Figure 39. Test 2.3: Optimum energy, daylight and roof shape solution transferred from the local twin model (right) to the Web-based model (left)

4.2.4. Discussion

In Case Study 2, the collaboration of two Grasshopper users in the process of BEP and roof shape optimization through the Web-based model was examined. The roof shape was parametrically explored and optimized to achieve the best possible shading in the south, the maximum preferred daylight (750-2000lux), and lowest energy use, using Pareto optimization with Genetic Algorithm. In order to optimize the roof shape for a minimum area while providing shading, Coverage Index was used. The Coverage Index referred to the inclusion of the room area through the corner points and a chosen point in the south at the roof level. When optimizing the roof shape, the Coverage Index should be maximized, the Scale X and Scale Y values in the parameters of the roof shape should be minimized.

Three tests were conducted as multi-objective optimization. In Test 2.1, there were two objectives: Maximum preferred daylight and minimum roof area with shading. In Test 2.2, there were two objectives: Minimum energy use and minimum roof area with shading. In Test 2.3, there were two objectives, but one of the objectives consisted of two objectives combined in a weighted sum. The objectives in Test 2.3 were minimum energy use as one objective, maximum preferred daylight and roof shape with shading in a weighted sum as the other objective. In this test, the impact of the roof coverage to the optimization objective function was increased by assigning a weight of 10 to the Coverage Index in the fitness function formula. In order to get all the solutions to cover the room area and the point, the weight of Coverage Index was increased.

In the beginning, the goal was to divide and distribute the optimization process between the users. If optimization could be divided and distributed, more complex projects could be optimized in a shorter time by using less computational power. Also, division and distribution of optimization would highly contribute to the collaboration in the process of design, performance simulation, and optimization. Therefore, the optimization was tried to be divided and distributed between two users working with one objective each. For this purpose, individual Galapagos evolutionary solvers were connected to each of the objectives. However, two Galapagos could not run at the same time; Galapagos can run only one instance at a time. As a result, the optimization cannot be divided and distributed into separate optimizations.

With the help of Octopus, the multi-objective optimization tool, the tests were conducted. However, two optimization processes still can’t be run at the same time. For the experiment, one user conducted the optimization process, and the other user got updates on the Web-based model, and retrieved data from the Web. In all three tests, at the end of the given number of generations, only one solution in the Pareto front provided the room area coverage with shading in each time. The best solution was evaluated by the user and transferred to the Web. The data recording, pausing, and resuming functions of Octopus were utilized conduct the optimization in different computers and reinstate the Pareto-optimal solutions. The local twin model of the Web-based model was utilized conduct simulation and optimization; the twin model also provided visual feedback in the assessment of the best solution before transferring the data to the Web-based model. If the Viewer prototypes for Dynamo and Grasshopper are modified to provide reciprocal communication between the Web-based model and the visual programming tools in the future, the multi-objective optimization process could be divided and distributed into optimizing separate objectives between the users through sharing parameters. For example, the Rotation angle, Scale X, and Scale Y parameters of the roof can be shared between two users while conducting roof shape with shading optimization by one user, and daylight optimization by another user.

5. FINDINGS AND DISCUSSION

In this research, the first objective was to search the ways to establish parametric relationships between the elements of the Web-based model, which had been lost during the conversion from BIM(Revit) to the Web-based model. However, prior to establishing these relationships, two different visual programming tools, Dynamo and Grasshopper had to be examined based on their capabilities to control the Web-based model. Two major interoperability issues were identified between Grasshopper and the Web-based model in the way they operate. The first major issue could be caused by using a transformation matrix. Although, both Rhino and Grasshopper's coordinate systems and the Web-based model's coordinate system follow the same right-hand rule for rotation, there is inconsistency between Grasshopper and the Web Viewer in rotation. The transformation matrix in Grasshopper is a world matrix which transforms a vector in a space where the positive Z-axis is perpendicular to the ground plane. The Z-axis in Grasshopper coincides to the with the positive Y-axis in the Web Viewer. This may cause inconsistency when assigning the rotation angles. Therefore, a coordinate transformation is needed to convert the matrix. For now, the angle variable in Grasshopper was flipped before sending the data from the local computer to the Web-based model to align the coordinate systems.

The second major interoperability issue is the reference point of transformations. In Grasshopper, the user has the control of creating a reference point for scale and rotate. On the other hand, by default all operations in the Web-based model reference the origin point, which is located in the center point of the model’s bounding box. Besides, a twin model was required to be built in Grasshopper in order to retrieve data for building parametric relationships. The twin model is the reproduction of the Web-based model in Rhino exported from a Revit file. The model elements in Rhino were carried to Grasshopper as Breps, and they together constituted the twin model. When the exported Revit file opened in Rhino, the origin point was at the same location with the Revit model. However, the center was changed during the conversion from Revit to Web-based model. So that, the twin model in Grasshopper and the Web-based model respond to Scale and Rotate differently because of the discrete reference points. In order to provide compatibility between the Web-based model and Grasshopper, either additional scripting is required or additional data flow in Grasshopper is required to be created. So that, additional data flow was created in Grasshopper. Since the Web-based model’s reference point is fixed and Grasshopper can modify its own reference points, the reference point of twin model in Grasshopper was relocated at the same location where the center of the Web-based model’s bounding box is, so that the origin point of Grasshopper (0, 0, 0) would match the model’s center of its bounding box. The relocation and flipping of the angle variable synchronized the transformations between Grasshopper and Web-based model. Since Dynamo is the visual programming tool for Revit and the Web-based model was converted from a Revit model, and no built-in transformation matrix is available in Dynamo yet, no modification was required in Dynamo in order to synchronize the angle or the reference point.

After the elimination of the two interoperability issues between the Grasshopper twin model and the Web-based model, the elements of both models began to operate the same way. However, an additional data flow was required to be created in Grasshopper in order to make the result of rotation or scale of the twin model to be identical with the Web-based model. The twin model provided visualization and data for building this data flow. For instance, when a wall was rotated, the rotation was around the coordinate system's origin point. Therefore, following the rotation, the wall of the Grasshopper twin model was not located at its initial position, but was moved out. In order to finalize the position of the wall as if the wall had rotated around the Z-axis of its own corner point, the wall needed to be translated back after the rotation. With the help of the twin model method, the distance between the location of the rotated wall and the expected location could be calculated. Creating a copy of the original geometry after each transformation served in calculating the displacement between the initial and the last positions. By using the twin model method and the data provided by this method, the elements in Grasshopper were modified to respond the same as the elements of the Web-based model respond in transformations, such as scale and rotate. Similarly, the twin model could provide the data to build the parametric relationships between the elements of both the twin model and the Web-based model. For instance, if a wall was rotated, the other wall was required to be scaled to meet that wall. In order to build that parametric relationship, first, the final location of the rotated wall had to be calculated. Second, the data from the rotated wall's final position had to be extracted in order to calculate the scale factor of the second wall to meet the first wall. However, the elements were scaled according to the origin, which resulted in them to be translated. In order to place the elements into where they were expected to be, and calculate the scale factors, the required data was supplied by the twin model elements.

After setting up the parametric relationships for the elements of the Web-based model through the local twin model, the model was tested in the Case Study 1 on the basic transformations: Scale, Translate, and Rotate. The framework, which included the Web-based model, Grasshopper, and Dynamo files successfully demonstrate the interoperable collaboration of the tools on the parametric model for Scale and Translate. Each user was able to control the Web-based model and the Web-based model was parametrically able to respond to the transformation requests. The results of the parametric changes were updated on each user's Web browser. The Web-based model also contained the data, which could be retrieved by any user. The users were able to collaborate by retrieving data from the Web-based model.

For rotation, one Grasshopper user with the Web-based model tested two diagonal walls. Although the parametric relationship was accommodated between the two diagonal walls, it required more time and scripting to build the same relationship between the walls and roof and floor because the parametric changes could only be applied uniformly now. For instance, if two walls were rotated, the related sides of the roof or floor could not adapt their geometry to this non-uniform change. In addition, the rotated diagonal walls did not meet precisely at joints. Since the parametric relationship between the elements was lost when the Revit model was converted to the Web-based model, the elements could not adapt their geometries at joints after transformations, even the parametric relationship was built by the user. For instance, the edges of the diagonal wall were determined in Revit prior to converting the model into a Web-based model. After the conversion, additional scripting was required to manipulate the edges of the walls. In order to overcome both limitations, the editing of geometry vertices should be provided which requires more development of the Dynamo and Grasshopper Viewer plugins. The issue of the walls not meeting precisely had also been observed in translation. The elements of the Web-based model could be scaled, translated, or rotated, however; they did not adapt their geometries at the joints. The edges of the elements did not respond to the parametric changes. As a result, the wall joints were not clean.

In the Case Study 2, the collaboration between two Grasshopper users through the Web-based model in the process of building performance optimization was examined. The daylight, energy, and integrated daylight and energy optimization was tested. In each optimization, one other objective was to obtain the minimum roof area that provided shading on south. The Dynamo user was not included in this process since the tools for BEP simulation in Dynamo was still limited. In collaboration between two Grasshopper users through the Web-based model, one user actively optimized the Web-based model according to the objectives: maximum daylight with minimum roof area with shading or minimum energy use and the same roof objective. The other user observed the updates on the Web-based model and retrieved data from the model. Previously, the process was experimented by using Galapagos evolutionary solver – the single objective optimization tool in Grasshopper - by assigning one Galapagos for each user and one objective, i.e. defining one user with objective 1 as maximum preferred daylight using one Galapagos node; defining another user with objective 2 as minimum roof shape with shading using another Galapagos node. However, two Galapagos solvers could not be run at the same time. Therefore, the Octopus multi-objective optimization tool was utilized by defining two objectives to be optimized at the same time in the Test 2.1 of Case Study 2. At the end of the optimization process, a set of Pareto-optimal solutions were obtained. In Test 2.1, optimization with maximum preferred daylight and minimum roof with shading, only one of the solutions provide cover for the roof and shading. In Test 2.2, optimization with minimum energy use and minimum roof with shading, despite more than one solution was generated that covered the room area, only one solution provided shading. In Test 2.3, the daylight objective was combined with roof shape with the shading objective, where the other objective was the minimum energy use. In order to ensure that both the room area and the chosen point for shading were covered, a weight factor (10) was assigned for the Coverage Index in the fitness function. However, among the Pareto front, only one solution provided the room cover and the shading, which also had the one of the highest energy use. It is expected by adjusting the parameters of the fitness functions, for example, the weight factor for the Coverage Index, more desired design options may appear in the Pareto front.

In all optimization processes, the user ended the optimization process after observing that there was an extremely low chance of retrieving improvement in the next generations. Afterwards, the user decided the best solution among the Pareto front. After choosing the best solution, the data was transferred to the Web-based model and both users got the updates on their Web browsers. While one user optimized the Web-based model, the other user could observe the updates on the Web-based model and could retrieve data from this model. The framework enabled two Grasshopper users to collaborate on daylight, energy, and roof shape optimization through Web-based model as sample optimization experiments. The usage of the framework requires the knowledge of the mentioned tools, Rhino, Grasshopper, Revit, Dynamo, Ladybug, Honeybee, and Octopus. Moreover, the users should have the understanding of building performance, basic terms, simulation metrics, and how to conduct simulations and optimization in order to collaborate in the process of performance optimization.

6. CONCLUSIONS

The research has demonstrated the advancement of a Web-based parametric modeling method and its prototypes, utilizing the full-fledged visual programming tools (Grasshopper and Dynamo), which enable parametric modeling, performance simulation, and optimization, while the Web-based model enables collaborative design exploration. There are many technical challenges, especially interoperability problems, found and resolved for the project through the carefully designed experiments. However, due to the high complexity of the problem, limitations are found to be investigated in future work.

The parametric Web-based modeling framework was experimented on two case studies and three tests in each case study. In the Case Study 1, answers to the two questions were investigated:

1. How could the parametric relationships between the elements of the Web-based model be built within the proposed framework?
2. How to implement the collaboration through a Web-based model while using Grasshopper and Dynamo and get real-time updates?

In order to build parametric relationships, data flow was required to be created both in Grasshopper and Dynamo. The transformations, Scale, Translate, and Rotate were tested in Test 1.1, Test 1.2, and Test 1.3, respectively. The findings showed that the framework worked on Scale and Translate, partially worked on Rotate. Two major interoperability issues were identified between Web-based model and Grasshopper: The reference point of transformations and the directions of rotation. The reference point difference was overcome by using additional data flow in Grasshopper. The use of transformation matrix could cause a difference on rotation between Web-based model and Grasshopper, which was handled by flipping the rotation angle. Since the data transfer was from visual programming tools to the Web-based model, the framework enabled the collaboration of one active and one passive user. While one user modified the Web-based model, the other user observed the updates and retrieved data with the help of Web-based model's instruments.

In the Case Study 2, an answer to the question "Is it possible for two Grasshopper users to collaborate on a Web-based model while working on building performance optimization?" was investigated. Three tests, Test 2.1, Test 2.2, and Test 2.3, were conducted by using Honeybee for performance simulation, Ladybug for importing weather file to Grasshopper and visualization of the simulations, and Octopus for multi-objective optimization. In Test 2.1, maximum preferred daylight and minimum roof shape with shading were the two objectives. In Test 2.2, minimum energy use and minimum roof shape with shading were the two objectives. In Test 2.3, weighted sum of maximum preferred daylight and minimum roof shape with shading was the one objective, minimum energy use was the other objective. The optimization was first experimented by using Galapagos; however, two Galapagos did not work at the same time. Since the prototype transferred data was from visual programming tools to the Web-based model, the framework only allowed the collaboration of one active and one passive user. Therefore, the optimization could not be divided and distributed between the users. The framework could be used as follows: One user optimized the Web-based model on the local computer and decided the best result by visualizing the Pareto front and evaluating the numeric data output of optimization. The best result was transferred from Grasshopper to the Web-based model. The other user observed the updates and retrieved data with the help of Web-based model's instruments.

6.1. Limitations

There are two types of limitations: The limitations of the thesis and the limitations of the framework. However, the limitations of the framework are also the outcomes of the research, since one of the goals of this thesis is to identify the ways of collaboration, building parametric relationships, and conducting optimizations which includes the identification of the limitations of the framework as well.

One of the outcomes and the limitations of the framework is the number sliders of Grasshopper and Dynamo not allowing to be updated by external data, data from the Web-based model in this instance. Data transfer being only one-way, from visual programming tools to the Web-based model, is a limitation in the collaboration of parametric modeling and optimization. Receiving the updates only on the Web-based model requires the manual data input by the user. Without the automatization of the data transfer bidirectionally between the Web-based model and Grasshopper/Dynamo, the collaboration of the users is limited as one active user and one passive user. The optimization process cannot be divided and distributed into different users and the parameters cannot be shared without reciprocal data transfer. Also, Galapagos running only one instance at a time limits this distributed optimization method and the use of Galapagos. Another limitation is the lack of control of the capability to directly change the geometry vertices of the Web-based model. This limitation creates imprecise joint problems, especially in the rotation transformations. In addition, the limited control on scale transformations results only in uniformly scaled geometries which do not always generate the intended forms. The control of the Web-based model by using Dynamo is limited on rotation and this prevents the collaboration of Grasshopper and Dynamo through the Web-based model on rotation. The need for the local twin model in Grasshopper in order to retrieve data for building parametric relationships and conduct optimization, instead of direct parametric modeling, simulation, and optimization with the Web based model, is a limitation.

One of the limitations of the thesis is the simplification of the case studies due to the time limit. For example, the current parametric relations in the proposed study are created based on the transformation of one single element, one of the narrow walls. Other elements are adjusted parametrically in their size, orientation, and scale in order to match that narrow wall's transformations. This is a limitation of the case studies. However, the simple prototype model can demonstrate the potential of the method for supporting Web-based parametric modeling and the problems of tools used. Second limitation of the thesis is the usage of the framework being based on simulated user experiments.

6.2. Future Work

Three different suggestions are proposed to improve the framework. The first suggestion is provide editing the geometry vertices of the Web-based model through the visual programming tools in order to achieve precise joints between the elements of the model after transformations. Editing geometry vertices individually enables the users to modify the geometry of the elements efficiently and flexibly for more comprehensive parametric modeling. With the control of the vertices, the users could transform the geometries in a non-uniform way.

The second suggestion is the automation of the data transfer from the Web-based model to the visual programming tools. By automation, the building parameter values would transfer reciprocally between the Web-based model and the visual programming tools. Then, after any change, both the Web-based model and the local twin parametric models would be updated automatically. The reciprocal data transfer will help divide and distribute the optimization process into two or more users for collaboration. When optimization is divided and distributed, the amount of time and the computational power required for the process would decrease. For instance, in large and complex projects, designers deal with a large number of design decisions which increases the number of inputs and the size of the search space. However, if optimization is divided and distributed, users could optimize simultaneously; and this way, the process would be more efficient and effective, and requires less computational power for each user when compared to a traditional optimization with one user. If the automation of reciprocal data exchange is provided, then the optimization could be collaboratively conducted by sharing parameters. Optimizing simultaneously by sharing parameters would result in the interaction between the genomes of each optimization. The best performing genomes are selected at the end of each generations, and the value of the shared genomes in Genetic Algorithm would be updated after each run of an optimization. When the optimization is divided and distributed, the selection of the best genomes in different users' optimization will affect each other's optimization process. This interaction is expected to generate optimum solutions for a Web-based model that satisfy two or more different optimization objectives. With the division and distribution of the optimization, specialists could collaborate on the same project and achieve a set of design solutions which are optimum for all parties, e.g. a lighting specialist and an energy specialist could collaborate on a shared model while optimizing.

The third suggestion for the improvement of the framework is the elimination of the need for the twin models in local visual programming environments, and enabling direct modeling, simulation, and optimization on the Web.

As a future work, integration of Dynamo would be improved to be used in the performance simulation process. In the future framework, visual programming would be employed inside the Viewer which would contribute to design from the scratch on the Web browser. Also adopting simulation GUIs and optimization plugins into the Web-based model would make the framework a collaborative parametric modeling, design, and simulation tool which would be compatible with many prevalent software. Collaboration between structural, mechanical and electrical engineers and architects by integrating detailed models of engineers would be another goal of future work.

Moreover, automatic creation of architectural 2D drawings and data sheets according to the 3D Web-based model would provide the BIM features to the Web-based model. In addition, the control of permission levels (view, edit, share) of users and the recording of logging and editing history of users would enhance the collaboration experience.

REFERENCES

Akashi, Y., Akizuki, Y., Cobham, M., Itoh, N., Miller, N.J., Schlangen, L. J. M., & van den Broek-Cools, J.H.F. (2017). *CIE 227:2017 Lighting for Older People and People with Visual Impairment in Buildings*. 10.25039/TR.227.2017.

Azhar, S. (2011). Building Information Modeling (BIM): Trends, Benefits, Risks and Challenges for the AEC Industry. *ASCE Journal of Leadership and Management in Engineering, 11 (3),* 241-252, http://dx.doi.org/10.1061/(ASCE)LM.1943-5630.0000127

Bakshi, A. (2017). 8 Visual Programming and Modeling Tools You Didn't Know About. Retrieved November 14, 2018, from https://www.linkedin.com/pulse/8-visual-programming-modeling-tools-you-didnt-know-arpan-bakshi

Bhowes, TT. (2015). Spectacles. Retrieved May 24, 2019, https://www.food4rhino.com/app/spectacles

BIMscript®. (n.d.). Retrieved May 24, 2019, from https://info.bimobject.com/solutions/bimscript

Buyya, R., Yeo C.S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, Volume 25, Issue 6, pp. 599-616. https://doi.org/10.1016/j.future.2008.12.001.

Caldas, L.G. & Norford, L.K. (2002). A design optimization tool based on a genetic algorithm. *Automation in construction*, 11, no. 2: 173-184.

Choi, J., Beltran, L. O., & Kim, H. (2012). Impacts of indoor daylight environments on patient average length of stay (ALOS) in a healthcare facility. *Building and Environment, 50,* 65-75.

Cichocka, J. M., Browne, W. N., & Rodriguez, E. (2017). Optimization in the Architectural Practice - An International Survey. In *Proceedings of the 22nd CAADRIA Conference, Xi'an Jiaotong-Liverpool University, Suzhou, China,* 5-8 April 2017, pp. 387-396

Cominetti, M. (2017). Dynamo Unchained 1: Learn how to develop Zero Touch Nodes in C#. Retrieved May 24, 2019, from http://teocomi.com/dynamo-unchained-1-learn-how-to-develop-zero-touch-nodes-in-csharp/

Crawley, D. B., Hand, J. W., Kummert, M., & Griffith, B. T. (2005). Contrasting the Capabilities of Building Energy Performance Simulation Programs. In *Proceedings of Building Simulation*, *2005*, Montreal, Quebec, Canada, IBPSA, 231-238 pp.

Daher, E. (2017). The Next Generation of AEC Design. Retrieved May 24, 2019, from https://www.linkedin.com/pulse/next-generation-aec-design-eli-daher?trk=pulse_spock-articles

Davis, D. (2013). History of parametric. Retrieved May 24, 2019, from http://www. danieldavis.com/a-history-of-parametric/

DiLaura, D. L., Houser, K. W., Mistrick, R. G., & Steffy, G. R. (2011). *The lighting handbook: Reference and application* (10th ed.). New York: Illuminating Engineering Society of North America.

Eastman, C, Teicholz, P, Sacks, R, & Liston, K. (2011). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Hoboken, New Jersey: John Wiley &Sons, Inc.

Edwards, L. & Torcellini, P. (2002). *A literature review of the effects of natural light on building occupants.* (No. NREL/TP-550-30769). Golden, CO: National Renewable Energy Laboratory.

EnergyPlus. (n.d.). Retrieved May 24, 2019, from https://energyplus.net/

Frei, R. (2013). Two in One House / Clavienrossier Architectes. Retrieved May 24, 2019, from https://www.archdaily.com/373375/two-in-one-house-clavienrossier-architectes

Fritz, R. & McNeil, A. (2019, Jan 04). About Radiance. Retrieved May 24, 2019, from https://radiance-online.org/about

Fonseca, C. M. & Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1), 1-16.

Gero, J. S. (1975). Architectural optimization-a review. *Engineering Optimization*, 1(3): 189-199.

Heschong Mahone Group, Inc. (2003a). Windows and Offices: A Study of Office Worker Performance and the Indoor Environment. Retrieved May 24, 2019, from http://h-m-g.com/downloads/Daylighting/A-9_Windows_Offices_2.6.10.pdf

Heschong Mahone Group, Inc. (2003b). Windows and Classrooms: A Study of Student Performance and the Indoor Environment. Retrieved May 24, 2019, from http://h-m-g.com/downloads/Daylighting/A-7_Windows_Classrooms_2.4.10.pdf

Honeybee. (n.d.). Retrieved May 24, 2019, from https://www.ladybug.tools/honey bee.html

Hudson, R. (2010). *Strategies for parametric design in architecture: An application of practice led research.* (Ph.D. Dissertation). Bath, UK: University of Bath. Retrieved May 24, 2019, from ProQuest Dissertations & Theses Global (ProQuest document ID 1314567001)

Is There a Place for SkecthUp in the BIM Process? (2017, June 22). [Blog post]. Retrieved May 24, 2019, from https://www.excitech.co.uk/Insights/Blog/June-2017/Is-There-A-Place-For-SketchUp-In-The-BIM-Process

Janssen, P., Li, R., & Mohanty, A. (2016). Möbius: A Parametric Modeller for the Web. In Living systems and micro-utopias: Towards continuous designing. In *Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA* (pp. 157-166).

Mace, R. L., Hardie, G. J., & Place, J. P. (1996). *Accessible environments: Toward universal design*. Raleigh: North Carolina State University.

Maile, T., Fischer, M., & Bazjanac, V. (2007). Building energy performance simulation tools-a life-cycle and interoperable perspective. *Center for Integrated Facility Engineering (CIFE) Working Paper*, *107*, 1-49.

Michalek, J., Choudhary, R., & Papalambros, P. (2002). Architectural layout design optimization. *Engineering Optimization*, *34*(5), 461–484.

Möbius Parametric Modeller (n.d.). Retrieved May 24, 2019, from http://design-automation.net/software/mobius/mobius_parametric.html

Nabil, A. & Mardaljevic, J. (2005). Useful daylight illuminance: a new paradigm for assessing daylight in buildings. *Light Res Technol*, 37(1):41–59

Nabil, A. & Mardaljevic, J. (2006). Useful daylight illuminances: A replacement for daylight factors. *Energy and Buildings*, *38(7),* 905–913. doi: 10.1016/j.enbuild. 2006.03.013.

NAHB National Research Center. (1996). *Residential Remodeling and Universal Design Making Homes More Comfortable and Accessible.* Upper Marlboro, MD: NAHB Research Center.

National Institute of Building Sciences. (2008). National Building Information Model Standard. Retrieved May 24, 2019, from https://www.nibs.org/

Rahmani Asl, M., Bergin, M., Menter, A., & Yan, W. (2014). BIM-based Parametric Building Energy Performance Multi- Objective Optimization. In *Proceedings of the Conference of Education and Research in Computer Aided Architectural Design in Europe (eCAADe),* Newcastle Upon Tyne, UK.

Rahmani Asl, M., Stoupine, A., Zarrinmehr, S., & Yan, W. (2015a). Optimo: A BIM-based Multi-Objective Optimization Tool Utilizing Visual Programing for High Performance Building Design. *In Proceedings of the Conference of Education and Research in Computer Aided Architectural Design in Europe (eCAADe)*. Vienna, Austria.

Rahmani Asl, M., Zarrinmehr, S., Bergin, M., & Yan, W. (2015b). BPOpt: A Framework for BIM-based Performance Optimization. *Energy and Buildings*, 108, pp. 401-412

Reference: US National Average Site EUI and LPD. (n.d.) Retrieved from May 24, 2019, https://2030ddx.aia.org/helps/National Avg EUI

Reinhart, C., Dogan T., Geisinger J, & Saratsis M. (n.d.). ArchSim Simulation Game Instructions. Retrieved from May 24, 2019, http://Web.mit.edu/sustainabledesignlab/projects/SimulationGameRevisited/SimulationGameInstructions.pdf

Roudsari, M. S. & Pak, M. (2013). Ladybug: A Parametric Environmental Plugin for Grasshopper to Help Designers Create an Environmentally-Conscious Design. In *Proceedings of 13th International Conference of the International-Building-Performance-Simulation-Association (IBPSA):* Aug 25-28, 2013, [Chambéry, France], 3128-3135.

Rutten, D. (2010, Sept 25). Evolutionary Principles applied to Problem Solving. [Blog post]. Retrieved May 24, 2019, from https://www.grasshopper3d.com/profiles /blogs/evolutionary-principles

Safe, M., Carballido, J., Ponzoni, I., & Brignole, N. (2004*). On Stopping Criteria for Genetic Algorithms.* Lecture Notes in Computer Science, 405–413. doi:10.1007/978-3-540-28645-5_41

Schumacher, P. (2008). Parametricism as Style: Parametricist Manifesto. Retrieved from https://www.patrikschumacher.com/Texts/Parametricism%20as%20Style.htm

Schade, J., Olofsson, T., & Schreyer, M. (2011). Decision-making in a model-based design process. *Construction management and Economics*, 29, no. 4: 371-382.

ShapeDiver (2018). Retrieved May 24, 2019, from https://www.food4rhino.com/app/ shapediver

Stover, C. & Weisstein, E. W. (n.d.). *Parametric Equations*. From MathWorld - A Wolfram Web Resource. Retrieved May 24, 2019, from http://mathworld.wolfram.com /ParametricEquations.html

U.S. Department of Energy (2011). *2011 Buildings Energy Data Book*. Maryland: D&R International. Retrieved May 24, 2019, from https://ieer.org/resource/energy-issues/2011-buildings-energy-data-book/

U.S. Department of Energy (August 2015). *Volume 7.3. Guide to Determining Climate Regions*. County Pacific Northwest National Laboratory.

U.S. Energy Use Intensity by Property Type (August 2018). Retrieved May 24, 2019, from https://portfoliomanager.energystar.gov/pdf/reference/US%20National%20Median%20Table.pdf

USGBC. (n.d.). Daylight. *LEED BD+C: Healthcare | v4 - LEED v4*. Retrieved May 24, 2019, from https://www.usgbc.org/credits/healthcare/v4-draft/eqc-0

Vierlinger, R. & Bollinger, K. (2014). Accommodating Change in Parametric Design. In *ACADIA 14: Design Agency, Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture*, edited by David Gerber, Alvin Huang, and Jose Sanchez, 609–618. Los Angeles: ACADIA.

Ward, G. J. (1994). The RADIANCE Lighting Simulation and Rendering System. *SIGGRAPH 94 conference proceedings: July 24 - 29, 1994, [Orlando, Florida]*. New York, NY: ACM. [doi>10.1145/192161.192286]

WebGL: 2D and 3D graphics for the Web. (n.d.). Retrieved May 24, 2019, from https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

Wong, L. (2017). A review of daylighting design and implementation in buildings. *Renewable and Sustainable Energy Reviews, 74,* 959-968.

Woodbury, R. F. (2010). *Elements of parametric design*. London; New York: Routledge.

Yan, W. (2014). Chapter 5 - Parametric BIM SIM: Integrating Parametric Modeling, BIM, and Simulation for Architectural Design. In Kensek, K & Noble, D (Eds.), *Building Information Modeling: BIM in Current and Future Practice, (*p. 59–77). Hoboken, New Jersey: Wiley.

Yan, W. (2017). WP-BIM: Web-based Parametric BIM for Collaborative Design and Optimization. In *Proceedings of the 35th eCAADe Conference*, Rome, Italy, 20-22, September.