PARAMETRIC FORM MAKING WITH MODULAR UNITS

by

Rachel Lutes

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Architecture and Information Technology

Charlotte

2020

Approved by:

Eric Sauda

David Thaddeus

David Wilson

Jefferson Ellinger

ABSTRACT

RACHEL LUTES. Parametric form making with modular units. (Under the direction of ERIC SAUDA)

Modular units are a bedrock of modern construction. Finding ways to design new forms and be creative with modular units is a challenge for architects and designers. Parametric design allows designers to create a wide range of configurations, but tends to result in designs that require lots of different parts, rather than using modular units that are common in construction. Parametric design requires time and experimentation with coding and visual coding systems. This thesis looks at how to create an intuitive user interface to allow designers to easily create complex designs with common modular units, using bricks as a common and simple modular unit. Bricks also have an extensive history of use to create creative forms throughout history, making them a good choice for this project.

DEDICATION

To my parents, for always supporting my academic goals even when they have no clue what I am doing. To my sister, for keeping me sane and productive during the last stretch of finishing this thesis during the Covid-19 pandemic.

ACKNOWLEDGEMENTS

I would like to acknowledge Eric Sauda's invaluable advising and constant assistance, as well as David Thaddeus, David Wilson, and Jefferson Ellinger who lent me their experience and knowledge. Without them, this thesis would not have existed.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

I first became interested in complex brick forms during an undergraduate architecture studio. The project was a library in a neighborhood of Charlotte, NC that had lots of brick buildings. I wanted to create a building that blended in with the neighborhood, but had plenty of natural light. However, since it was a library, I knew that I needed to protect the books from direct sunlight as much as possible. This lead me to exploring brick screen patterns. Since I had just learned Grasshopper for the first time, I dove into the challenge of how to parametrically stack and manipulate bricks. At the time I did not have the knowledge or time to do what I wanted, so I set the idea aside until grad school and this thesis.

CHAPTER 1: INTRODUCTION

This project is an important tool to assist designers to create and analyze a wide range of wall structures built out of modular units without coding or determining the logic for each different design or idea.

This project has two main research questions which are listed below.

1. Can I create an open source, simple to use, well featured, and flexible program to assist architectural designers in the creation and understanding of complex brick structures?

2. How can I use parametric computation to create a graphical user interface (GUI) to allow architectural designers without programming knowledge to design and structurally analyze a complex geometric form made up of individual modules such as a brick wall?

The first question focuses on the requirements- the functionality and features needed to create a complex parametric brick wall with a wide range of potential forms.

The second question focuses on the design and effectiveness of a user interface (UI) to allow users to interact easily with the functional program created in the first question.

Together, these two questions allow this project to enable the user to easily create a variety of forms built with individual modules and analyze the forms for structural stability at different levels.

CHAPTER 2: TERMINOLOGY

Modular units can be used to create complex forms and structures if the designer has the knowledge and experience to ensure that the structure is stable. A tool that allows a designer to manipulate set variables while ensuring that the resultant structure is stable would allow designers to easily design complex forms without needing to become experts in the structural and geometrical rules of the module. Brick is an example of this sort of module that has simple enough geometry and construction rules to make it a good starting point for such a tool.

This project aims to create an interface to allow architects to design custom modular structures without needing to learn to code.

The main goal of the interface is to allow the user to create complex custom structures using the modular unit of a standard brick. The basic transformations, manipulations, and variables that will be included in the program to allow for design and variation are brick dimensions, rotation, translation, removal, wall form, and brick gap.

2.1 Brick Dimensions

The dimensions of the brick itself. There are a variety of different brick sizes commercially available, and the most common varies depending on the location and application. To make the most flexible program, the user will have the option to choose from a set of preset standard sizes, or will be able to change the dimensions to suit their design goals.

2.2 Rotation

The rotation around a vertical axis through the centroid of each individual brick (Figure 2.1). This creates gaps in the wall without removing bricks. These gaps can be utilized to create a shade screen with brick. Each brick can be rotated the same as all the others to create a uniform pattern, or individual bricks or rows can be rotated separately to create a design. The downside of this rotation is that the overlap connecting the brick to the others around it is decreased which affects the structural stability of the wall as a whole.

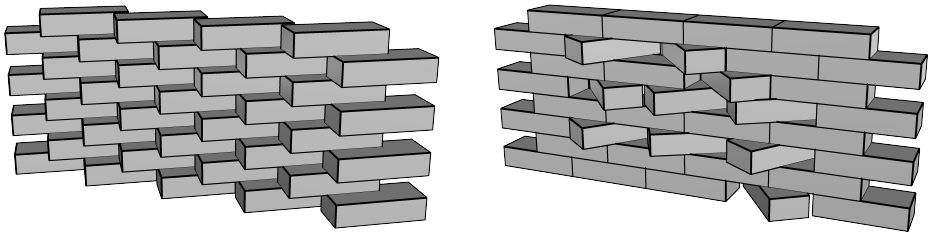

Figure 2.1: Uniform and varied rotation

2.3 Translation

The movement of a brick forward and backward in the thickness of the wall (Figure 2.2). This creates a subtle pattern with the shadows created by the light hitting the wall. The effects of the decreased surface area connecting the brick to others is minimal with translations, though should still be considered, as the larger the translation, the more the effects on stability.

2.4 Removal

The strategic removal of specific bricks (Figure 2.3). This creates a lighter, more transparent wall at the cost of the structural stability of each individual brick and the whole wall. This can be used to create a more transparent structure than can be achieved with rotations, but the bricks to be removed must be carefully chosen and located to ensure that all the other bricks remain properly supported.

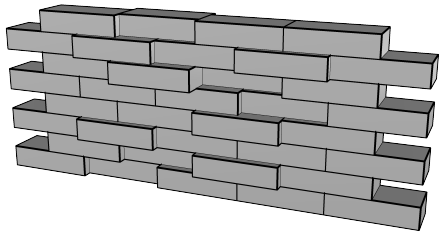

Figure 2.2: Translated bricks

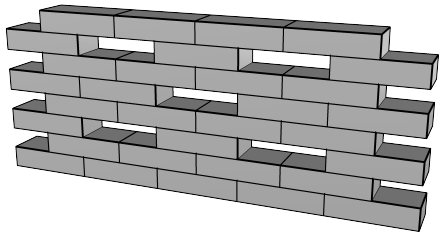

Figure 2.3: Removed bricks

2.5 Wall Form

The base form that the wall will attempt to create (Figure 2.4). This is basically the surface of the wall face. The examples up to this point have all used a basic vertical rectangular form. Figure 2.4 is an example of how a curved form like the one shown would be followed by the bricks being used to create it. More complex and curved forms can add structural stability to the overall wall.

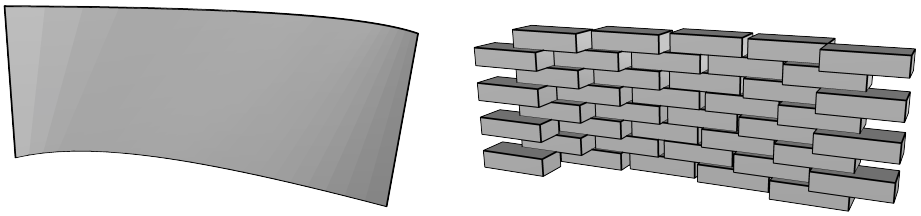

Figure 2.4: Wall surface and bricks following the surface

2.6 Brick Gap

How much space is left between bricks (Figure 2.5). Traditionally, this space would be filled with mortar, though in this project the gap will be left empty as all structures will be gravity/dry fit. This determines the horizontal spacing of the bricks along the wall and can be used to create a shade screen or a lighter structure. The surface area between bricks is decreased, but the loss is minimal.

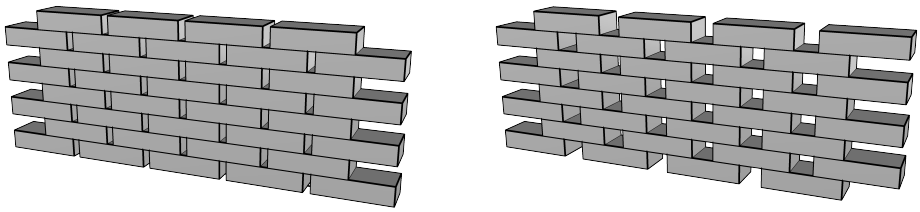

Figure 2.5: Brick gap, horizontal spacing

2.7 Other Terms

Usually in working with bricks, each horizontal set of bricks is called a course. In this project, the term row will be used instead, as a reminder that this project is using brick as an example of a modular unit, and thus that there is the possibility that the concepts discussed could be applied to other modular building units.

CHAPTER 3: MODULAR FORM MAKING LITERATURE REVIEW

Regular modules are commonly found in architecture and construction. A variety of work has been done on the use of modular construction units and the ways in which they can be utilized. Most of the existing work focuses on the use of robotics to create precise and complex forms [1] [2] [3]. However, the physical construction process is not what this project is focused on. Most robotic construction work uses a simple and custom-made computer model of the units as input into the robot. However, these custom models require a lot of work to create and a human to carefully consider if the designed structure is self-supporting or not.

The parametric wall is a simple coding problem that has been tackled many times [1] [4] [5] [6] [7]. This consists of making a script that takes the base number of bricks per row and the number of rows and places the bricks, creating something like what is seen in Figure 3.1.

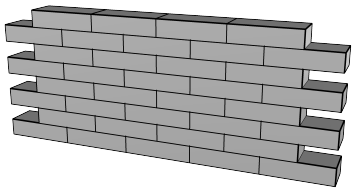

Figure 3.1: The simple wall

The BrickDesign Rhino plugin by ROB Technologies [8] allows the user to create a standard wall and control the variables of Bond type, Wall form, Mortar thickness, and Brick dimensions. The plugin does not allow for the rotation, translation, or removal of individual bricks. Additionally, the plugin is not user friendly and requires a fair amount of effort to figure out exactly what each tool in the plugin does and how

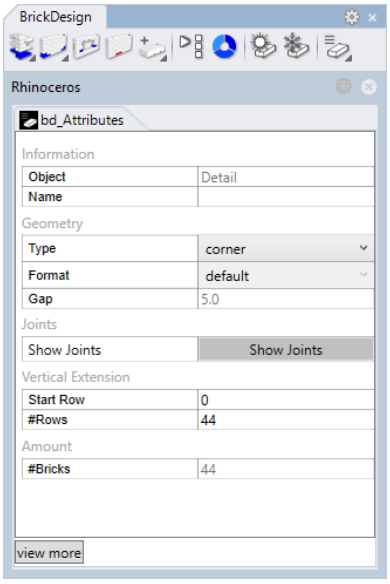

Figure 3.2: BrickDesign plugin by ROB Technologies [8]

to control the form of the structure. The plugin is formatted as a set of several Rhino toolbars with a variety of tools with names that are not clear as to their function (Figure 3.2). The use of this plugin even by someone familiar with the complexity of brick stacking requires a lot of trial and error and/or reading the extensive user's guide to figure out how to create even the most standard of walls. Many of the options to modify the wall are buried in a deep collection of object information menus and are difficult to find. With such a complex problem with so many variables, it is important to create an interface that assumes that the user is unfamiliar with all the terminology used, as well as the process of creating such a complex structure, and thus the interface should group the functionality into useful sections.

The simplest form of brick wall design is the use of rotation to create images on a standard wall. In the paper, Integrated Generative Technique for Interactive Design of Brickworks [7], Afsari utilizes image analysis to rotate each brick around its central vertical axis to create patterns and images on a simple brick wall (Figure 3.3). Afsari lays out the process in Grasshopper to create a parametric wall with customizable bricks and other variables, as well as the process for analyzing images and mapping

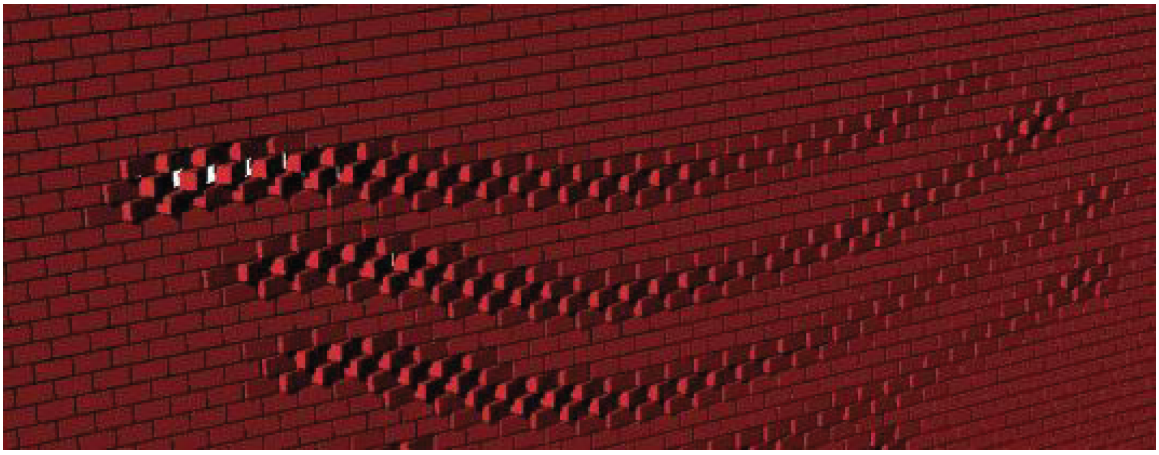

Figure 3.3: Pattern created on a simple brick wall by Afsari [7]

them onto the wall. This is one of the only papers that actually specifies the process that was done in such a way that a reader could have a hope of replicating the work, however the reader would need to utilize knowledge of coding to do so, making it inaccessible to the average designer. For its simplicity it lays the groundwork for creating more complex parametric models of brick walls, however it restrains the designer to simply creating an image on a flat wall.

Creating more complex forms of structures has been done in (Re)Thinking the Brick: Digital Tectonic Masonry Systems [4]. In this work Imbern looks at how bricks as a standard unit can be warped in simple ways during the manufacturing stage to create new and unexpected structures (Figure 3.4). This addresses the use of Wall form, Rotation, and Translation; however, it does so by modifying and warping the standard units, which takes away from the advantages of using a modular unit and is not easily replicable.

Wall form is the base of creating a unique structure with bricks. Parametric consideration of the wall form with guidelines is shown in Robotic Fabrication of Acoustic Brick Walls [2] which describes the process of utilizing a robotic arm to construct an acoustic wall system that utilizes geometry and principles of sound diffusion to create visually appealing and effective modular walls to diffuse sound. The paper shows

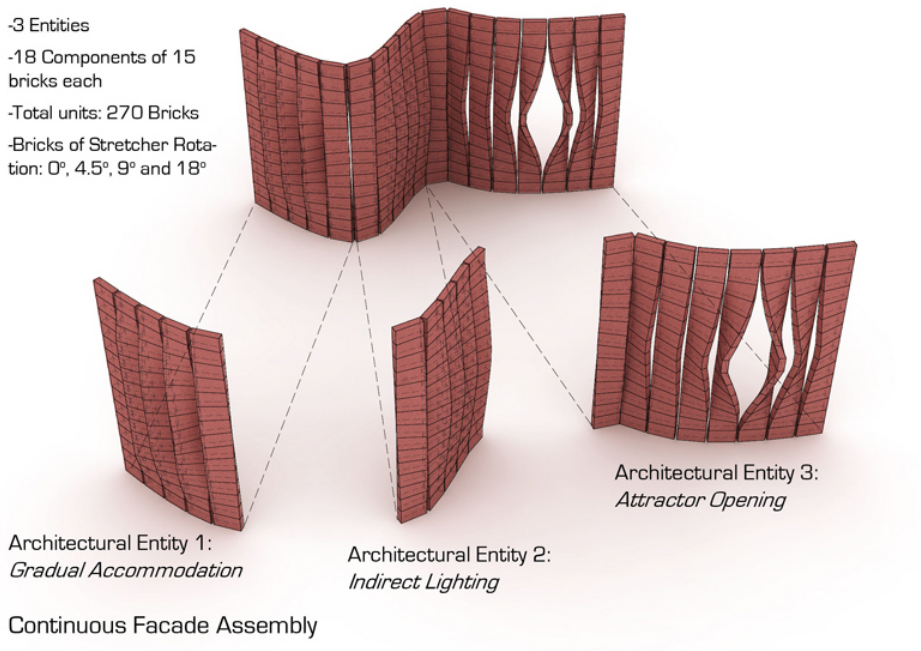

Figure 3.4: Unexpected brick structures by Imbern [4]

how the rules of acoustic diffusion can be applied to create a complex structure and then the program can utilize the structure and the base units to create a wall that structurally conforms to the needs of the unit, while also mimicking the imported form as closely as possible.

Wall form has also been addressed in the Amsterdam School of Architecture, which historically utilized bricks to create complex forms [9] [10] [11]. These works of architecture utilize rotation, translation, bond types, wall form, and removal. The designs are all complex and unique and create visual appeal (Figure 3.5). However, the creation of such designs requires a thorough understanding of the physics of bricks and their limitations, which is often not something that designers have or can easily learn for the sake of a design. These sorts of designs are what this project aims to make simpler to create and design.

Translation of individual bricks can create patterns and textures, especially when combined with rotation of bricks [7]. However, this project takes the concept even

Figure 3.5: Amsterdam school of architecture [9] - See the complex curves and arched forms created in this brick structure

further by combining these factors with complex wall forms, not just applying them to standard straight walls. Additionally, individual bricks can be removed from the system, leaving openings that add to the complexity of the structure, but leaving the structure susceptible to the perils of gravity. This project will take gravity and physics into account to analyze the structural stability of the construct and allow the user to remove individual units and customize the wall as they wish through a clear and simple interface.

There are papers and projects that address the different parts of creating a parametric construction, but there is not a publicly available, intuitive, and functional program that addresses all the transformations and manipulations that this project addresses with the ability to expand upon the functionality.

The concept of a modular unit is found widely throughout the building and construction industry, not just with bricks. A more general program with flexibility to

be applied to a variety of materials and situations beyond its creators initial intent would allow designers to utilize the ideas and techniques of bricks with other building materials. This flexibility in the size or proportions of modules is not found in any of the examples mentioned here. This project uses brick as a starting point, but is built in such a way as to allow the user to substitute any type of rectangular prism as the material, with the ability to completely control the dimensions of each unit.

CHAPTER 4: METHODOLOGY

The goal of this project is to create a plugin for Grasshopper for Rhino. The project starts with the base wall components, then moves to the basic structural analysis, then geometric translations, before finally doing the overall structural analysis.

The plugin provides components to input brick dimensions and spacing and the base geometry of the desired wall as a series of polyline profiles to loft into a surface.

After the base wall, there are two categories of components in the plugin. The first is geometric translations, and the second is structural analysis.

For the geometric translations, components allow the user to rotate, translate, or remove bricks in a strategic manner. The user interface allows the application of these geometric translations to selected bricks of the wall or to the whole wall in a pattern for more customizable control.

For structural analysis, there are three separate analysis types. First, the wall surface is analyzed to determine if it is likely to stand upright. Next, each individual brick is analyzed in relation to the row below to ensure that it is properly supported. Lastly, the whole wall is analyzed to find out the stability of the composite object.

For the wall surface analysis, the centroid of the surface is found and compared to the footprint of the base row of bricks. If the centroid is over the base, it is assumed that the structure will be stable. This is not a comprehensive or reliable method of analysis, in that it only looks at the centroid of the wall surface, not the center of mass of the final brick structure.

For the individual brick analysis the plugin implements an algorithm component to check that each brick unit is supported by the row below it, highlighting problematic bricks. This allows the user to check that a translation or movement has not led to

any single brick being inadequately supported. This method is a secondary check of the bricks, and could be useful in determining where issues would occur while stacking the bricks, but is not indicative of the stability of the structure as a whole.

For the final analysis, the plugin utilizes a physics engine for Grasshopper, to analyze the structure in relation to gravity. This allows the user to check the wall against gravity without having to code the whole of physics. This expedites the structural analysis programming and allow greater accuracy than the user can create themselves. Depending on the accuracy of the physics engine, this method could prove to be very reliable in determining the stability of the final brick structure.

Once basic functionality was completed, user testing was conducted with an early version of the user interface (Appendix A). Feedback from this testing was used to complete the final version of the functional interface (see Appendix B).

CHAPTER 5: FUNCTIONALITY

The base functionality of the plugin is a series of Grasshopper components which can be connected together to form a complete script.

5.1 Wall Surface Creation

There are 2 similar methods/components for creating the wall surface. The first is a simple 2 curve lofted wall. This has 4 inputs-

- The top curve of the wall
- The bottom curve of the wall
- Brick height
- The desired number of rows of bricks

The component takes these inputs and lofts a surface between the two curves with a height equal to the number of rows of bricks multiplied by the brick height.

The second option is similar to the first, but instead of 2 single curve inputs, there is a single list input of multiple curves. These curves are then spaced equally along the potential height of the wall, again using the brick height and number of rows.

5.2 Brick Wall Creation

The primary component is for wall creation. This component takes 6 inputs-

- The surface of the wall
- Brick length
- Brick width

- Brick height
- The horizontal gap between bricks
- The desired number of rows of bricks

The wall surface is first divided vertically at the level of each row of bricks. The contours of the wall at each row are then sorted into two alternating lists to create a list of even rows and a list of odd rows. The even rows are divided up into sections of the length of the brick plus the gap between bricks, and the points and corresponding vectors are stored in a tree structure. The same is done with the odd rows, but with the dimension divided in half and every other point removed to offset the rows of bricks. Then the data trees of points and tangents are interwoven to create a master data tree (Figure 5.1).

- Row 1
  - Brick 1
  - Brick 2
- Row 2
  - Brick 1
  - Brick 2

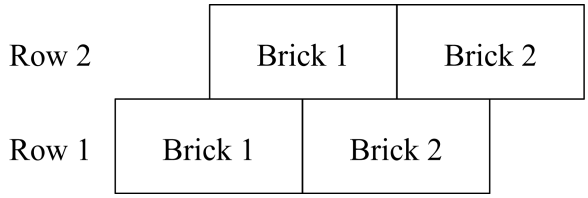

Figure 5.1: Data structure

Next, the tangent vectors and points are combined to create a plane for each brick to be placed upon and the bricks are generated with the provided plane at the geometrical center of the brick. The component then analyzes the data tree and the bricks and provides a variety of information as outputs as well as the final data tree that represents the wall.

5.3 Wall Analysis

The wall analysis is an initial analysis step to check the potential stability of the wall. It takes the surface and the brick geometry created and checks if the centroid of the surface is over the base of the bottom row of bricks. This allows the user to ensure the wall does not lean too far in any direction (Figure 5.2). This is not a comprehensive or 100% accurate analysis, it just alerts the user if the structure is likely to be unstable.

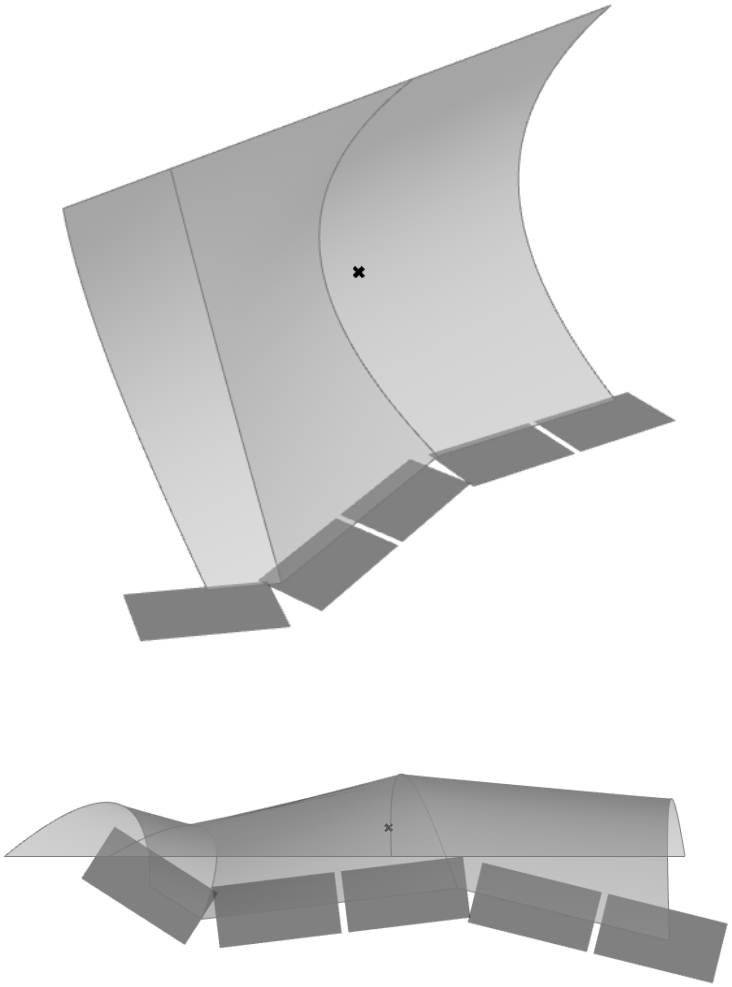

Figure 5.2: Wall centroid analysis using the surface of the wall, the centroid, and bases of the bottom row of bricks.

5.4 Rotation

The rotation component takes four inputs-

- The current Brick Geometry in a data tree
- The row of the brick to rotate
- The position of the brick in its row
- The degree of rotation

These inputs are then put into a C# script component that extracts the specified brick and rotates it around its centroid before placing the modified geometry back into the data tree and returning the now modified brick structure (Figure 5.3).

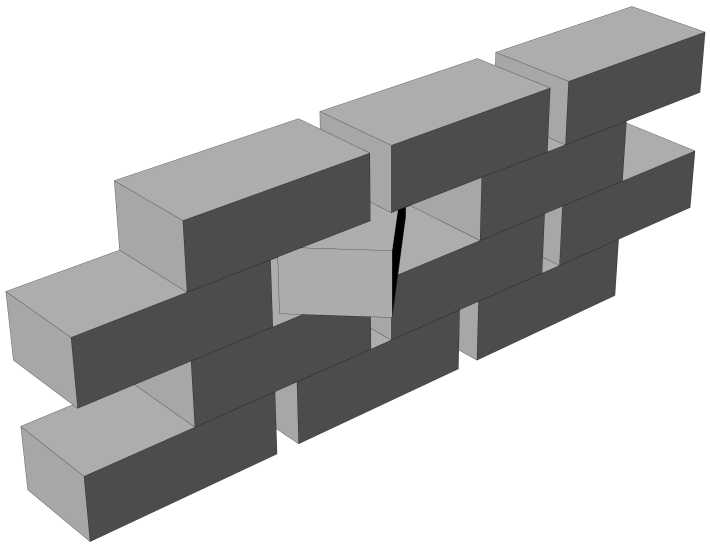

Figure 5.3: Rotation

5.5 Removal

The removal component takes three inputs-

- The current Brick Geometry in a data tree
- The row of the brick to remove
- The position of the brick in its row

A C# script then replaces the specified brick with a null value. This maintains the structure of the data. The component then returns the modified brick structure (Figure 5.4).

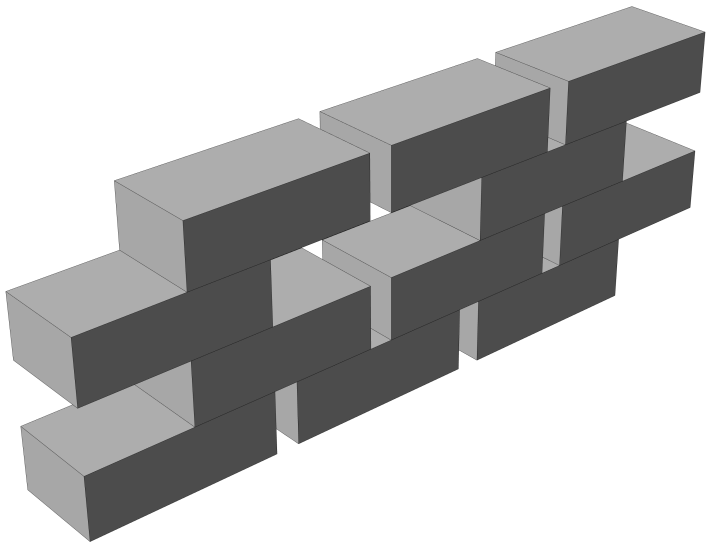

Figure 5.4: Removal

5.6 Translation

The translation component takes four inputs-

- The current Brick Geometry in a data tree
- The row of the brick to translate
- The position of the brick in its row
- The distance of translation

These inputs are fed into a C# script that extracts the specified brick, translates it forward or backward from its centroid the distance specified, with positive numbers translating forward and negative numbers translating backward. Then the modified geometry is placed back into the data tree and the component returns the now modified brick structure (Figure 5.5).

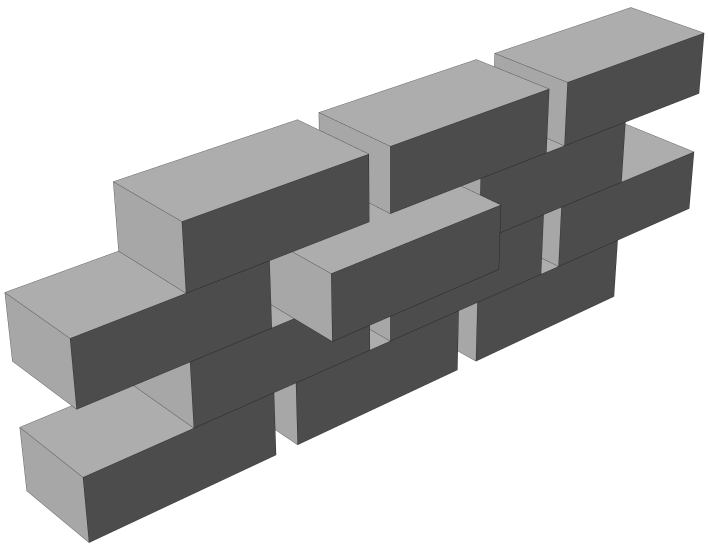

Figure 5.5: Translation

5.7 Individual Support Analysis

The next level of analysis is a check of each brick against the bricks below it to ensure that it is supported. This is done by selecting the two bricks below each brick and checking the area of the brick which overlaps the ones below. This overlapping area is then used to calculate the percent of the brick supported and compared to a percent set by the user to determine if the brick is supported or not. The component then returns a data tree of 0 and 1, with 0 indicating the brick is not supported and 1 indicating it is supported. The values of the data tree correspond to the brick geometry stored in with the same structure. This method of analysis is simple and can be easily scaled to handle varied size walls. While it can tell the user if each individual brick is supported a certain amount, this does not take into account where on the brick the support is located. A brick may only be supported by 20% of its surface, but if that 20% is placed with 10% on each of its two ends, the brick will be supported, whereas if that 20% is all at one end, cantilevering the brick, gravity will cause the brick to fall (Figure 5.6). Because of this, the initial percent supported analysis cannot be the final say in the stability of each brick, let alone the stability of the structure of a whole.

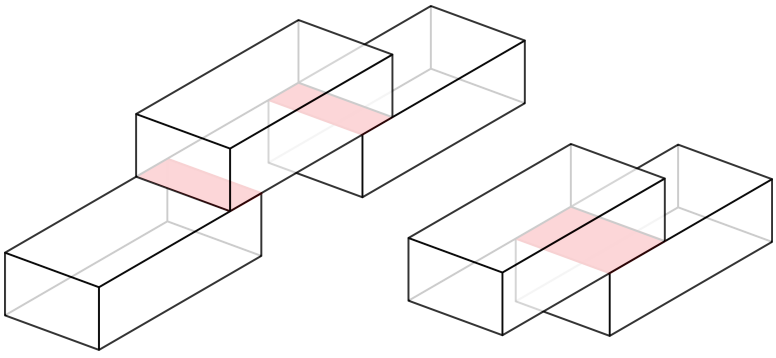

Figure 5.6: Support Variation- Two bricks supported by the same amount, but whose stability is clearly different.

5.8 Intersection Check

The intersection check component checks if any of the bricks are intersecting each other and outputs a data tree of true and false, with false indicating a brick that does not intersect another, and true indicating a brick that does intersect others. This component does not change the position of any overlapping bricks, it only alerts the designer to the issue.

5.9 Analysis- Kangaroo

The initial plan for further structural analysis was to use Kangaroo, a physics engine built for Grasshopper and Rhino [12]. Kangaroo applies materiality and forces to structures with specified support points. Given an individual pair of bricks, it is simple enough to apply a gravity force and test if a brick will fall when another brick is used as its support. However, with a large group of bricks all supporting each other, it becomes much more challenging to test the structure as each row of bricks must be tested then added to the supporting structure. This method is not easily scalable to varied numbers of rows. With simple spheres, a many to many test can be performed with Kangaroo's built in components, but there is no way to easily modify the component to take rectangular prisms instead.

5.10 Analysis- Karamba3D

The next method of physics analysis considered was Karamba3D [13]. Karamba is a Grasshopper plugin for structural analysis. Karamba's main use is for beams and other linear load structures such as trusses, but it also has some capability for analyzing shell structures. After some extensive trial and error, it seems that Karamba cannot analyze a brick wall in the way that is needed for this project, as it lacks the ability to analyze many discrete objects.

5.11 Analysis- PhysX.gh

The final method of physics analysis was PhysX.gh, a rigid body physics engine for Grasshopper that uses NVIDIA PhysX and C# [14]. With PhysX.gh it is simple to create each brick as an individual object and create a fixed base plane underneath the wall to simulate the effect that gravity has on the inputted structure (Figure 5.7).

PhysX.gh has not been officially validated. Its intended use is for game development, and thus it does not have a strong need to be 100% accurate. From basic use, it seems to be reliable and accurate. The results are identical when run repeatedly with the same input, and all the results are consistent with basic expectations.

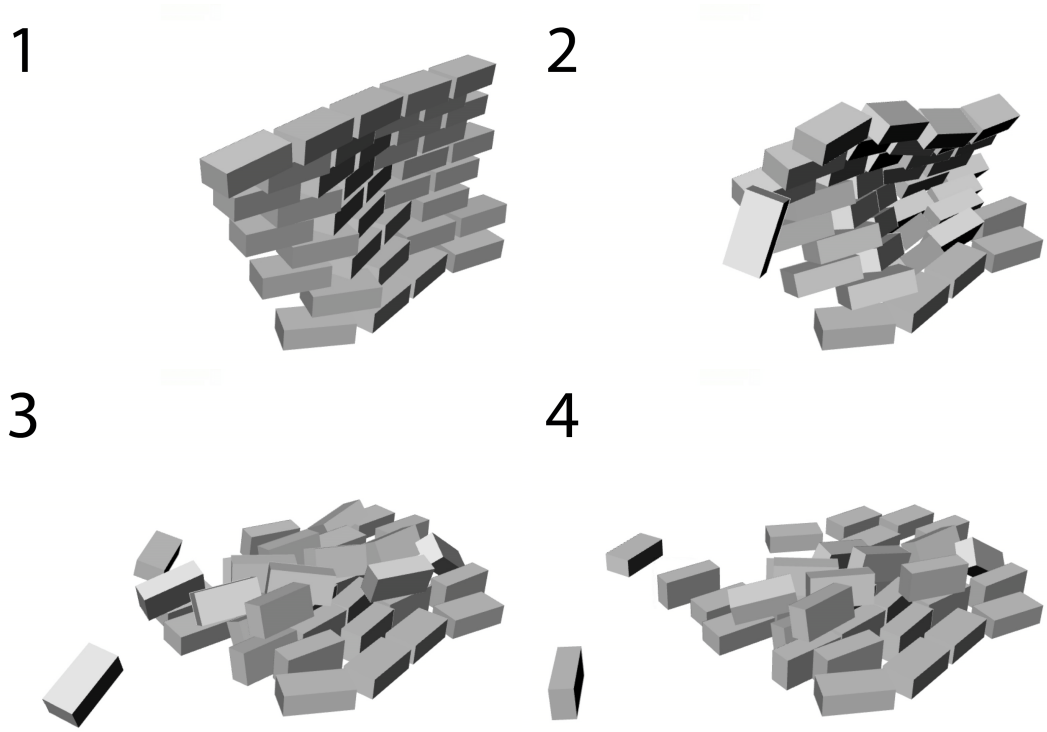

Figure 5.7: PhysX.gh gravity simulation

5.12 Display

The display component allows the user to control the display settings of the bricks, as well as providing some basic information.

The geometry is input as a data tree, from a variety of potential sources.

- The brick wall creation component (Section 5.2)
- The rotation component (Section 5.4)
- The transformation component (Section 5.6)
- The removal component (Section 5.5)
- The physics engine (Section 5.11)

The component then takes an input from the various analysis components and sets the display properties of each brick to indicate the status of the brick. For color, the component takes an input from the individual support analysis and uses this to choose colors to display the bricks. 1 will result in a green brick, 0.5 will result in a grey brick (for when this analysis is not displayed), and 0 results in a red brick. For transparency, the component takes an input from the intersection analysis and if there is a value of true in the data, the corresponding brick will appear transparent (Figure 5.8).

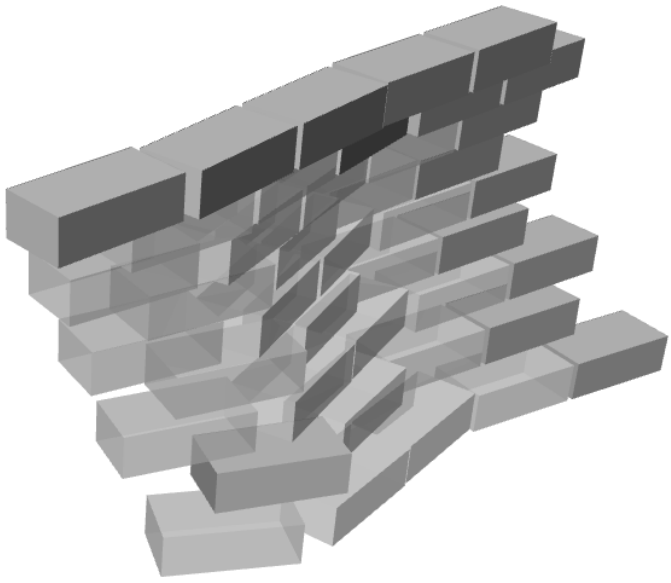

Figure 5.8: Bricks displayed as transparent due to overlap

5.13 Bake Rhino Geometry

The next component generates a random number using the current time as a seed and uses this number to bake the geometry of the wall to Rhino as a group. This can be done with either the base geometry color, or using the colors assigned by an analysis, similarly to the display component. This will allow users to save structures to compare or utilize within Rhino.

5.14 Functional User Interface - Human UI

The final functional user interface created for this project uses the Grasshopper plug-in Human UI [15] to create an interface that incorporates all the prior mentioned functionalities. The interface consists of a set of collapsible sections, each controlling a different component of the functionality.

The first section contains two radio buttons to select the wall surface input type (Figure 5.9). Each then has buttons underneath that prompt the user to select the appropriate curves for the wall from the Rhino document. If the wall analysis check determines that the centroid is not over the base an alert is displayed to tell the user that it is likely that the wall will not stand upright.

The second section provides the user with sliders to control the dimensions of the bricks, the number of rows, and the gap between bricks (Figure 5.9). A drop down menu provides some options for standard brick sizes. The gap between bricks is limited to the length of the brick and changes dynamically as the brick length is modified. If the intersection check discovers any intersecting bricks, an alert is shown to the user to tell them what the issue is and to recommend that the easiest solution is to increase the gap between the bricks (Figure 5.10).

The third, fourth, and fifth sections provide the controls for Rotation, Removal, and Translation respectively (Figure 5.11). Each contains a check box to activate or deactivate the transformation and selection type. Pattern transformation prompts the

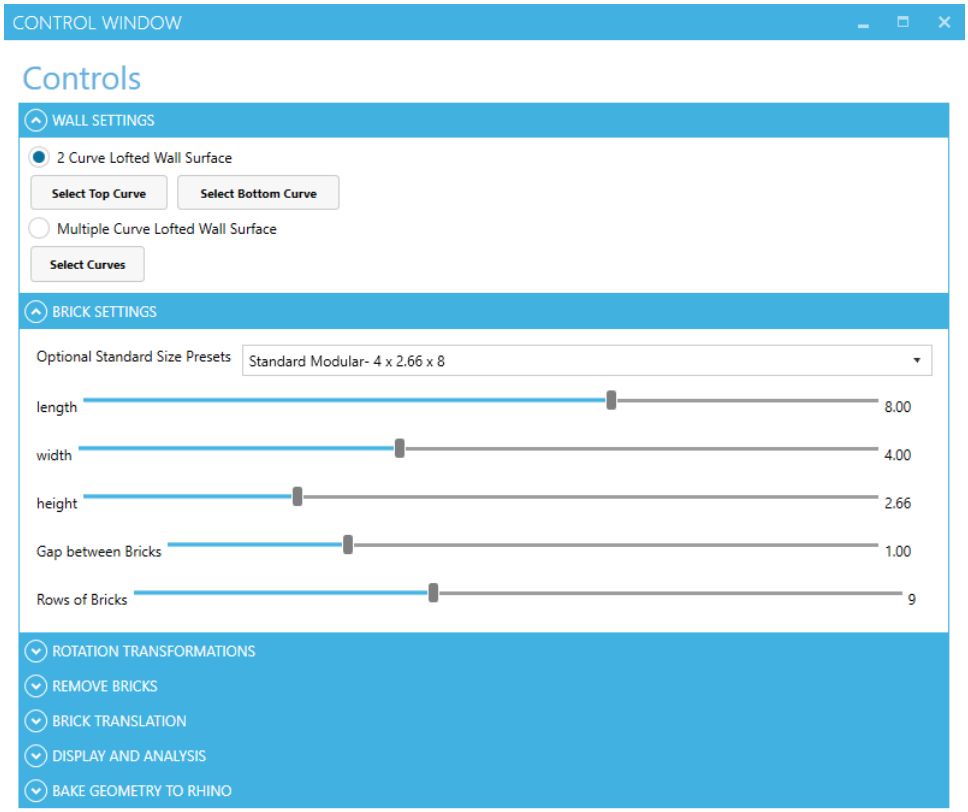

Figure 5.9: Wall and brick user interface controls

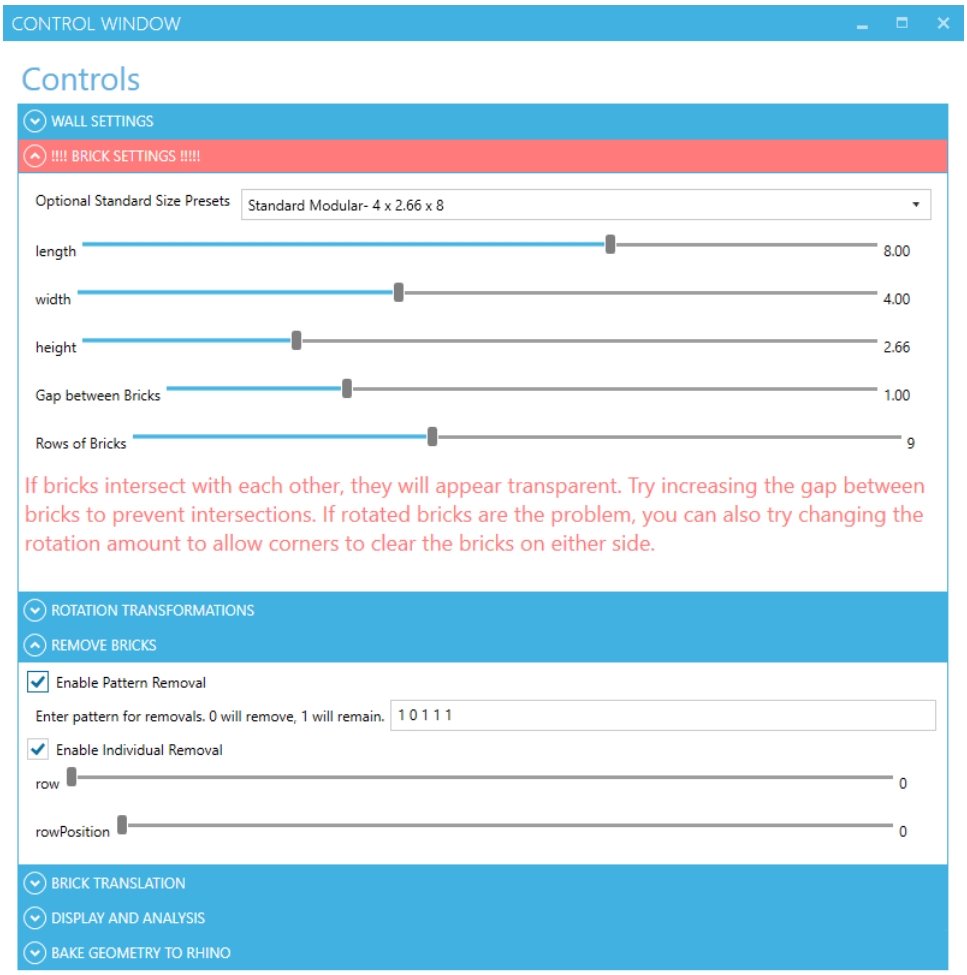

Figure 5.10: When two bricks intersect, an alert in the user interface will let the user know and recommend how to fix the issue. The bricks will also appear transparent in the Rhino display.

user to enter a series of numbers separated by spaces to indicate the transformation on each brick in series. For rotation and translation, the number indicates either the degrees of rotation or the distance of translation, while 0 indicates a brick left alone. For removal, 0 leaves the brick in place while 1 removes it. Individual transformation prompts the user to select the row and position in the row of the desired brick. For rotation and transformation, individual transformation also provides a slider to select the degrees of rotation or the distance of transformation. In individual transformation mode, the distance of translation is limited to the width of the brick.

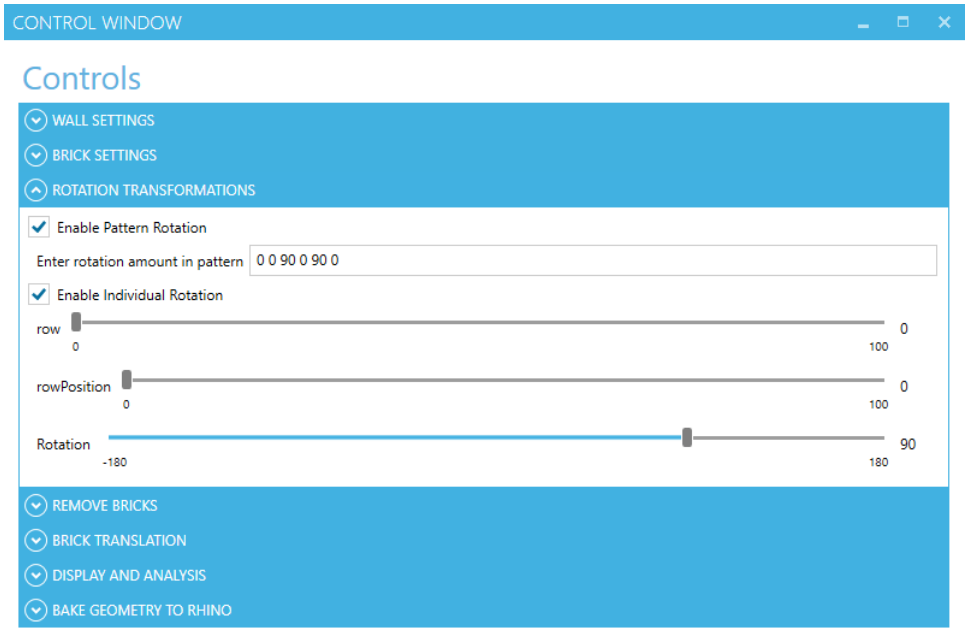

Figure 5.11: User interface transformation selections

The sixth section provides the controls for the user to toggle the initial support analysis visuals as well as the PhysX.gh physics analysis (Figure 5.12). Each type of analysis is toggled by a checkbox, and when enabled will show its options beneath it. The brick supported analysis only offers one option, the percent to be supported. The physics analysis provides a button to reset the geometry and a toggle to turn on and off the running of the physics simulation.

The seventh and final section of the UI contains the buttons to allow the user to

bake the geometry of the wall to Rhino (Figure 5.12).

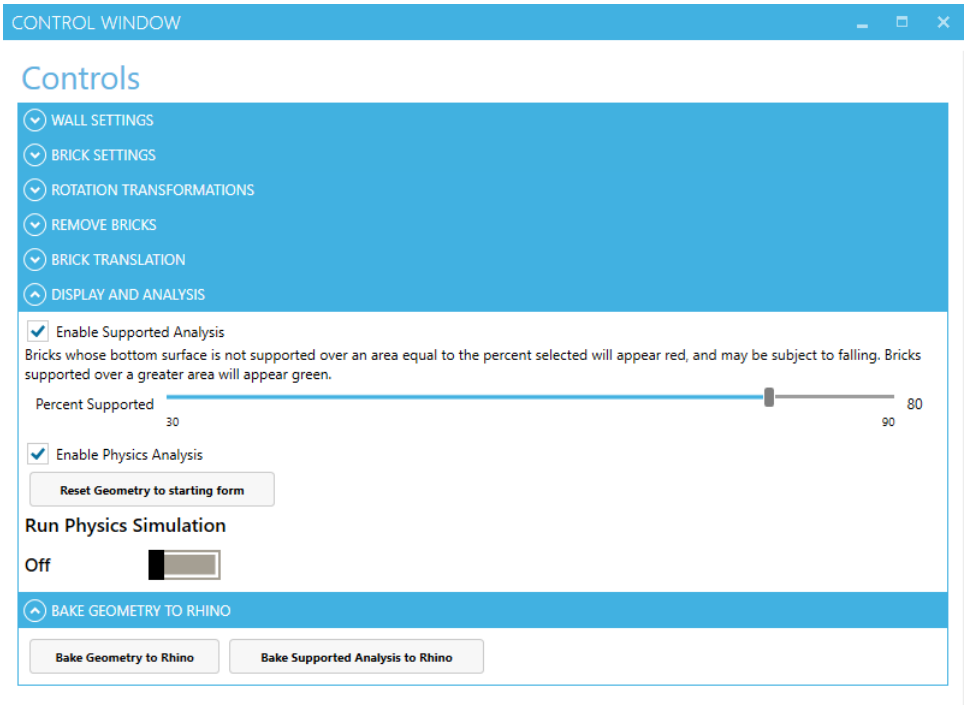

Figure 5.12: User interface analysis controls and bake to Rhino buttons

CHAPTER 6: USER TESTING

Once basic functionality was completed, the next step was to conduct user testing on the software. The User Testing was conducted on the initial user interface described in Appendix A and was conducted online. Four participants, all architecture students, were recruited through the UNCC School of Architecture.

Participants were provided with instructions on how to install Human UI [15] on their computers, then provided with a Grasshopper file containing the first draft of the user interface. Participants were asked to spend around 15 minutes using the tool and attempting to recreate an example wall (Figure 6.1).

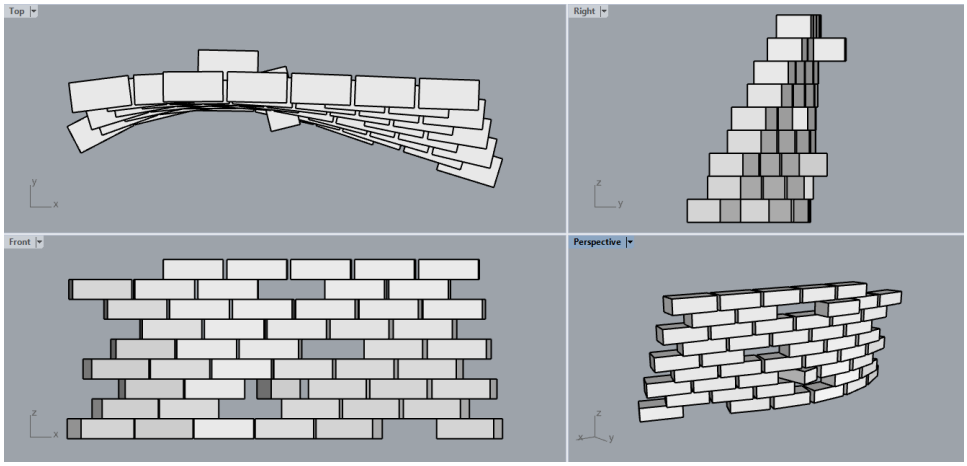

Figure 6.1: The example wall provided for user testing

Participants were then asked to complete a short questionnaire on their experience with the tool. The questions asked were as follows:

1. What did you find most challenging about replicating the example?

2. If you were asked to design a similar structure without this tool, how would you go about doing so?

3. Were there any particular ways in which you would have liked to be able to manipulate the structure that you were unable to?

4. Did you encounter any particular issues with the interface or software functionality not working the way that you expected it to? If so, what happened?

5. Did the structural analysis display assist you in your design process?

6. Were you able to be creative within the limitations of the interface?

Full participant responses can be found in Appendix B.

In general, user feedback requested further help and descriptive text throughout the document, with a general positive feeling on the usefulness of the tool.

Suggested functionalities included an undo function, live updates, and highlights to show which brick is selected. Live updates were implemented in the final version of the interface (see Section 5.14). An undo function is outside of the scope of this project, though it would be useful. Highlighting the brick selected raises some workflow questions about when the highlight would go into effect, and was not been implemented. The additionally requested help text was included in the ideal design (see Appendix D), and added to later versions of the functional script (see Section 5.14).

The responses to the user study can tell us a lot about the potential value of the project. The second research question of this project (see Chapter 1) is about the potential effectiveness and value of the proposed user interface. The answers to the questionnaire give us a chance to evaluate the ways that the participants feel that the project would impact their design process. The participants responded favorably to the ways in which the project could assist or even guide them to try new forms. One participant stated that the basic structural analysis was "the most interesting part of it" as "it can act as a guide." All four participants stated that the interface was able

to allow them to be creative, with one participant saying that "it helped with trying different alternatives easily."

CHAPTER 7: FUTURE PLANS

In the future, this project could be converted from Grasshopper clusters to full C scripts in order to become a full fledged Grasshopper plugin. This final version of the plugin would likely want to include the user interface in the Grasshopper components as seen in Appendix D. This would enable the plugin to seamlessly fit into Grasshopper without needing the additional Human UI plugin. It would also allow the advanced user to modify the workflow as they see fit and incorporate the output geometry into custom scripts to do anything they want.

It would also be useful to do more extensive and comprehensive user testing. The original plan was to conduct face to face demos and interviews with individual participants, however due to COVID-19 arriving in NC in mid March 2020, user testing was moved online and completed asynchronously. The testing completed in the course of the project was a small sample group working with an early version of the UI, and as such, the information collected is a small glimpse as to the insights that could be collected with more in depth user testing and observation of users interactions. Future testing would require more participants and could be done with a more complete version of the plugin for better results. It would also be beneficial for further user testing to be recorded in order to analyze user interaction more fully. This would expose any issues or small UI problems that are not obvious to users and thus are not reported in the survey.

Future work also includes potential other methods or types of structural analysis. The current analysis only looks at gravity, but the reality of the world is that there are many other forces at work that can affect anything we build and must be taken into account. It would be useful to look at the application of lateral loads to the structure

and how the structure and form would impact the ability to withstand such forces. It is well established that curved walls are stronger than straight walls, and this can allow for some interesting form creation with single brick structures. This concept is found in the notes of Thomas Jefferson [16] and can be seen on the campus of UVA (Figure 7.1). Incorporating analysis of lateral loads would give the designer useful feedback on the structural integrity of their design, extending the current analysis.

Figure 7.1: Serpentine walls on the campus of UVA [17]

CHAPTER 8: CONCLUSIONS

The final version of the user interface includes all the functionality planned for in this project. While the final product is not a fully functional plugin, it is a functional script. The analysis was successful in showing the structural integrity of the structure at a variety of points in the script as well as through a variety of methods.

The initial research questions for this project were about functionality and the effectiveness of a user interface to implement the functionality.

In Chapter 3 we looked at the Rhino plugin Brick Design by ROB Industries [8]. This plugin has a lot of functionality in overall wall form, but lacks the functionality to transform individual modules within the structure. It focuses solely on bricks as a modular unit, which restricts the user in ways that this project aims to free the user. This project also intended to look at the effectiveness of a user interface (see Chapter 1) with the consideration of the novice user, with the menu interface of BrickDesign as a baseline of what could be done to guide the user through the creation and design of a structure. The user study was instrumental in guiding the direction of the final design to be user friendly and intuitive to start using quickly. The participants all reported that they were able to quickly figure out what they could do and how to do it, showing that even with an early version of the user interface, this project was successful in improving upon the options already available.

REFERENCES

[1] J. P. Sousa, P. A. Varela, and P. F. Martins, "Between Manual and Robotic Approaches to Brick Construction in Architecture," *Proceedings of the 33rd eCAADe Conference - Volume 2, Vienna University of Technology, Vienna, Austria*, 2015.

[2] M. Vomhof, B. S. Vasey, Lauren, K. Eggenschwiler, J. Strauss, F. Gramazio, and M. Kohler, "Robotic Fabrication of Acoustic Brick Walls," *ACADIA 14: Design Agency*, 2014.

[3] J. Braumann and S. Brell-Cockcan, "Parametric Robot Control: Integrated CAD/CAM for Architectural Design," *ACADIA 11: Integration through Computation*, 2011.

[4] M. Imbern, "(Re)Thinking the Brick: Digital Tectonic Masonry Systems," *Rethinking Comprehensive Design: Speculative Counterculture*, 2014.

[5] A. Fingrut, K. Crolla, and D. Lau, "Automation Complexity - Brick By Brick," *Intelligent & Informed - Proceedings of the 24th CAADRIA Conference - Volume 1*, 2019.

[6] T. Bonwetsch, D. Kobel, F. Gramazio, and M. Kohler, "The Informed Wall: applying additive digital fabrication techniques on architecture," *Synthetic Landscapes [Proceedings of the 25th Annual Conference of the Association for Computer-Aided Design in Architecture]*, 2006.

[7] K. Afsari, M. E. Swarts, and T. R. Gentry, "Integrated Generative Technique for Interactive Design of Brickworks," *Digital Crafting [7th International Conference Proceedings of the Arab Society for Computer Aided Architectural Design]*, 2014.

[8] R. Technologies, "BrickDesign."

[9] W. Herfst, *The Amsterdam School*. Architectura & Natura, Jan. 2017.

[10] M. Casciato, ed., *The Amsterdam school*. Rotterdam: 010 Publishers, 1996.

[11] W. d. Wit, ed., *The Amsterdam school: Dutch expressionist architecture, 1915-1930*. New York : Cambridge, Mass: Cooper-Hewitt Museum ; MIT Press, 1983.

[12] McNeel, "Kangaroo physics."

[13] C. Preisinger and B. und Grohmann ZTGmbH, "Karamba3d."

[14] G. Ting-Chun Kao, L. Nguyen, and T. A. Coders, "Physx.gh."

[15] A. Heumann, M. Syp, N. Holland, and B. Ringley, "Human UI."

[16] T. Jefferson, "Jefferson drawing for the serpentine walls 1934.," Place: Charlottesville, Va Publisher: Special Collections, University of Virginia Library.

[17] T. Inc, *LIFE*. Time Inc. Google-Books-ID: xEQEAAAAMBAJ.

APPENDIX A: USER INTERFACE FOR USER TESTING

This early version of the user interface built with Human UI[12] was used for user testing. It is similar to the final version, with the main differences being the lack of real time updates and physics analysis. Some other functionality was also added later based on feedback from the users as well as the committee members.

The first section contains two buttons that prompt the user to select either the top or bottom curve of the wall from the Rhino document (Figure A.1).

The second section provides the user with sliders to control the dimensions of the bricks, the number of rows, and the gap between bricks. A drop down menu provides some options for standard brick sizes. The gap between bricks is limited to the length of the brick and changes dynamically as the brick length is modified (Figure A.1).

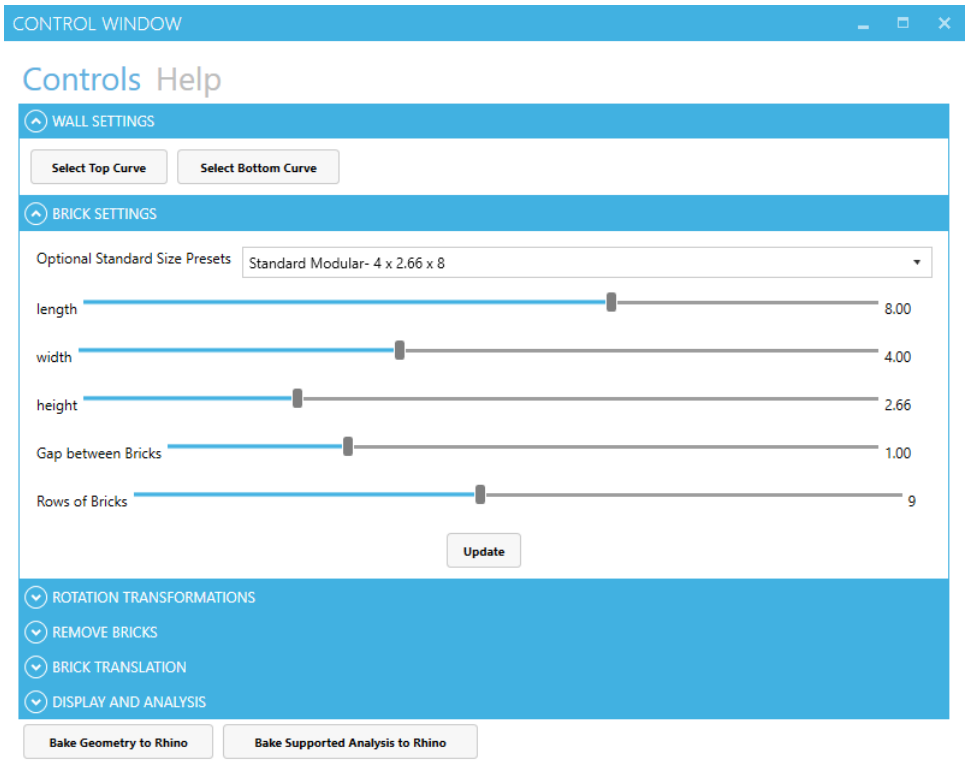

Figure A.1: User testing UI - wall and brick controls

The third, fourth, and fifth sections provide the controls for Rotation, Removal, and Translation respectively. Each contains a check box to activate or deactivate

the transformation type and a drop down to choose between pattern and individual transformation. Pattern transformation prompts the user to enter a series of numbers separated by spaces to indicate the transformation on each brick in series (Figure A.2). Individual transformation prompts the user to select the row and position in the row

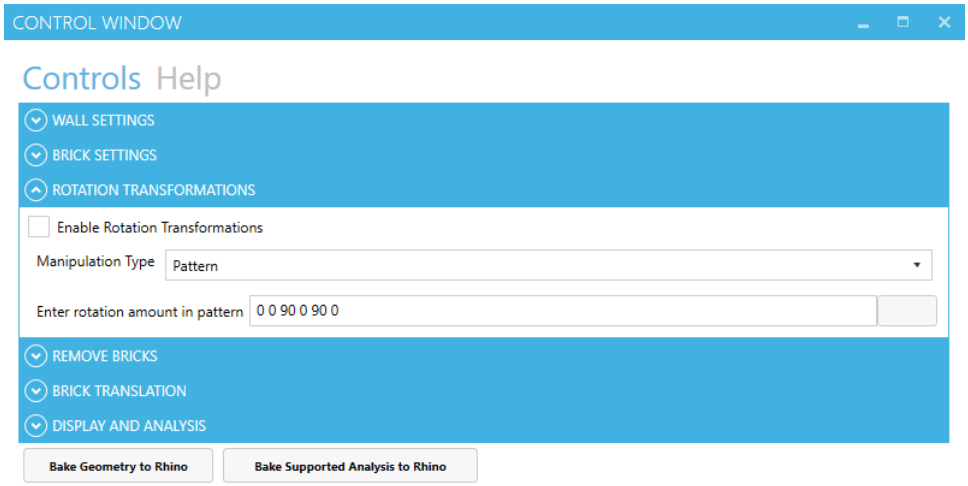

Figure A.2: User testing UI - pattern selection

of the desired brick, as percentages of the total number (Figure A.3). For rotation and transformation, individual also provides a slider to select the degrees of rotation or the distance of transformation. The distance of transformation is limited to the width of the brick.

The sixth section includes controls to select the display type. The user can choose from the drop down if they wish to display the supported analysis or not (Figure A.4). When supported analysis is turned on, the user is also given the option to choose the percent that the brick needs to be supported in order to appear as supported.

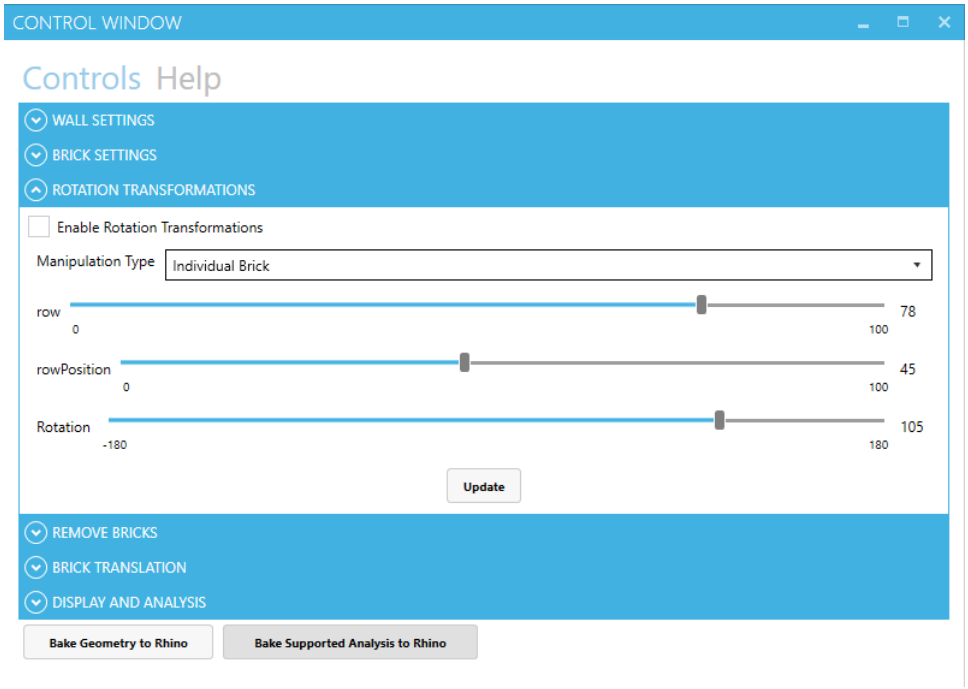

Figure A.3: User testing UI - individual brick selection. Here a selection of row 78 and row position 45, selects the closest to the 78% of the number of rows and 45% of the number of bricks in that row, or a brick about 3/4 of the way up the wall and near the center.

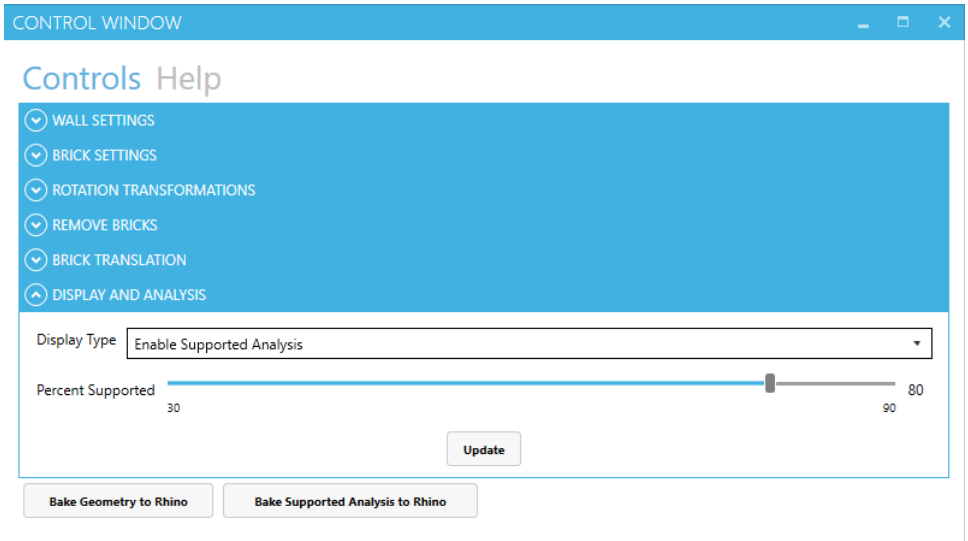

Figure A.4: User testing UI - display and analysis

APPENDIX B: USER STUDY RESPONSES

The User Study had a total of 4 participants who were able to complete the exercise and responded to the questionnaire. Their responses are reproduced below as they were provided.

1. What did you find most challenging about replicating the example?
    - Drawing the base curves.
    - Using the interface.
    - To understand difference between individual brick vs pattern took some time but after that it was easy.
    - Understanding the UI.
2. If you were asked to design a similar structure without this tool, how would you go about doing so?
    - I had to do it manually.
    - I would create a Grasshopper script to generate the desired structure.
    - By building a grasshopper/python script.
    - It is pretty mind boggling to do so specially with the gravity complexity.
3. Were there any particular ways in which you would have liked to be able to manipulate the structure that you were unable to?
    - Having multiple curves.
    - No, the program has multiple manipulation methods, or at least the most important ones.
    - I would have like automatic update to see how each change would affect the design. To click update after every move was breaking my flow of thought.

- No it looked sophisticated enough.

4. Did you encounter any particular issues with the interface or software functionality not working the way that you expected it to? If so, what happened?

   - It crashed several times I had to open it and re do it again.
   - Yes, some of this include the lack of an "undo" functionality, which made it hard to go back to previous iterations.

     I find the "update" button to be completely ineffective as the user is not watching changes to be done in real time making it hard to visualize what exactly is being modified.

     The rotation, translation and removal tools although are simple concepts, the implementation is problematic, given that in the "pattern" manipulation type there is not clear on what the numbers mean and how exactly will they affect the structure.

     There was no highlight on the selected brick or row that was being modified, which makes it hard to see what is being modified at the moment.

     On the rotation, translation and removal tools on the "individual" manipulation type following the previous point it is hard to know which brick is modified without having to actually count which brick is being selected.
   - Couldn't understand what the pattern string was for example what does 010101 mean. how many digits to we have to specify?
   - No.

5. Did the structural analysis display assist you in your design process?

   - Yes it made me aware of my design choices.
   - No, because, since I'm able to modify the "percent supported" it is easy to trick the system. Also, I'm not entirely sure how it detects which bricks

are supported and which are not and why. Maybe a little text indicating that bricks with certain amount of overhang will not be supported would be helpful. As in some cases where some bricks were in similar positions some where indicated as "not supported" and others "were supported".

- No, the analysis didn't make sense, it was showing red when it should have been green. I couldn't understand how it was working.
- That was the most interesting part of it it can act as a guide.

6. Were you able to be creative within the limitations of the interface?

    - Yes indeed it helped with trying different alternatives easily.
    - Yes.
    - Yes.
    - Yes I wish it was more space.

APPENDIX C: GRASSHOPPER SCRIPTS

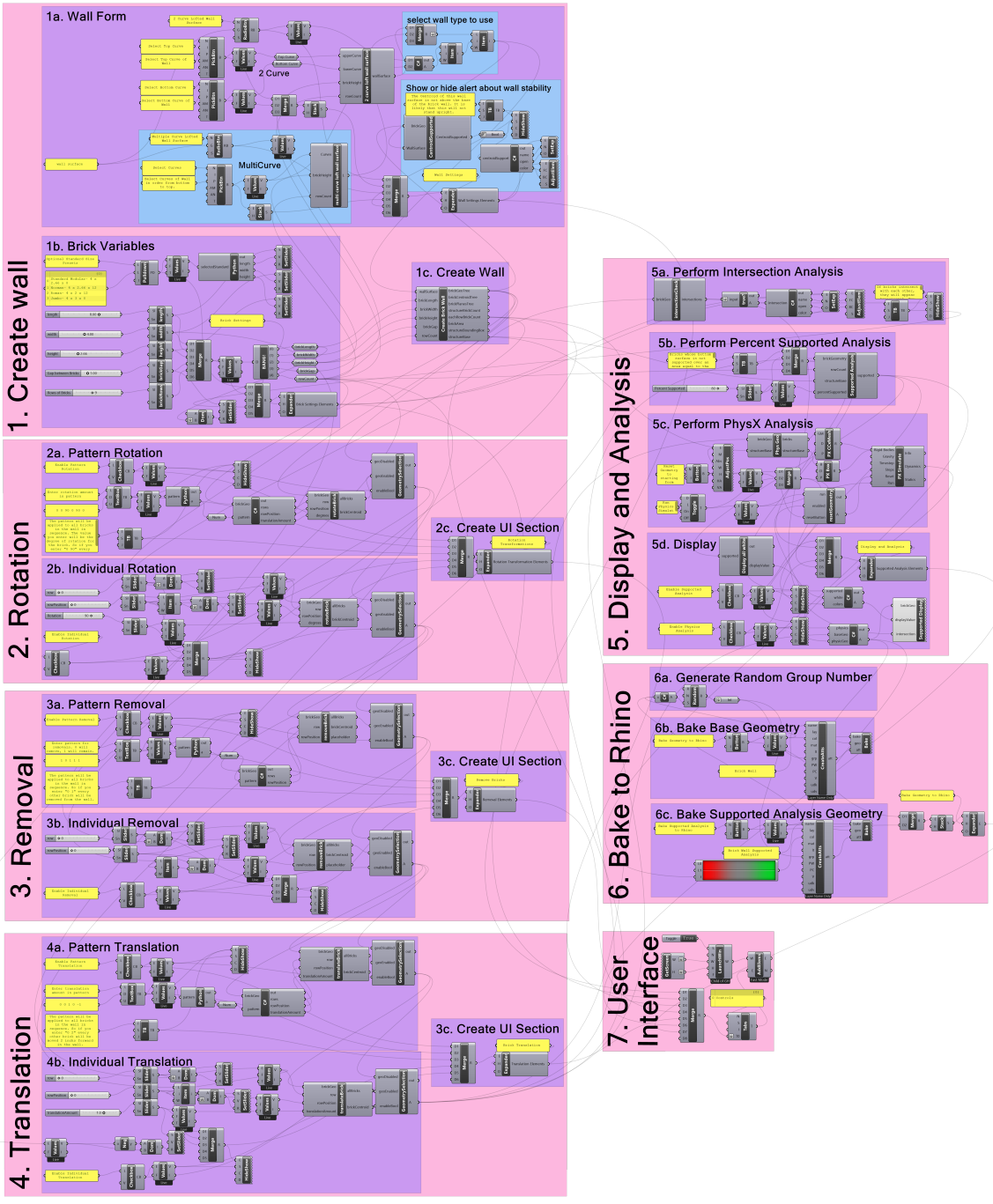

Figure C.1: Thumbnail of the Grasshopper code for the UI

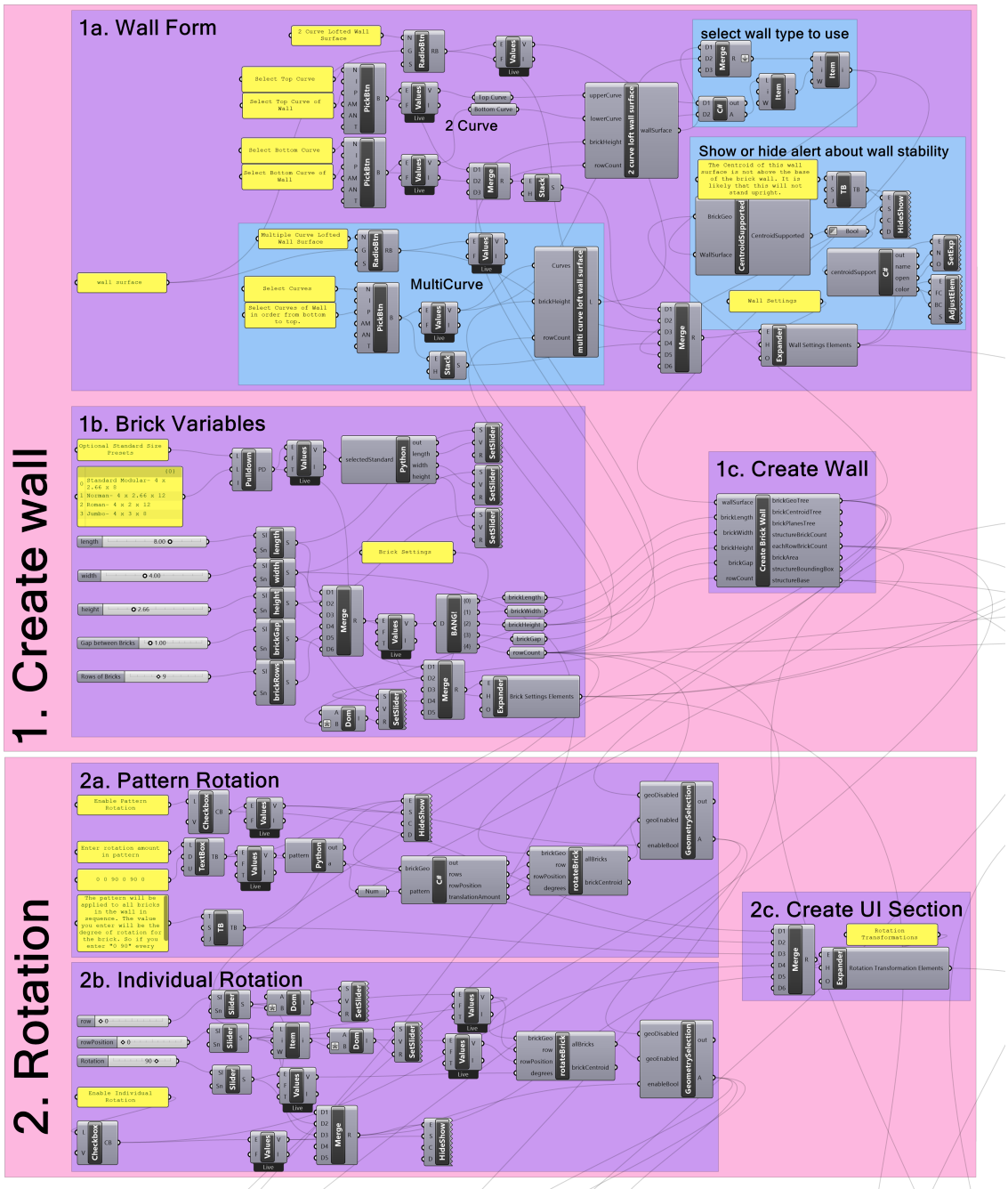

Figure C.2: Detail 1 - the Grasshopper code for the UI

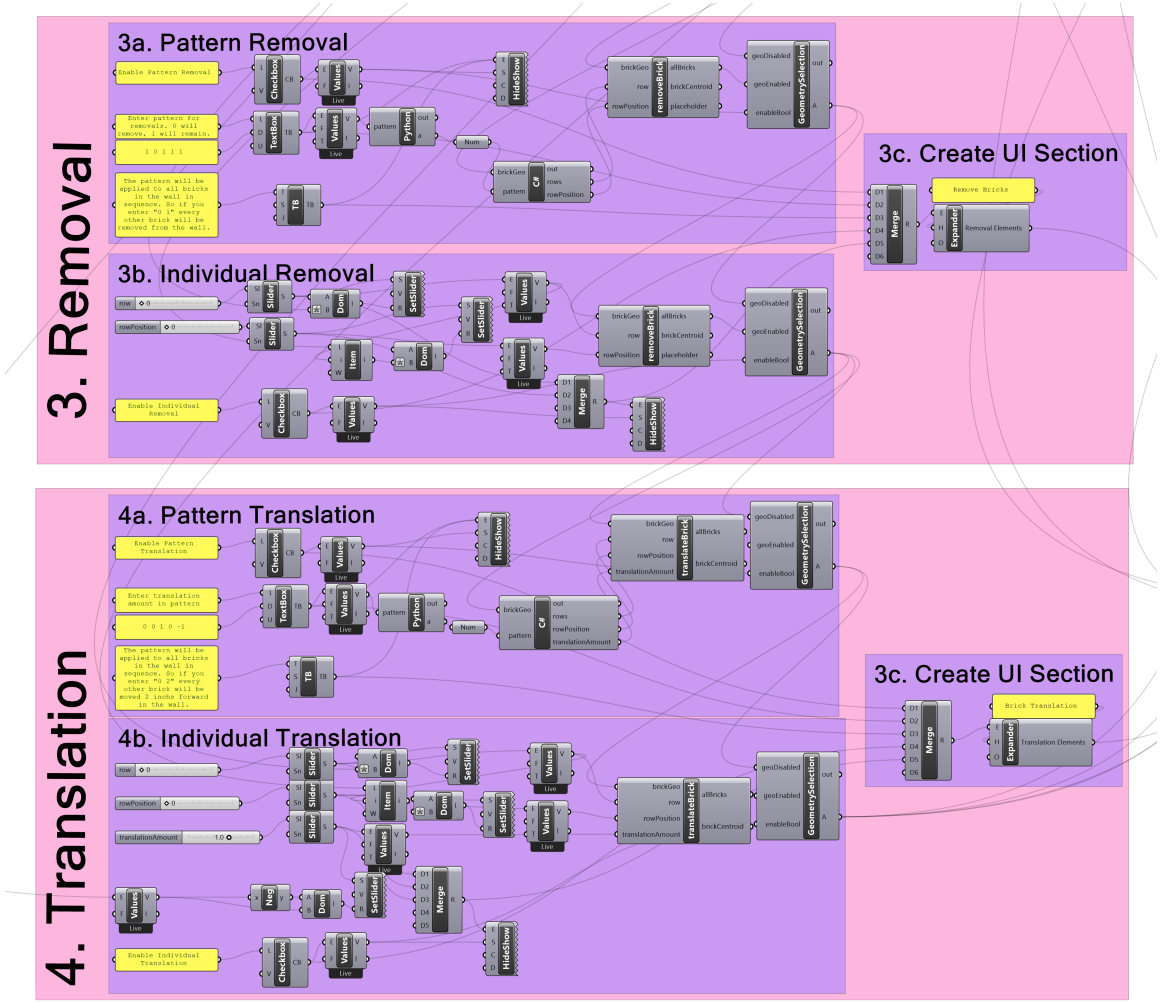

Figure C.3: Detail 2 - the Grasshopper code for the UI

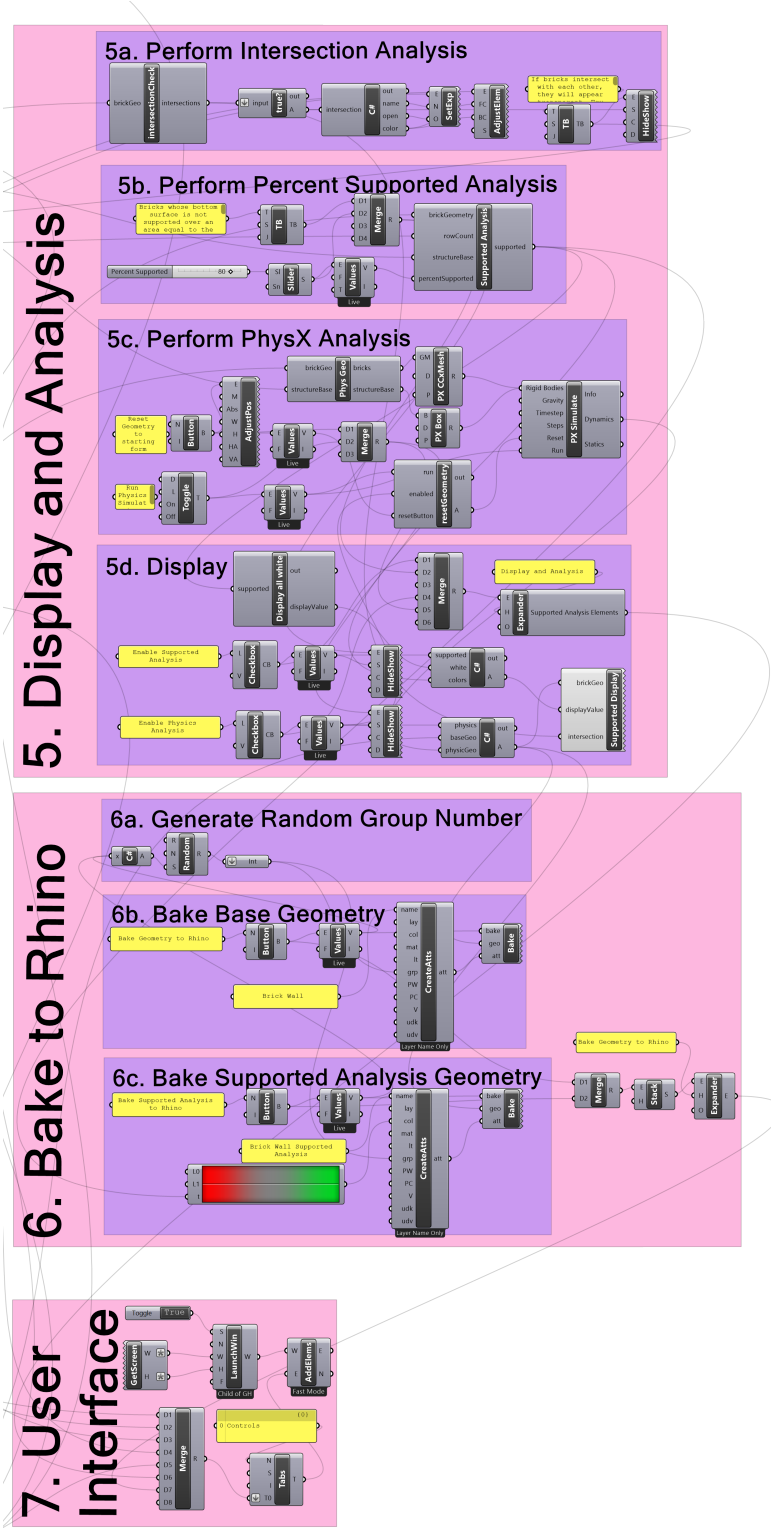

Figure C.4: Detail 3 - the Grasshopper code for the UI

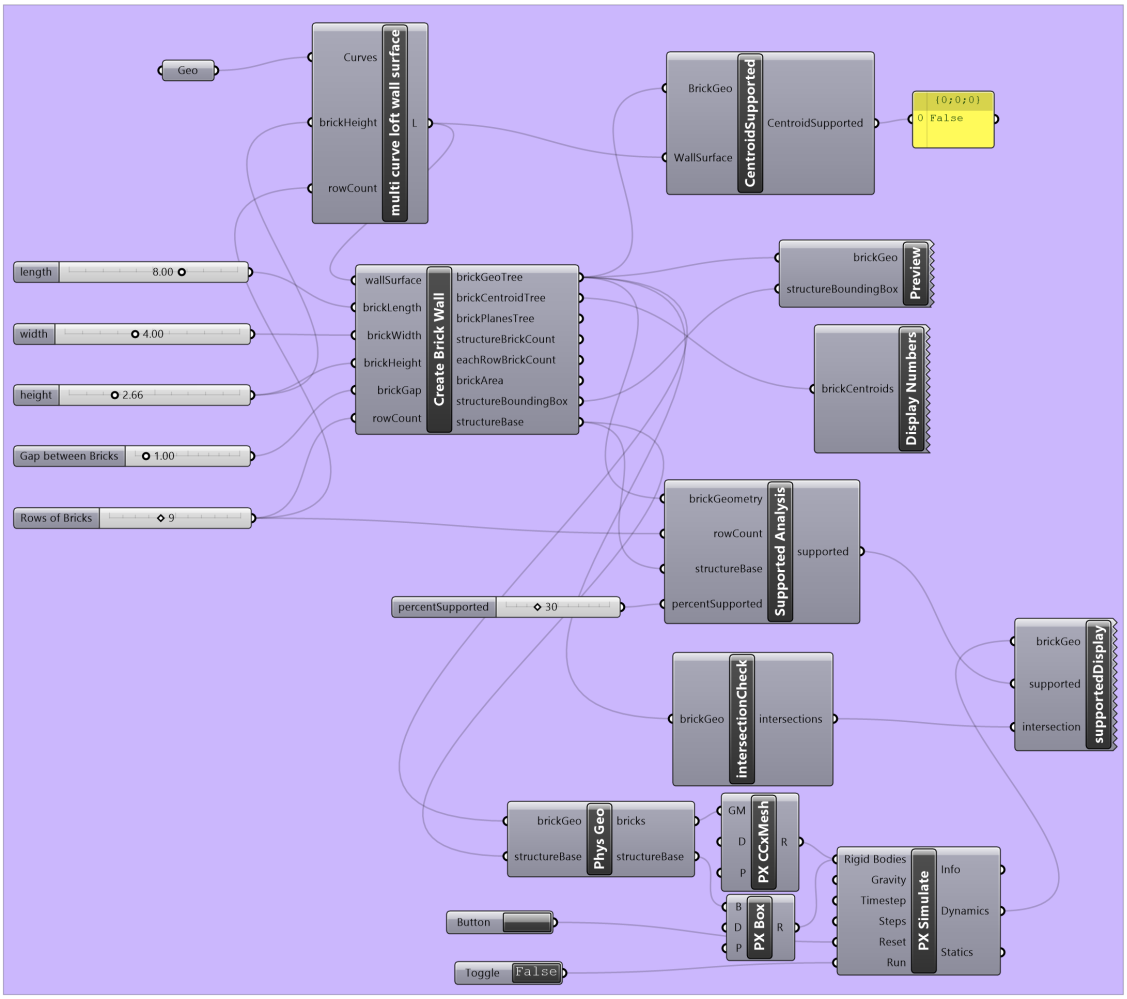

Figure C.5: The base Grasshopper script with no UI or transformation

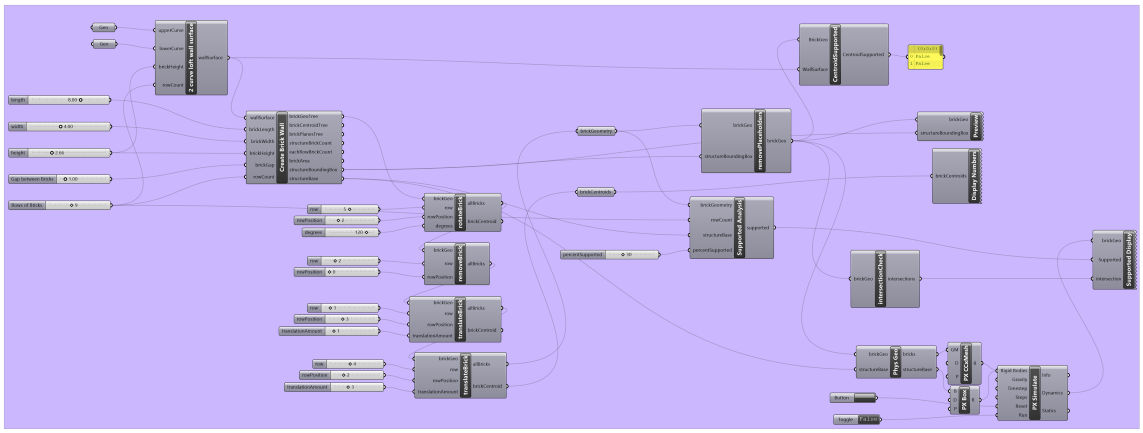

Figure C.6: The base Grasshopper script with no UI

APPENDIX D: IDEAL UI DESIGN

Ideally, the plugin would function as individual components within Grasshopper, each with a collapsible section for options like brick dimension sliders and selection types. This would enable users to embed portions of the plugin into their own Grasshopper components, as well as guide users through the Grasshopper script. Users would be able to add multiple of the various transformation types in series with each other, as well as to display what they want to see. The simple script as it may look is shown in Figure D.1. The full components with all the expanded sections are shown in Figure D.2.

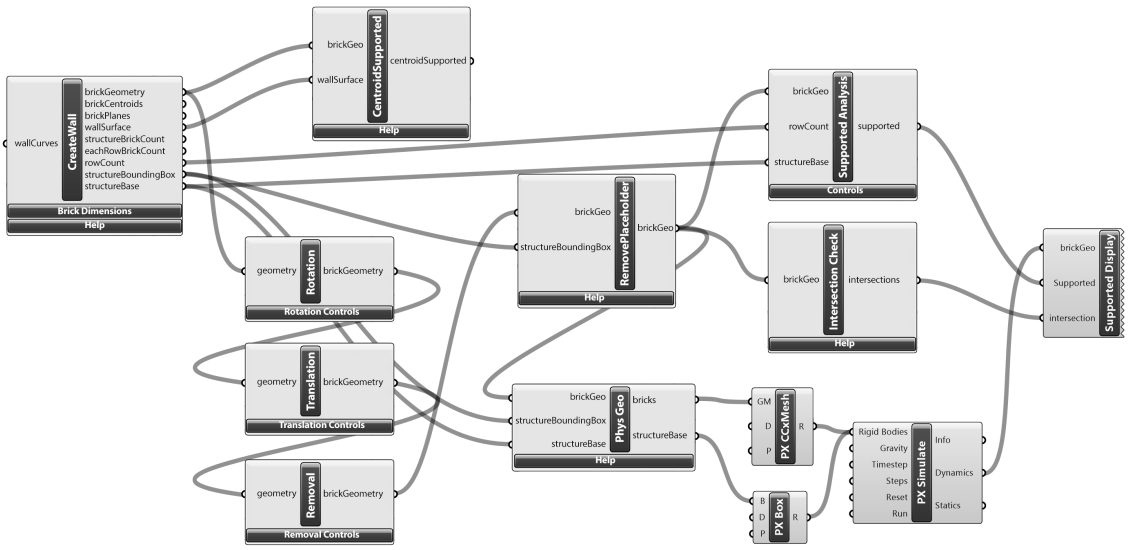

Figure D.1: The ideal script

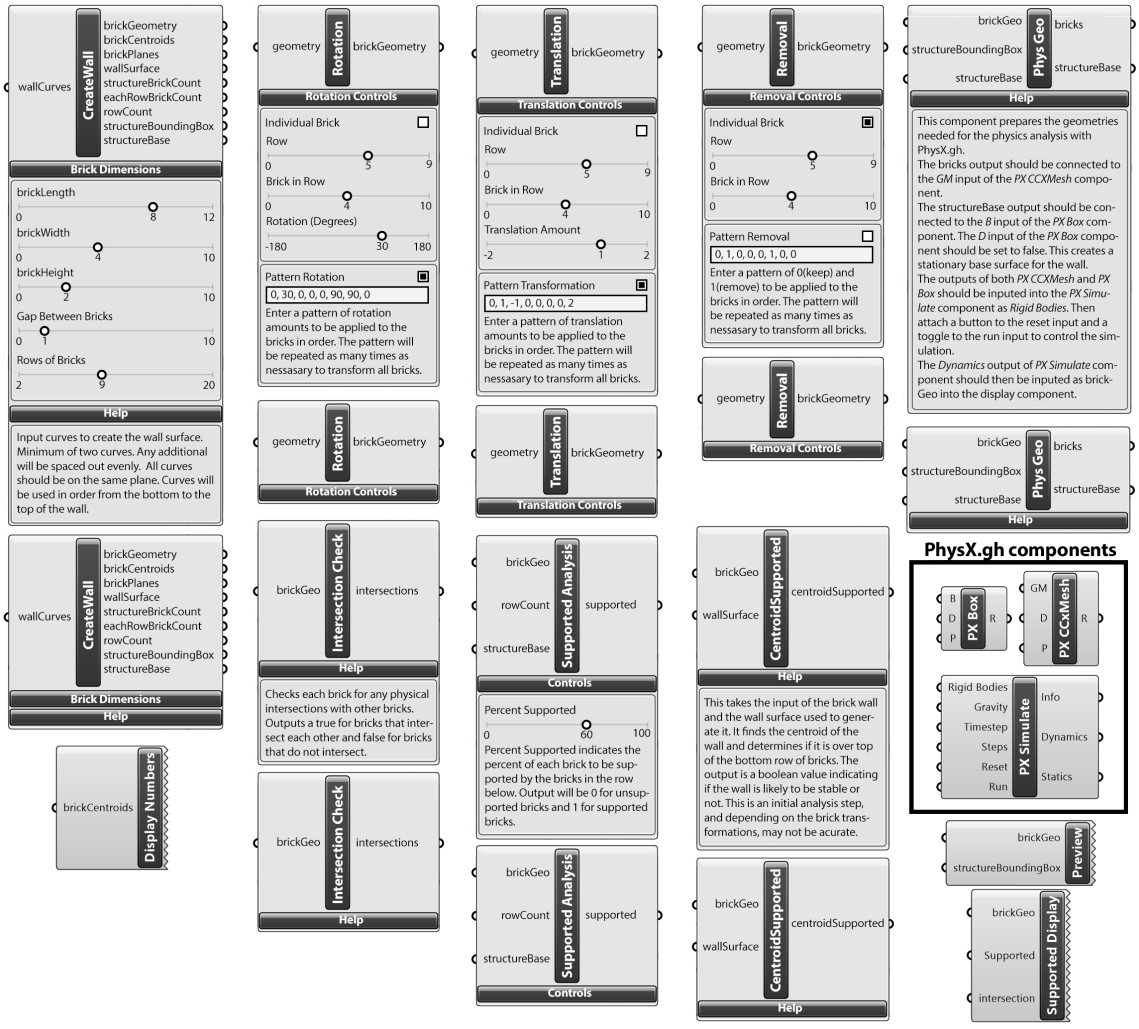

Figure D.2: The ideal components expanded to show options and help and collapsed to save space