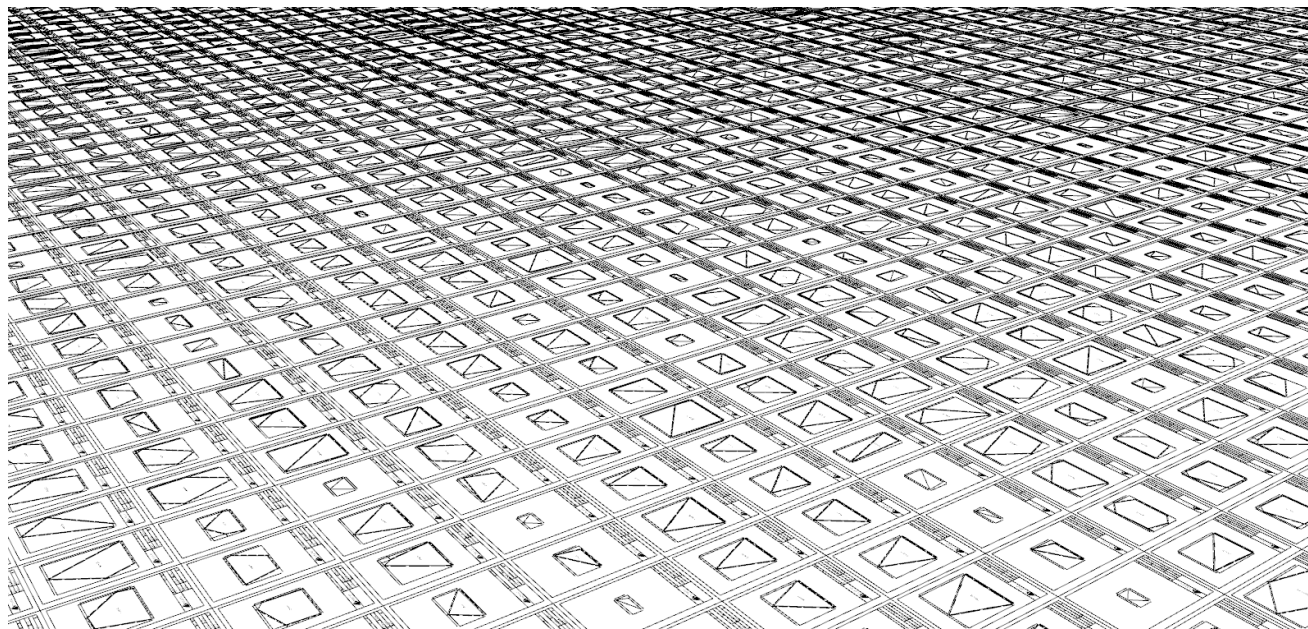




**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

---



# **Automation of the dimensioning of glass and production of glass drawings**

Master's thesis in Structural Engineering and Building Technology

**MARTIN NYGREN**

---

Department of Civil and Environmental Engineering  
*Structural Engineering*  
*Concrete Structures*  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2017  
Master's thesis BOMX02-17-49



MASTER'S THESIS BOMX02-17-49

# Automation of the dimensioning of glass and production of glass drawings

*Master's thesis in Structural Engineering and Building Technology*

MARTIN NYGREN

Department of Civil and Environmental Engineering

*Structural Engineering*

*Concrete Structures*

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2017

Automation of the dimensioning of glass and production of glass drawings

MARTIN NYGREN

© MARTIN NYGREN, 2017

Master's thesis BOMX02-17-49

ISSN 1652-8557

Department of Civil and Environmental Engineering

Structural Engineering

Concrete Structures

Chalmers University of Technology

SE-412 96 Gothenburg

Sweden

Telephone: +46 (0)31-772 1000

Cover:

A few hundred drawings displayed side by side. They were created using the tool-kit developed in this thesis.

Chalmers Reproservice

Gothenburg, Sweden 2017

# Automation of the dimensioning of glass and production of glass drawings

Master's thesis in Structural Engineering and Building Technology

MARTIN NYGREN

Department of Civil and Environmental Engineering

Structural Engineering

Concrete Structures

Chalmers University of Technology

## ABSTRACT

Dimensioning is done for a huge amount of glass pieces every year at Skandinaviska Glassystem, and drawings need to be created for every piece separately. This is currently done with a lot of manual work using design rules and other guidelines. The manual work associated with the dimensioning is extensive, tedious and time consuming and therefore also expensive. The purpose for the thesis has been to look at how this process can be made more effective.

A set of small programs/a tool-kit has been developed to automate the process. Each tool solves a small part of the process and by dividing the automation into several smaller parts the solution has become flexible and suitable for future changes and development. The software is developed to be used in the graphical algorithm editor “Grasshopper”, which runs within the “Rhinceros 3D” application. The possibility to use the tools developed in this thesis in combination with many other tools and functionalities available in “Rhinceros 3D” and “Grasshopper” has further expanded the capabilities of the tool-kit.

The software developed in this thesis can already massively reduce the time required for dimensioning of glass and production of glass drawings. There is still room for improvement though, more design rules could be considered and a proper software testing process could further improve the usefulness of the software.

Keywords: Automation, Glass façade, Streamlining, Grasshopper, Drawing production

## ACKNOWLEDGEMENTS

First and foremost I want to thank Filip Nilenius and Jakob Brusewitz for their encouragement and guidance throughout their supervision of this thesis. Filip has given a lot of support and much appreciated help with the report writing and Jakob has contributed with important feedback and insightful thoughts to support the software development in the thesis.

I also want to thank Peter Engstrand, for giving me the opportunity to collaborate with Skandinaviska Glassystem for the thesis.



# CONTENTS

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>i</b>
<b>Contents</b>	<b>iii</b>
<b>Preface</b>	<b>vii</b>
<b>Nomenclature</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem description . . . . .	1
1.3 General aim . . . . .	1
1.4 Method / Outline . . . . .	2
1.5 Limitations . . . . .	2
1.6 Objectives . . . . .	2
<b>2 Design process - from geometry to drawings</b>	<b>2</b>
2.1 Design rules . . . . .	3
<b>3 Automation</b>	<b>6</b>
3.1 Automation in general . . . . .	6
3.1.1 Principles of automation . . . . .	6
3.1.2 Types of automation . . . . .	6
3.1.3 CAD/CAM and CIM . . . . .	7
<b>CHALMERS, Civil and Environmental Engineering, Master's Thesis, BOMX02-17-49</b>	<b>iii</b>

3.1.4	Advantages and disadvantages of automation . . . . .	8
3.2	Automation in the field of building technology . . . . .	8
3.2.1	Building Information Modeling (BIM) . . . . .	8
3.3	Automation of the design process - from geometry to drawings . . . . .	8
<b>4</b>	<b>Implementation of an automated design process</b>	<b>9</b>
4.1	Principles of a expandable and adjustable system . . . . .	9
4.2	Quantify design rules . . . . .	9
4.3	Object oriented design . . . . .	10
4.4	Software . . . . .	10
4.4.1	Rhinoceros . . . . .	10
4.4.2	Grasshopper . . . . .	11
<b>5</b>	<b>Results</b>	<b>12</b>
5.1	The tool-kit . . . . .	12
5.1.1	Component categories . . . . .	12
5.1.2	Geometry components . . . . .	13
5.1.3	Glass components . . . . .	16
5.1.4	Template components . . . . .	19
5.1.5	Drawing components . . . . .	22
5.2	Intended practice . . . . .	24
5.3	Ease of use and flexibility . . . . .	25
5.4	Test case - The Scandic Victoria Tower . . . . .	26
5.4.1	Working hours . . . . .	27
5.4.2	Output and drawing quality . . . . .	28



<b>6 Discussion</b>	<b>32</b>
6.1 Developing for Rhinoceros/Grasshopper - Advantages and disadvantages . . . . .	32
6.2 Possible future improvements and development . . . . .	33
<b>7 Conclusions</b>	<b>33</b>
7.1 Suggested further development . . . . .	34
<b>References</b>	<b>35</b>
<b>Appendix A Appendix A</b>	<b>35</b>
A.1 Intended practice for the software - an example . . . . .	35
A.1.1 Step 1 . . . . .	36
A.1.2 Step 2 . . . . .	38
A.1.3 Step 3 . . . . .	39
A.1.4 Step 4 . . . . .	40
A.1.5 Step 5 . . . . .	42
A.1.6 Step 6 . . . . .	43



## PREFACE

This Master's Thesis comprises 30 credits and has been carried out during the spring of 2017. The work has taken place at the office for, and in collaboration with, Skandinaviska Glassystem in Gothenburg.

Supervisors have been Filip Nilenius, assistant professor working at the division of Structural Engineering at Chalmers, and Jakob Brusewitz, working at Skandinaviska Glassystem. Filip Nilenius has also assisted as the examiner for the thesis.



# NOMENCLATURE

## Abbreviations

BIM	Building Information Modeling
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CIM	Computer-Integrated Manufacturing
DXF	Drawing Exchange Format
GUI	Graphical User Interface
NURBS	Non-Uniform Rational Basis Spline
PDF	Portable Document Format
SDK	Software Development Kit



# 1 Introduction

## 1.1 Background

Skandinaviska Glassystem is a company based in Gothenburg working with advanced building enveloped in Scandinavia. They develop, market and assemble their own patented glass systems and many of the projects that they work with include huge amounts of glass pieces. Dimensioning is done for all of these glass pieces and drawings needs to be done for every piece separately. This is currently done with a lot of manual work using different design rules and other guidelines depending on the system that the glass piece should be used in. Carrying system, wind loads, geometry and openings are a few examples of what needs to be considered in the design. The manual work associated with the dimensioning is extensive and time consuming and therefore also expensive. There is potentially a lot of money to save by making this process more effective.

## 1.2 Problem description

Today the glass drawings at Skandinaviska Glassystem are all created manually, where the the basic shapes (boundary lines) often are based on a 3D-model of a façade provided by an architectural office. The drawing must contain information about, among other things, the overall dimensions, fixings, silicon depth, scale, weight and current project manager. The fixings should be located at certain distances between each other and are also of different kinds depending on the system/systems used in the project. This is done according to design rules provided in an internal handbook at Skandinaviska Glassystem. The weight depends almost solely on the amount of glass used and can be calculated using information about the area of the window and the different layers used in the construction. In most cases this process (explained in more detail in chapter 2) including the dimensioning and creation of drawings is relatively straight forward but still takes a lot of time using the current methods and work flow.

## 1.3 General aim

The purpose of the thesis is to explore how this design process can be made more effective. A lot of time and money can be saved if the manual work can be streamlined or perhaps avoided all together. With a more effective process the accuracy of the final drawings can potentially be improved and the staff at Skandinaviska Glassystem can hopefully focus on the more qualitative and difficult tasks.

## 1.4 Method / Outline

The relatively straight forward work flow and methods used to create these drawings are well suited for automation. The process can be divided into several smaller tasks that needs to be performed according to certain rules. In this thesis an attempt has been made to quantify the design rules in a way suitable for automation. A computer program has been developed to automate as much as possible of this process. It is not a single tool created to solve the hole design (“a black box”) but rather a set of tools to solve different steps in the design process separately.

## 1.5 Limitations

The whole process is in a way essentially management and translation of information. The geometry and choice of system and other initial choices will lead to a certain design and kind of drawing. The different choices and corresponding design rules that could be considered in a design like this at Skandinaviska Glassystem is however very extensive. The focus in this thesis has been to cover the most common cases and to create a good basis for further development.

## 1.6 Objectives

The aim is to automate the design process by creating a flexible, user friendly system/tool kit that can easily be modified and developed further. It should, with necessary input, be possible to produce finished glass drawings as per design rules and existing drawing templates. The project can easily be enlarged if time is given; more design rules can be considered and refined and additional outputs can be created.

# 2 Design process - from geometry to drawings

The starting point of the design process at Skandinaviska Glassystem is often the receiving of a 3D-model from an architectural office. The intended design and shapes of the glass pieces and the façades are generally well illustrated. However, if a model is examined in detail it can often be seen that it's a poorly modelled and structured model. The building blocks, that is lines, surfaces, meshes and other sorts of geometry, are often partly overlapping, trimmed, duplicates and intersecting (see figure A.2b in appendix A for an example). The first thing that needs to be done is to redraw the shapes of all glass pieces manually, this is often done using the computer program “AutoCAD”. The boundary line (also called the module line) of every glass piece is drawn separately in 3d-space and then defined as a so called “block”. A block is a collection of objects that are combined into a single named object. Several blocks of the same kind (or rather static references to the specified block) can be used at once and if the definition of



the block is changed all of the blocks that are used will be updated/changed accordingly and automatically. Static references to all blocks will be inserted on the XY-plane (plane-to-plane projections) and put into template blocks containing text fields for things like date, weight and project manager to be filled in manually. The next step is to measure the length of the segments, the glass edges, and distribute fixings at recommended positions according to the in house design handbook. The needed silicone depth along the edges (also according to design rules) needs to be specified as well. When all fixings are drawn with required annotations and all text fields are filled the only thing left is to print all drawings to pdf-files.

## 2.1 Design rules

The design rules used by Skandinaviska Glassystem are confidential and can therefore not be presented in detail here. Still, some of the general principles without any actual numbers will be presented in this chapter for the reader to be able to understand what sort of rules that needs to be considered.

There are several different systems used to fix the glass pieces to the load bearing parts of the façade/building. They all have different names and type of glass supports, and different design rules are to be followed depending on what system that is used. In some projects a mix of different systems can be used, even on the same glass piece. Some examples of different glass fixing systems are “Plastic pocket”, “Light clip”, “Insert 45” and “CT”.

One thing that is different from system to system is the required silicone depth. A drawing showing the cross section of a glass edge, where the silicon depth is displayed, can be seen in figure 2.1 and figure 2.2 shows how the required silicon depth vary depending on what glass fixing system that is used along a glass edge.

Other parameters governing the design is the distances between fixings and distances between fixings and the edge ends (illustrated in figure 2.3). For some fixing systems it also exists design rules concerning adjacent fixings, that is rules that has to do with the distances between fixings attached to different glass pieces close to each other (as seen in figure 2.4).

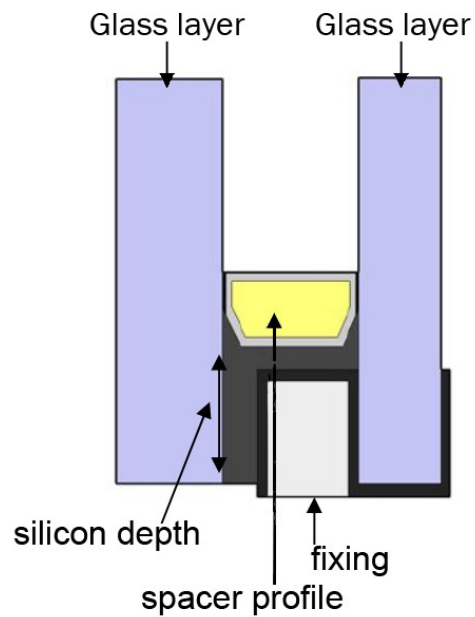


Figure 2.1: A cross section of a glass piece. The silicon depth, a fixing, two layers of glass and a spacer profile can be seen in the figure. Figure is adapted from a figure found in the inhouse handbook for glass dimensioning at Skandinaviska Glassystem by Wadman and Hansen 2012.

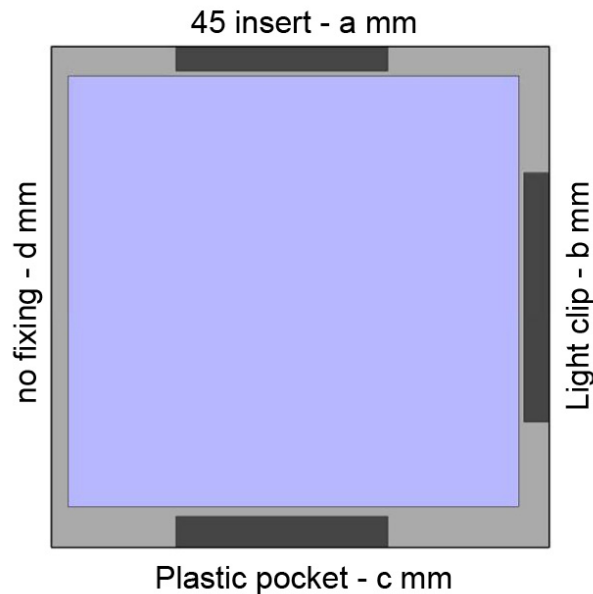


Figure 2.2: Different silicon depth is required for different fixing systems. The figure is adapted from a figure found in the inhouse handbook for glass dimensioning at Skandinaviska Glassystem by Wadman and Hansen 2012.

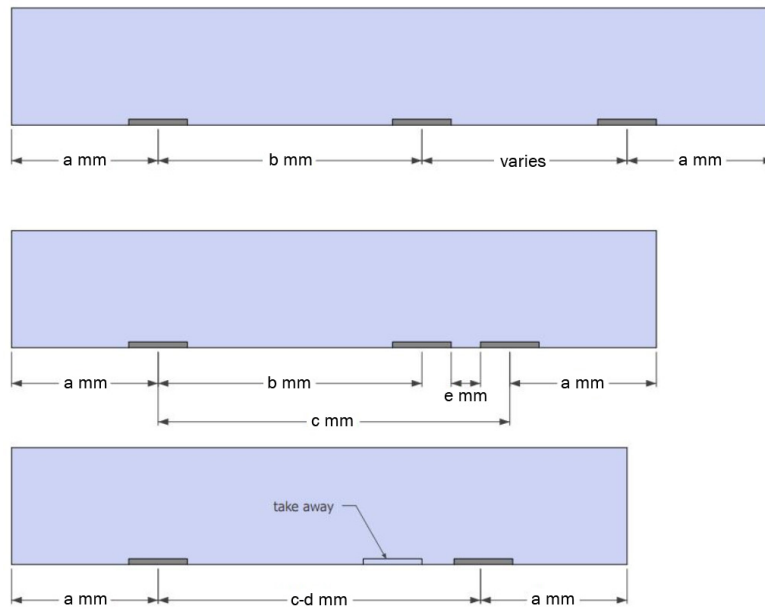


Figure 2.3: Example of fixings placement rules (this example is for the insert 45 system). Figure is adapted from a figure found in the inhouse handbook for glass dimensioning at Skandinaviska Glassystem by Wadman and Hansen 2012.

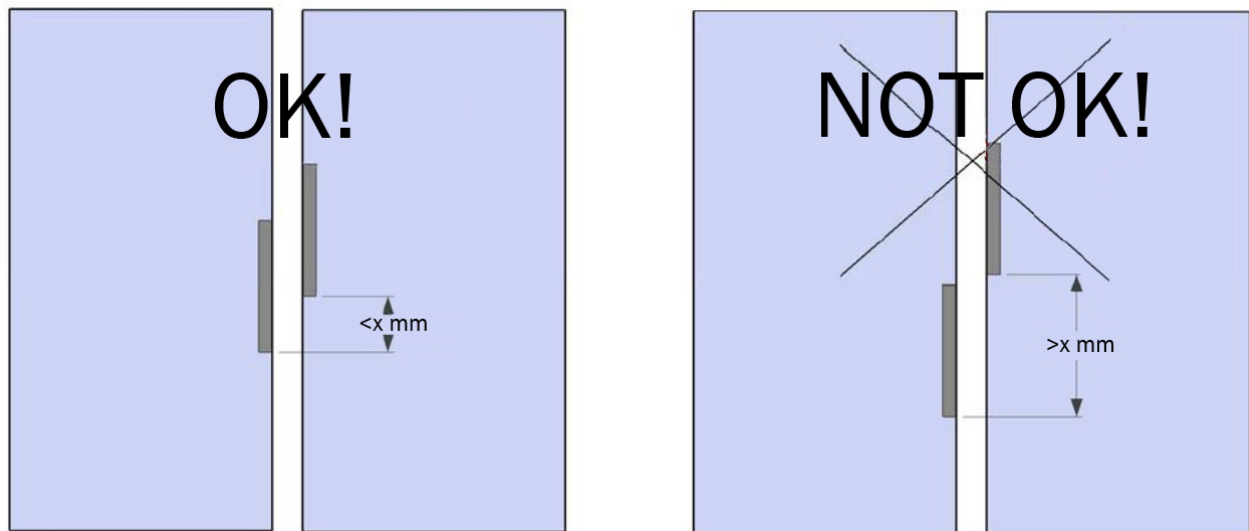


Figure 2.4: Example of design rule for adjacent fixings (this example is for the SG 45/60-system). The figure is adapted from a figure found in the inhouse handbook for glass dimensioning at Skandinaviska Glassystem by Wadman and Hansen 2012.

# 3 Automation

## 3.1 Automation in general

Automation is the application of machines to tasks that would otherwise be performed by human beings. The term automation is used to describe the integration of machines into some kind of self-governing system while mechanization is the term that refer to the simple replacement of human labour by machines (Groover 2017).

### 3.1.1 Principles of automation

An automated system will, almost always, consist of three basic building blocks: a source of power, feedback controls and machine programming.

The source of power, most commonly electricity, is needed to accomplish the actions performed by the system. The actions are generally some kind of processing or transfer and positioning. The processing may involve for example the moulding of plastic, the shaping of metal or the processing of data. In these actions the energy is used to transfer the entity from one state to a more valuable state. Transferring and positioning is used to move the entities between different processes in an accurate way.

A feedback control system consists of five basic components: The input, the process being controlled, the output, sensing element and controller devices. The sensing elements are used to monitor the values of the output variables and the controller is the mechanism used to make changes in the process to influence the output variable.

The machine programming is used to determine the set of actions that is to be performed by the system. Some of the actions may be executed without need for a feedback loop to verify that the command has been properly carried out. Other actions require feedback control to ensure the quality of the output, this is for example often the case when there are variations in the raw materials being fed into a production process. Programmable machines often have some sort of decision-making capacity which makes them respond differently depending on the circumstances. Several potential benefits can be obtained by providing an automated system with this capacity, including safety monitoring, interaction with humans, error detection and recovery and process optimization (Groover 2017).

### 3.1.2 Types of automation

There are three different types of automation in production, fixed automation, programmable automation and flexible automation.

Fixed automation is also known as hard automation and refers to when the equipment configuration determines the sequence of processing operations. The programmed commands are contained in the machines in the form of wiring, cams and other hardware which might not be easily changed. This kind of automation often means high initial investments but also high production rates.

Programmable automation is often used when the products are produced in batches. The system needs to be reprogrammed and changed between every batch or new product and this means periods of non-productive time. The equipment used in the process are in general designed to facilitate product changeover rather than for product specialization and because of this the production rate is usually lower than for fixed automation

Flexible automation is similar to programmable automation except that the variety of products is sufficiently limited. This makes it possible to make the changeover of the equipment can be done automatically. The reprogramming needed is done off-line at a computer terminal without using the production equipment itself. It is not necessary to group identical products into batches (Groover 2017).

### **3.1.3 CAD/CAM and CIM**

Computer-aided design (CAD) and computer-aided manufacturing (CAM), has been used increasingly since about 1970. This technology is based on the capacity of a computer system to process, store and display data. It involves automation of elements in the entire design-and-manufacturing process.

CAD can be used to assist in the creation, modification, analysis and optimization of a design. The designer can create an object composed of lines and surfaces and store the design in the computer database. With the appropriate CAD software, various analysis can be done on the object and detailed drawings can be generated with ease.

CAM can be used to assist the planning, control and management of production operations. This can be done either with direct or indirect connection with the production operations. Direct connection means that data from the production process is collected analysed and the process-performance results are communicated to production management. Indirect connection involves applications in which the production process is supported without actually monitoring or controlling them. These applications include management functions that can be performed by the computer or by humans working with the computer more efficiently than by humans alone.

Computer-integrated manufacturing (CIM), includes all the engineering functions of CAD/CAM but also the business aspect of the firm. This includes for example order entry, cost accounting, employee time records and payroll and customer billing. CIM represents the highest level of automation in manufacturing (Groover 2017).

### **3.1.4 Advantages and disadvantages of automation**

There are many advantages commonly attributed to automation including increased productivity, higher output, more efficient use of materials, better and consistent quality of products, improved safety, shorter work weeks for labour and greater control. A few of the disadvantages associated with automation is the risk of worker displacement, high capital expenditure required, higher level of maintenance needed compared to manually operated machines and generally lower degree of flexibility (Lamb 2013).

## **3.2 Automation in the field of building technology**

The construction industry hasn't kept pace with many other industries when it comes to automation and technological improvements over the last decades. This has created a fertile ground for new tools and products that offer better ways of doing things (McCool and Hardin 2015).

### **3.2.1 Building Information Modeling (BIM)**

Building Information Modeling (BIM) has had a big influence on the industry. Modes of design practice and standards of building design, delivery and operation have been, and will most probably continue to be, fundamentally transformed (Kensek and Noble 2014).

According to the US Building Information Model Standard Project Committee BIM is a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition (Building Sciences 2016).

BIM is not only a 3D tool but also an information-rich database that links to the model components. This is often referred to as parametric modeling. The ultimate value of BIM lies in the ability to aggregate, edit, sort and put together this information to drive at better answers to different design and construction questions (McCool and Hardin 2015).

## **3.3 Automation of the design process - from geometry to drawings**

The design process for the production of glass drawings can be described as a series of information translations. The initial intended shape of the geometry will together with a number of choices, regarding for example fixing systems and glass compilations, lead to the final drawings displaying those choices and that geometry in a different way. This time suited for production and in better compilation. This kind of problem is well suited for automation.

## **4 Implementation of an automated design process**

In this thesis an attempt has been made to automate the design process by creating a flexible and user friendly tool kit/system that can easily be modified and developed further. It has been important to make it flexible for several reasons. A flexible system can be utilized even if the tasks to be solved changes to some extent from project to project. It is also important for the user to be able to make manual changes and modifications when the system doesn't consider something. There are too many different rules, special cases, intuitive decisions and changing components to automate it all completely in a program developed within this master thesis.

### **4.1 Principles of a expandable and adjustable system**

System flexibility is often considered a requirement for firms but the investment cost required is often high which can jeopardize the profitability. It is key to find a good level of flexibility. If a system offers more flexibility than what is needed the investment and operative costs will be unnecessarily high. A flexible system is often harder to and takes more time to operate due to the increased number of choices and settings that needs to be done along the production. A dedicated system however, would not be able to adapt to changes of the product or production method. The flexibility level of the system should be related to its ability to cope with volume, input and technological changes, both present and future ones. The level of flexibility very much impacts on the architecture of the system and often leads to hybrid systems, i.e. automated integrated systems in which parts can be processed by both general and dedicated processes (Tolio 2009).

### **4.2 Quantify design rules**

The rules for dimensioning and design are not always suitable for a computer program. Recommendations and requirements stated in sentences and images in design books and manuals needs to be translated into measurable and comparable variables to govern the output of the computer program. This is the first challenge in creating a computer program to handle the design process. Questions that needs to be answered are “What kind of inputs are appropriate and can provided the needed information?” and “In what way can a structure be created to always fulfil the requirements?”. The rules that must be fulfilled does often not directly reveal how the output can be created but rather describes how the output must look. In essence; the qualitative design rules needs to be quantified in order to be able to be implemented in a computer program.

## 4.3 Object oriented design

A choice that was made in this master thesis was to use object-oriented design as much as possible in the creation of the tool kit. This means that the objects are the centrepieces of the software design. This kind of design makes it possible to re-use a lot of the system when changes need to be done. This can be compared to design centred around processes, where often very little of the old system can be re-used when the system needs to change. To every kind of object there are associated data and operations (or methods) that could be used to modify it. An abstraction of an object can be created and stable regardless of the changing requirements on the application. The objects need to be able to store the data needed and provide all the necessary operations. The system can then be “assembled” from the objects. A functional specification can in a systematic way be translated into a conceptual design (Dathan and Ramnath 2015).

## 4.4 Software

In this master thesis a kit of components have been created to be used in the program grasshopper (described more in detail in the following chapter). Another possible choice worth mentioning would have been to develop components in a similar way but for the program called “Dynamo”. Dynamo has the advantage that it is completely compatible with “Revit”, which is a powerful BIM program with a lot of possibly useful functionality. Dynamo is however, as a relatively new program, unfortunately still lacking in functionality, documentation, community, stability and ease of use in comparison to grasshopper.

### 4.4.1 Rhinoceros

Rhinoceros (Rhino) is a commercial software application used in processes of computer-aided design (CAD). The geometry used in Rhinoceros is based on the Non-Uniform Rational Basis Spline (NURBS) mathematical model to get as mathematically precise representations as possible of curves and free form surfaces in computer graphics. The software is widely used in several different industries such as architecture, industrial design, product design, multimedia and graphic design (LLC 2017). Among a lot of other functionalities and features, the program supports the use of “blocks” (which has been described a bit more in chapter 2) which will be utilized in this thesis.

Rhinoceros’ open Software Development Kit (SDK) and application architecture makes it modular and makes it easy for the user to customize the interface and create custom components and menus. This has resulted in dozens of plug-ins that can expand Rhinoceros’ capabilities in fields such as architecture, engineering and rendering (McNeel and Associates 2017).





## 5 Results

The result of the implementation will be presented in this chapter. A tool-kit/a set of Grasshopper components have been developed to automate different steps in the process of the dimensioning of glass and the production of glass drawings.

### 5.1 The tool-kit

In this chapter the software developed during this master thesis will be presented.

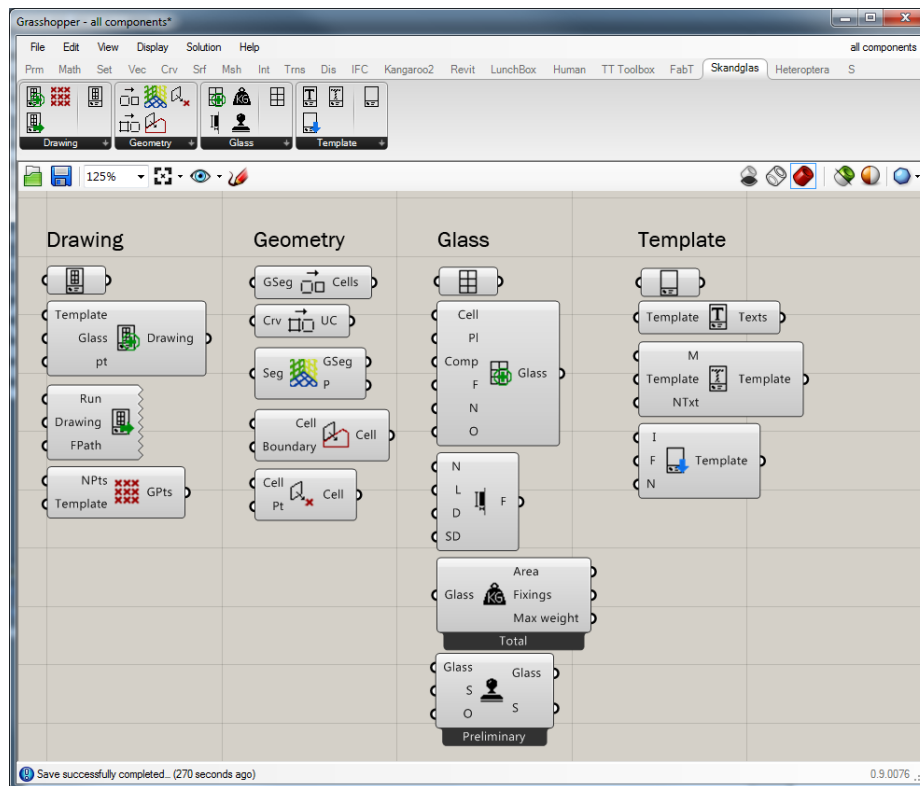


Figure 5.1: The developed tool-kit with all its components displayed in the Grasshopper GUI.

#### 5.1.1 Component categories

There are in total eighteen different grasshopper components in the tool-kit. They are meant to be used together with each other and also together with already existing grasshopper components. The components are sorted into 4 different categories; Geometry, Glass, Template, and Drawing. All components are briefly described in the sub-chapters below.

### 5.1.2 Geometry components

The geometry components, as the name suggests, process geometry in different ways. Before a glass can be created, a closed curve is needed to represent the glass boundary (also called a “cell” in this thesis). The geometry source, often a 3D-model provided by an architect firm, rarely provides this. The geometry components are meant to make the process of creating these closed curves from the provided geometry much easier.

#### Clean geometry

This component will in different ways try to extract curve segments suitable to create cells with. The input is a set of curves and they might be overlapping, kinked and perhaps duplicates of the same curve on top of each other. The component will split all curves in intersection points and discontinuities, remove duplicates and try to simplify curves (removing control points that does not affect the shape of the curve). It will also remove “loose ends”, that means removing segments of which at least one of the end points are not connected to any other segment.



Figure 5.2: The component “Clean geometry”.

#### Group segments

The component “Group segments” will group segments in a way so that the output is suitable to use as input to the component “Build cells”. All segments sorted in the same group are connected to each other (not necessarily directly connected) and lies in the same plane. The algorithm used to sort the segments is briefly described in Algorithm 1. The output of the component is both the groups of segments that are defined and the corresponding planes.

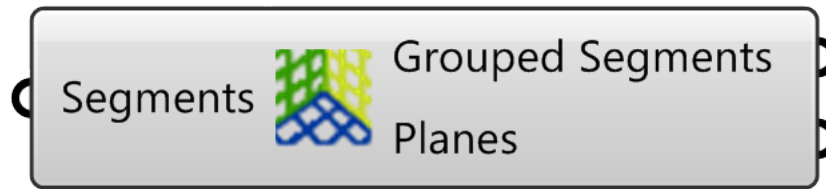


Figure 5.3: The component “Group segments”.

**Data:** arbitrary segments

**Result:** segments sorted by plane and cluster

```

foreach segment in arbitrary segments do
  if segments is not flagged as part of a cluster then
    flag this segment as part of a new cluster;
    find all connected segments (recursive function);
    flag all connected segments as part of the cluster;
  end
end
foreach cluster do
  foreach segment in cluster do
    if segment is not flagged as part of a plane group then
      find directly connected segments that somehow deviate in angle from the original
      segment and are not flagged as part of a plane group;
      try to find a plane defined so that at least the current segment and one of the found
      segments lies in it;
      find all segments that lies in this plane and store this information;
    end
  end
end

```

**Algorithm 1:** The algorithm used to sort segments into groups by plane and connectivity.

## Build cells

The segments/curves provided as input to this component will be used to create cells, that is closed planar curves that can be used as input to the component Create glass. The component defines a connected network based on how the segments are connected and then uses a right handed turning detection to traverse through and find cells.



Figure 5.4: The component “Build cells”.

### Orient cell components

A cell is, as already mentioned, a planar closed curve that can be used as one of the inputs to the component “Create glass”. When a glass is created, the direction of the glass plane normal needs to be defined. Using the tool-kit developed during this thesis, there are a few ways of defining this. One of them is to use the direction of the cell curve. The cell curve consists of a series of segments of which the first two can be used to define the x and y-axis of a plane. The cross-product of vectors along these axis can then be used to get the direction of the normal vector. If the orientation of the cell curve is reversed, the normal of the plane created from the cell will be flipped.

The orient components can be a help for the user to orient cells as desired. The “Orient cell (point)” component has a point input and the cells provided as input will be flipped so that the normal of a plane created from a specific cell will match the direction of a vector from the point to the cell as much as possible. This means that a point located inside a 3d-model of a house will most of the time make all cells get the correct direction. If not, the component “orient cell (boundary)” might do the job. This component uses a slightly more sophisticated method to orient the cells. The input “boundary” is suppose to be a curve representing the footprint of a building.

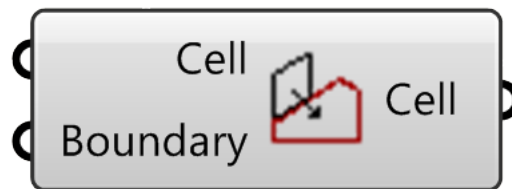


Figure 5.5: The component “Orient cell (boundary)”.

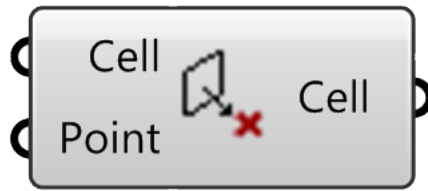


Figure 5.6: The component “Orient cell (point)”.

### 5.1.3 Glass components

A “Glass” is a defined data type in the software which contains all sorts of information about the glass. This information includes the boundary curve geometry, information about fixings, glass composition, glass name, glass plane, glass status and weight of the glass. All components in the glass category has to do with the storing, information extraction, creation and modification of glasses.

#### Glass parameter

The glass parameter contains the data type “Glass” and makes it possible to save and load glass objects from a grasshopper file. It also provides the possibility to preview glass objects and to “bake” them. When a glass is baked there will be a block created in Rhino representing the glass. The glass block, defined in rhino, will contain all the information that was stored in the glass object in grasshopper when the baking took place. The geometry of the block can also be modified, using all the tools available in Rhino, to later be read back into a glass parameter in grasshopper.



Figure 5.7: The parameter “glass”.

#### Create glass

The “Create glass” component is used to create a glass object from scratch. There are a bunch of inputs of which the only one required for the component to work is the one called “cell”, that is the closed curve representing the outer boundary of the glass. If no other inputs are provided the component will create a glass object anyway, using default settings for everything else that needs to be specified in order to create a glass.

Another input is the “glass plane”, by providing this input the user can decide the orientation of the glass and also what side of the glass that is inside and what is outside. If this is not provided a plane will be created from the cell and then rotated so that the plane y-axis will be aligned with the world z-axis (the z-axis of the coordinate system of the 3d-model) as much as possible.

The glass composition can be defined using the “Composition” input. This is essentially a text input where the composition of the glass, the different layers and their thickness, are specified according to a set rules. This information can later be used to, for example, calculate weight of the glass.

The fourth input of the component is called “Fixing”. Different types of fixings can potentially have different sizes, shapes and different rules for dimensioning associated to them. One function available is that the user can connect a “Value list” component (an original grasshopper component) to this input and it will automatically be populated with all the predefined and common fixing types available.

The second last input can be used to specify the name of the glass, if the automatically generated name is not desirable, and the last input can be used to offset the cell curve provided. If so, the offset of the curve will be used as the glass boundary instead of the one provided as the “cell” input.

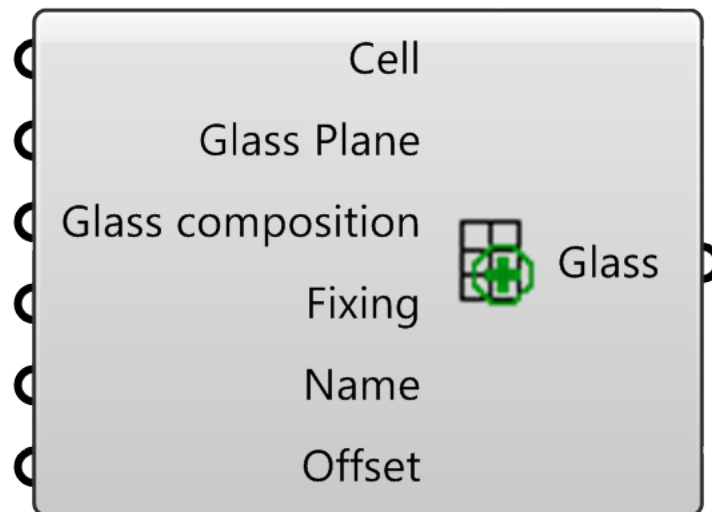


Figure 5.8: The component “Create glass”.

### Custom fixing

If the type of fixing that is going to be used for a glass can not be found among the predefined and common fixing types this component could be of help. The user can, by specifying a name, length, depth and a needed silicone depth, define a custom fixing. This custom fixing can then be used as the “Fixing” input of the “Create glass” component.

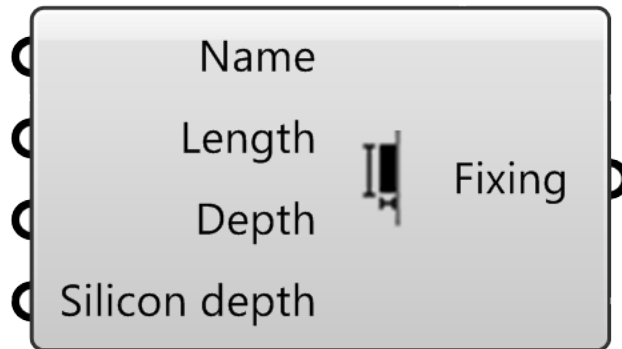


Figure 5.9: The component “Custom fixing”.

## Quantities

The “Quantities” component can be used to examine glass pieces provided as input, either unitarily or as a group (per list). The output values are the max weight, total area and information about the fixings. The information about what kind of fixings that are used in a project and how many there are of each kind will be useful when making orders for goods in preparation for the construction phase of a project.

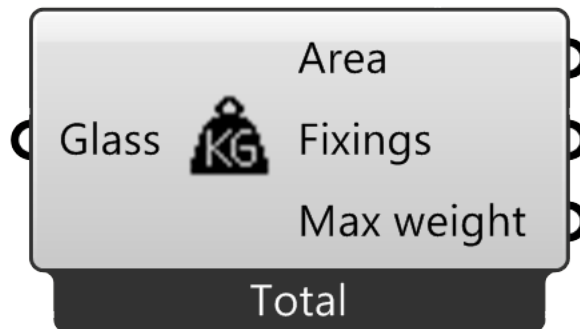


Figure 5.10: The component “Quantities”.

## Status

The status of a glass can be changed using the “Status” component. The status is something that is always stated in the glass drawing and could be for example “Preliminary” or “Building plot”. The user can also choose if the status of a block definition in the Rhino model, corresponding to the glass given as input,



should be changed as well.



Figure 5.11: The component “Status”.

#### 5.1.4 Template components

A “Template” is a data type in the software which can contain geometry (for example a logo and some text boxes) and text that the user wants to include in a drawing. The template components are used to create, store, import and modify template objects.

##### Template parameter

The template parameter contains the parameter “Template” and makes it possible to save and load template objects from a grasshopper file. It is also possible to both preview and “bake” a template object. When a template is baked there will be a block created in Rhino representing the template. The geometry of the block can be modified, using all the tools available in Rhino, to later be read back into a template parameter in grasshopper. Any rhino block can be used to define a template but to be able to later create a proper drawing using it, the user can choose to name some of the geometry included to manipulate how the template will be used. If a curve included in the template block is named “drawing boundary” this curve will, when the drawing later is created, be used as a guide for where in the template the glass drawing and annotations should be placed. The text of a text object named “date” will be exchanged to the date the drawing is created, the text of a text object named “scale” will be exchanged with a text displaying the drawing scale. The same kind of technique can also be used to display glass weight, status and drawing numbers in the drawings.



Figure 5.12: The parameter “Template”.

## Import template

It makes sense to use the same kind of template in many different projects, and also to avoid recreating a template over and over. With the “import template” component the user can import a template (by importing a rhino block definition) from any rhino file on the computer. The file path to the Rhino-file (.3dm) and name of the block intended to be imported are required as inputs to the component. During this thesis a few default templates was modelled and saved in a Rhino-file named “SkandGlasTemplates.3dm”. The templates created in this thesis was adapted from the current default drawing templates used at Skandinaviska Glassystem.

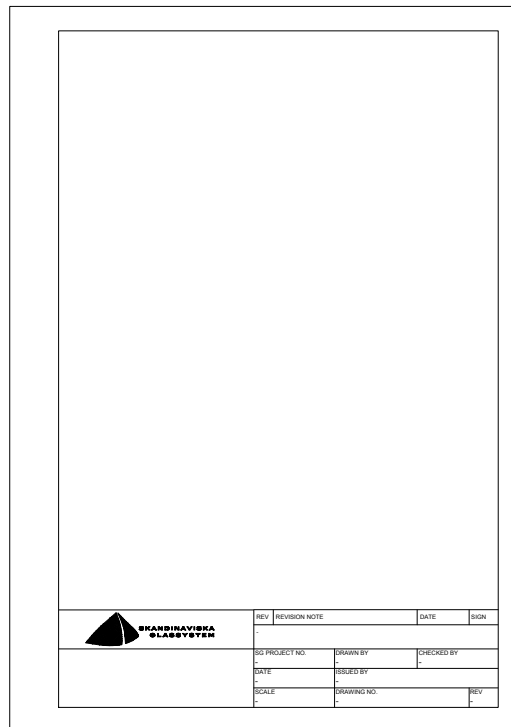


Figure 5.13: Example of a template. This template is adopted from the most commonly used template at Skandinaviska glassystem.

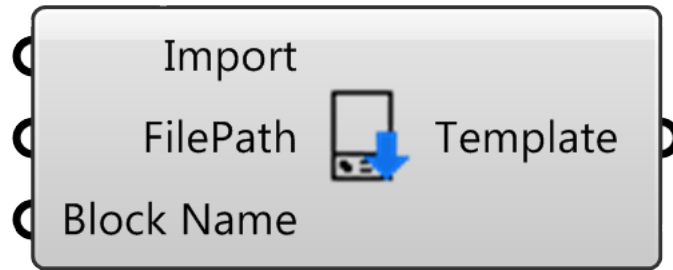


Figure 5.14: The component “Import template”.

### Extract template texts and Modify template texts

Some of the template texts needs to be project specific or maybe specific for a set of drawings. It would be possible but not practical for the user to manually edit templates to achieve this. The components “Extract template texts” and “Modify template texts” were developed to make this easy. The component “Extract template texts” will find all named texts objects within a template provided as input. The same template can be provided to the component “Modify template texts” together with new texts to replace the old ones. The old text objects will not be removed or change name, but the content of the text objects (the texts) will be replaced.



Figure 5.15: The component “Extract template texts”.



Figure 5.16: The component “Modify template texts”.

### 5.1.5 Drawing components

A “Drawing” is a data type in the software which contains geometry adapted from a template, a scaled version of a glass and annotations of different kinds. The drawing components are used to create, store, preview and export drawing objects.

#### Drawing parameter

The drawing parameter contains the parameter “Drawing” and makes it possible to save and load drawing objects from a grasshopper file. It is also possible to both preview and “bake” a drawing object. When a drawing is baked there will be a block created in Rhino representing it. The geometry of the block can be modified, using all the tools available in Rhino, to later be read back into a drawing parameter in grasshopper.



Figure 5.17: The parameter “Drawing”.

#### Create drawing

The component “Create drawing” can be used to create a drawing using a glass object and a template object. The component will place a scaled version of the glass object in the template, update some template texts (texts specifying date, scale, weight and drawing number) and add annotations to describe the glass geometry. There is also an input to the component called “Base point”. The base point is where the drawing will be previewed and baked.

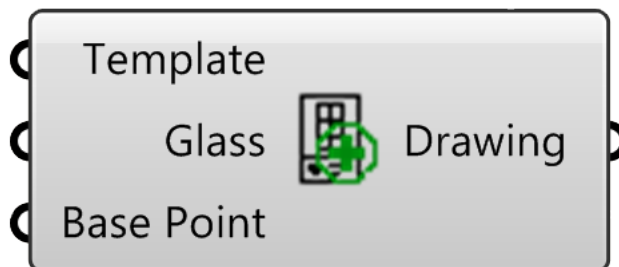


Figure 5.18: The component “Create drawing”.

## Grid points

The user might want to check the drawings to see that they are correct before they are exported to another format. One way of doing this is to preview them side by side in the Rhino window. This can be done by providing a grid of points to the input “Base point” which is an input to the component “Create drawing”. The component “Grid points” can be used to, in an easy way, create a suitable grid of points. The inputs to the component is a template object and the desired number of points. The component will check the height and width of the template object and then create a grid of points based on the measurements. See figure 5.20 for an example of a preview of multiple drawings in a Rhino viewport.



Figure 5.19: The component “Grid points”.

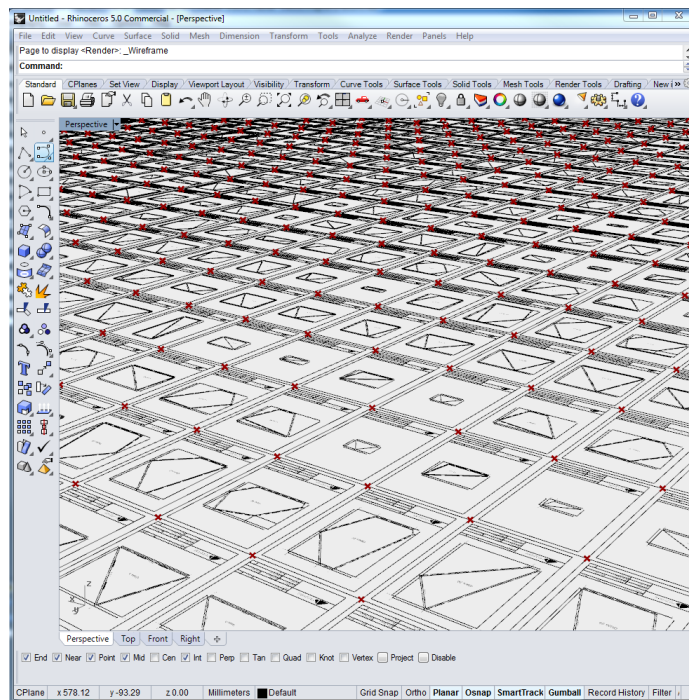


Figure 5.20: Multiple drawings previewed side by side in a Rhino viewport.

## Export drawing

A drawing can be exported using the “Export drawing” component. There are three different inputs to the component; “Run”, “Drawing” and “File Path”. If the component is set to “Run”, using the first input on the component, it will bake the drawing provided as input, select the rhino block created in Rhino, send a command to Rhino to export the selected geometry and finally it will delete the block. The input “File Path” lets the user choose where the exported file will be saved, what it should be named and what file format it should be exported as (for example .dxf, .dwg, .pdf, .ai or .3dm). If the drawing should be exported as a PDF, this is something of a special case. Rhino does not provide any integrated pdf-printer, this means that to make it possible to print the drawings to PDF an independent pdf-printer is used. The component will send a command to Rhino to in turn send a command to the printer to print the drawing. A printer called “Foxit Reader PDF Printer” is used. A bunch of different pdf-printers have been tested during this thesis but this one was chosen due to the useful settings available; option to bypass menus when printing, possibility to predefine a default save location (“Target location”) and an option to add a numeric suffix to a file name if a file with a certain name already exists.

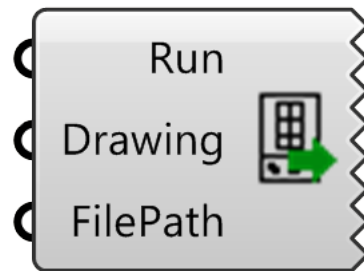


Figure 5.21: The component “Export drawing”.

## 5.2 Intended practice

There is not just one single way of using the components in the tool-kit. They are all helping the user with some specific part in the process of creating the desired drawings or extracting the desired information about the glass objects. The process will be a bit different depending on for example the geometry provided in the 3d-model, the amount of manual modifications needed and what the desired output looks like. Algorithm 2 briefly describes what the different steps in such a process could include and figure A.1 shows what the different steps could look like using the developed grasshopper components. A more detailed example of how the software can be used can be found in Appendix A. The output in this example are finished drawings printed to PDF. In another case the user might just want to know how many fixings and of what kinds that is needed in a project. If so, only step 1-3 in algorithm ?? needs to be followed and then the component “Quantities” can be used to extract the information needed about weight, area and fixings from the glass objects created.

**Data:** A 3d-model (.3dm) containing curves or other geometry to outline glass pieces or module lines

**Result:** Drawings (one per glass piece) printed in PDF

1. Use original grasshopper components, “Clean geometry”, “Group segments” and “Build cells” to create closed planar curves outlining every glass piece or boundary line.
2. Use an “Orient Cell” component to define what is inside and outside.
3. Create glass objects using the component “Create glass”. Provide glass specific information as input. Update glass status using the “Status” component if needed. 4. Create a new or modify an old template to be used for the drawings. Update template specific (most often project specific) information using the template components.
5. Create drawings (one per glass object input) using the “Create drawing” component. Glass objects and a template is needed as input. Provide base points (one per drawing) if there is a need to preview the drawing before printing them.
6. Print drawings using the component “Export drawing”.

**Algorithm 2:** An algorithm describing the process intended to create drawings, from a 3d-model in rhinoceros to drawings printed in PDF, using the tool-kit developed in this thesis.

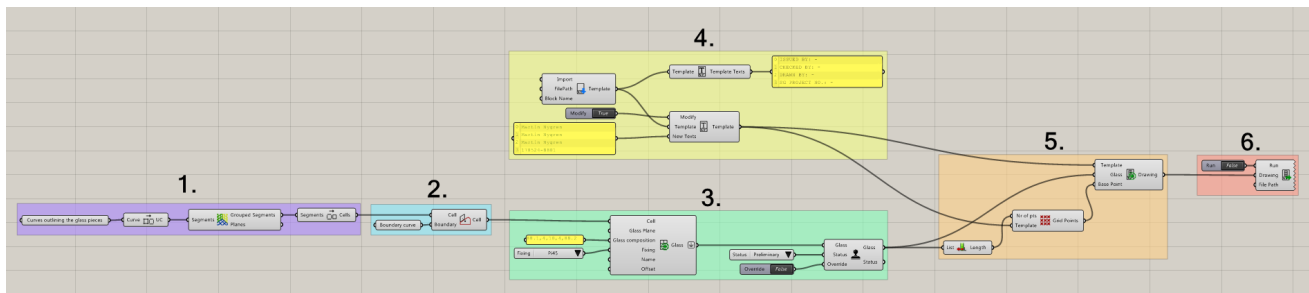


Figure 5.22: Showing an example of what the grasshopper definition could look like when the developed tool-kit is used for the whole process. The different parts of the definition is grouped and numbered according to the steps described in algorithm 2.

## 5.3 Ease of use and flexibility

That the software should be as user-friendly as possible have been considered throughout the the whole project. The object oriented design, the naming of components and parameters and the way the tool-kit is structured similar to other grasshopper tool-kits are a few ways of trying to achieve this. Another important objective has been to make the software as flexible as possible. The aim has been that the software should be useful even for rare or new cases. This has been considered for example by making sure that the user can create custom fixings, create their own templates, change annotation styles and modify both glass objects, fixing objects, template objects and drawing objects manually in Rhinoceros (all of them can be baked as block objects). The two goals, to make the software user-friendly and to



make it flexible, have proven to sometimes be contradictory. If more flexibility should be provided this often means more settings available for the user to relate to which in turn means that the software gets less flexible. In the development of the software in this thesis one way of reaching these goals have been to try to make the software very easy to use in the most common cases with a lot of default settings but still provide the possibility for custom settings and modifications when needed.

## 5.4 Test case - The Scandic Victoria Tower

Towards the end of this master thesis the software created was tested on an old project that Skandinaviska Glassystem worked on in 2011. Skandinaviska Glassystem was awarded the contract for the construction of the “exterior skin” on the Scandic Victoria Tower in Kista, Stockholm (Glassystem 2011). The building is a 117 meter tall skyscraper hotel (the tallest hotel in all of Scandinavia). The planning of the project was done by Wingårdh Arkitektkontor (Magazin 2012).



Figure 5.23: The Scandic Victoria Tower (Glassystem 2011)



### 5.4.1 Working hours

It took a very long time to create all the glass drawings for the Scandic Victoria Tower in 2010. It has been hard to find out exactly how long time it took but staff from Skandinaviska Glassystem that was involved in the process estimated it to about two working months (300-400 working hours). The same task takes about ten minutes using the new tool-kit. It is clear that a lot of time can be saved but it also needs to be said that the comparison is not quite fair.

First of all the number of working hours needed using the tool-kit is also hard to estimate without actually using the software on a full scale project for real. At this stage the software must be considered as untested and can't be trusted entirely. A hearty review needs to be done of all the drawings created before they can be approved. And even so, the time saved by using the software in its current form would probably vary a lot depending on the project. Most time can be saved on projects with a large amount of glass pieces and where mostly just the most common design rules needs to be considered. A project of a more unusual type would need some manual work, some additional grasshopper definitions or a further developed tool-kit.

To print the drawings to PDF currently takes by far the most time (see figure 5.24). The solution for this step (described a bit more in chapter 5.1.5) is slow but can hopefully be improved quite easily if an integrated PDF-printer is added to Rhinoceros in the future, which according to the author can be considered likely. Another solution worth investigating might be to export the drawings to DXF and then print the drawings in some effective way using some other software that can handle DXF (for example AutoCad).

Another important time saving factor is the possibility to sort all glass pieces with the exact same (or similar enough) shape in groups. It is often enough to dimension and create a drawing for only one of these glass pieces (or a block definition linked to them all). This has also been done for the test on the Scandic Victoria tower. We can see in table 5.1 that the execution time went down from almost three hours to about ten minutes using this technique. The sorting/search for unique shapes was done using a Grasshopper component developed by Jakob Brusewitz and took a few seconds to run. In this example all glass pieces were assumed to have the same glass composition. If all different glass compositions used in the project were to be considered the number of unique glass pieces would have been greater.

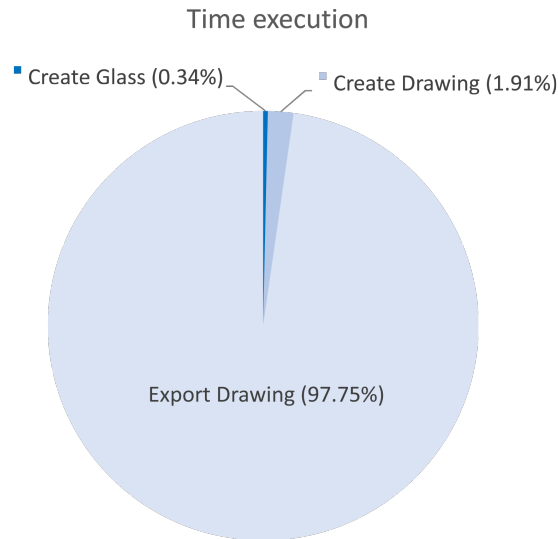


Figure 5.24: Execution time breakdown

Table 5.1: Test results - execution time

Test case	Number of cells	Create glass [s]	Create drawing [s]	Export drawing [s]
Small test	100	0.371	2.08	106.46
All cells	10093	37.4	209	10744.8
All unique cells	574	2.13	11.9	611

## 5.4.2 Output and drawing quality

Two different drawings, of the exact same glass piece, can be seen in figure 5.25 and figure 5.26. The first one drawn back in 2010 and the second one created using the tool-kit developed in this thesis.

There are several differences between the drawings. Many of them depend on the fact that different templates have been used. The templates used at Skandinaviska Glassystem have been updated several times the last couple of years and can also sometimes be different from project to project or building-phase to building-phase depending on what information the drawings needs to include. If needed, principle drawings to show coating position or the orientation of the fixings could easily be added to the drawing created by the tool-kit by modifying the template. However, if the section and principle drawings needs to be glass piece specific (automatically generated from glass composition data and information about fixing system) this would need some additional development of the software.

Another difference is the locations of the fixings. Both drawings display acceptable ways of placing the fixings for the very same glass piece. The design rules available is most of the time not a detailed step by step guide for how to do things but rather a set of rules about what not to do, such as minimum distances to corners that needs to be fulfilled.

The annotations also look different in the new and the old drawing. In the drawings created by the tool-kit the font, the text height and the style of the annotations can easily be changed by the user. The annotations are baked as Rhino annotation objects according to an annotation style defined in Rhino called “SkandSysDimStyle”. If the user wants to change the annotation style this can simply be done in Rhino, either before or after the drawings are baked as blocks in Rhino. Something that is a bit harder for the user to change is the location of the annotations, what measurements that are shown and in what way. It turned out to be a bit of a challenge to come up with a system that creates reasonable annotations for all sorts of shapes and at the same time avoid unnecessary overlapping of annotations and keep the drawing clean and simple. The measurements that preferably should be displayed in the drawings depends a lot on how the glass pieces are produced. In the production process smaller glass pieces are often cut from bigger rectangular glass pieces.

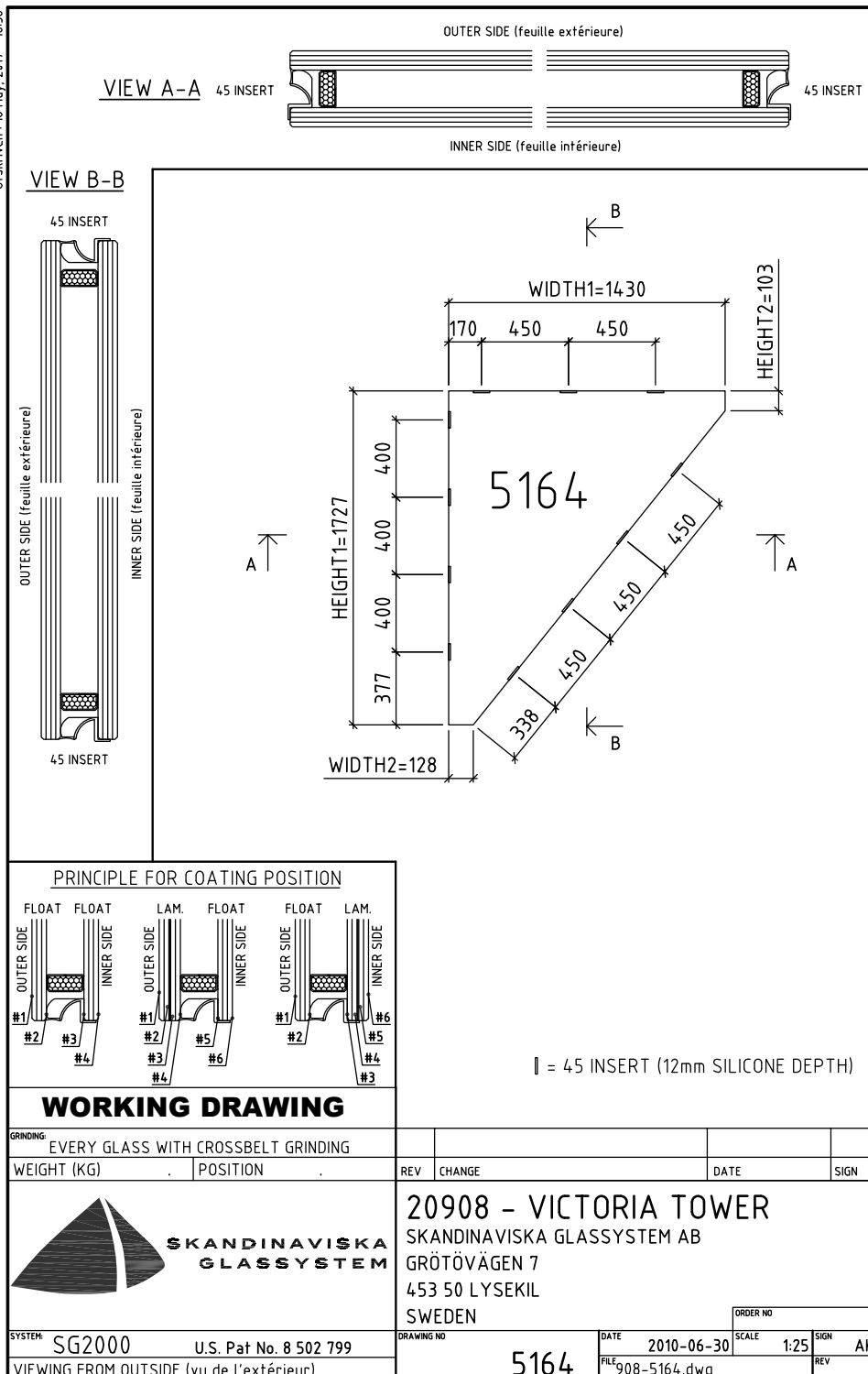


Figure 5.25: Glass drawing example from the Scandic Victoria Tower project. This drawing was created in 2010. The figure is not in scale.

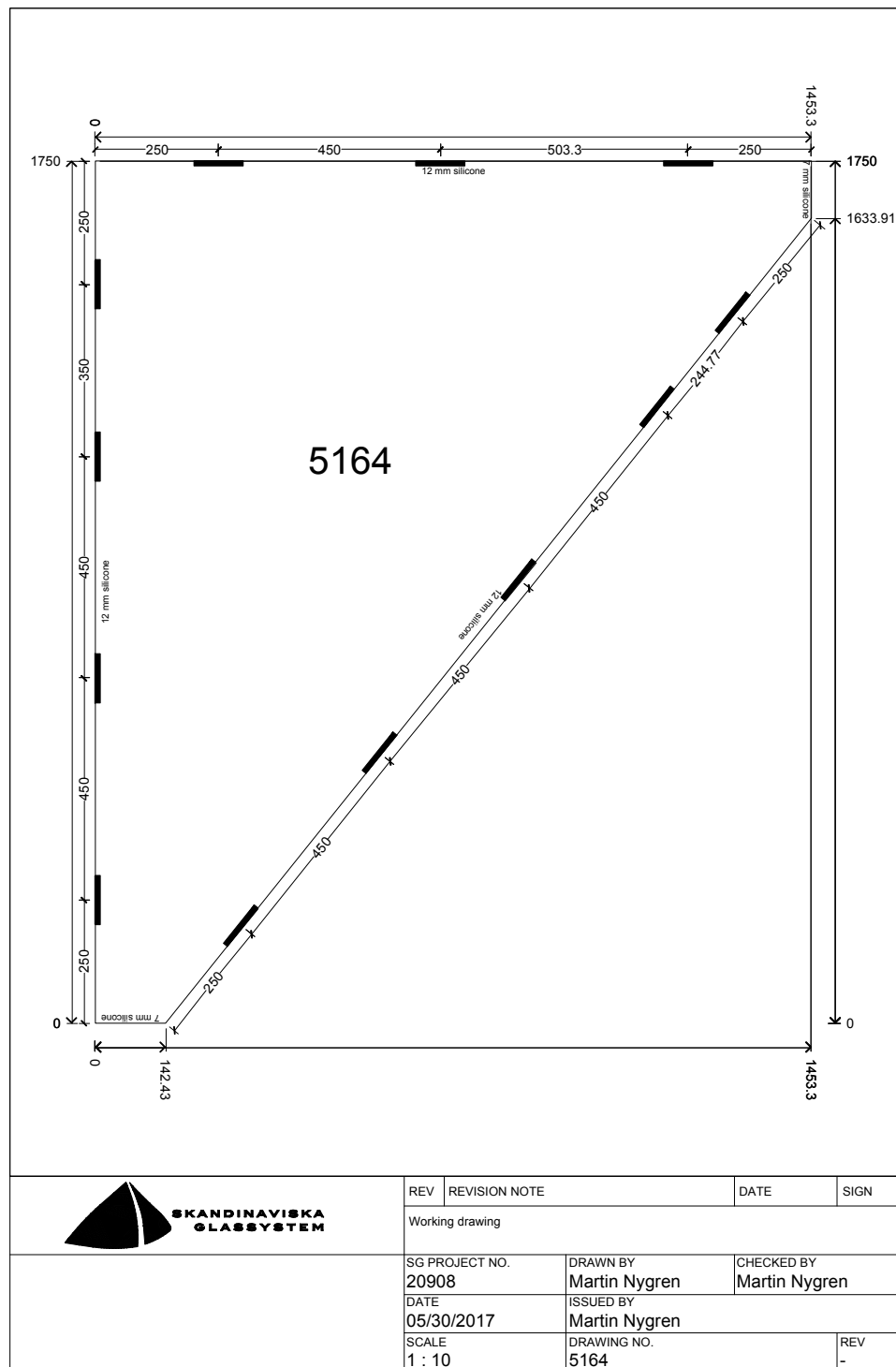


Figure 5.26: Glass drawing example from the Scandic Victoria Tower project. This drawing was generated using the tool-kit developed during this master thesis. The figure is not in scale.

## 6 Discussion

To automate the dimensioning of glass and production of glass drawings by developing a range of grasshopper components seems to be a good way to go. The tool-kit developed in this thesis can massively reduce the time needed for this process. There is still room for improvement though, the possibility to handle less common design rules without manual modifications/custom scripting and a proper software testing process to reduce the number of bugs and time needed for verification of the drawings produced would further improve the usefulness of the software.

### 6.1 Developing for Rhinoceros/Grasshopper - Advantages and disadvantages

Both Rhinoceros and Grasshopper are well established software with a lot of great methods and tools available to model, display and modify geometry. The geometry created can be exported to a range of different formats including for example DXF and then opened in any of the common CAD programs. To create a flexible tool-kit in the environment of Grasshopper is to a great extent a natural and intuitive task. The many already available grasshopper components displays different good ways of structuring and solving things. It is not for nothing that grasshopper is often called visual programming language and environment.

However, great flexibility often follows a somewhat steep learning curve. This is also true when it comes to grasshopper even though less so then one might think. There are great tutorials available to get started and understand the basic concepts. Even so, one of the challenges during the development of the tool-kit have been to provide the desired flexibility without compromising too much on ease of use.

Another challenge was due to how the preview geometry (Grasshopper data types) and baked geometry in Rhinoceros differs from each other. There exist a lot of conversion methods for this but the data types are not always directly convertible and sometimes a data type in one platform does not have an equivalent data type at all in the other platform. This is for example the case for annotations. The drawing objects created using the tool-kit developed in this thesis are baked with annotation objects supported by Rhinoceros but previewed using custom geometry solely created for preview purposes.

To work with block objects was also difficult. It is a great advantage to support the use of blocks since several objects looking the same can be defined and modified at the same time using only one block. The problem is that block objects are not supported by any original grasshopper component and the implementation of blocks in the Rhino SDK is, according to the author, relatively complicated.

The lack of a proper way of printing the drawings to PDF has also been a disadvantage. The solutions for this ended up being messy to code, relatively slow and lacking some desirable options.

## 6.2 Possible future improvements and development

Even though the software developed already make the process much more effective there is definitely room for further development and improvements. Some proper software testing, to detect and fix bugs and other problems, would improve the robustness and the reliability of the software. There are also several additional design rules and recommendations, currently not considered, that the software could be developed to handle in the future. This includes rules that has to do with perforation, screen printing and special solutions for corners where several glass pieces meet. The software does not necessarily need to always generate a solution that fulfils all design rules, that might not even be possible given a certain geometry, but a warning message telling the user what and where some design rules are not met could be sufficient.

Another possible further development is to add the option to, using data about the glass composition and the fixings used, generate additional section drawings to complement the current ones.

The inhouse handbook for glass dimensioning at Skandinaviska Glassystem (Wadman and Hansen 2012) contains, besides the type of design rules mentioned in chapter 2.1, also data about what prices, range of sizes and different glass types different suppliers can offer. A possible future development is to create a component that can give a cost estimate for a set of glass objects. It would then be possible to for example check how, perhaps small, changes in the geometry could reduce the cost in a project a lot.

## 7 Conclusions

In this work software to automate the production of glass drawings has been developed and, even though it is not quite a finished product, it is a useful software with a lot of potential.

Rhinoceros/Grasshopper is an extraordinary platform for sophisticated geometry modelling. Thanks to the open SDK, great support for custom scripting and development and the many already existing components available it is a great environment for developing components or setting up grasshopper definitions to create smart and effective solutions to all sorts of geometry related problems.

The design rules and guide lines handled in this thesis mostly had to do with the geometry of the glass pieces and the fixings. Likewise, the drawing production is also a lot about geometry transformation and modelling. To automate this process in the environment of Rhinoceros/Grasshopper seems to be very suitable. However, it also seems to still be more things to ask for. Especially things to do with annotation, management of text objects and printing was lacking in development for the applications made in this thesis.

The software developed in this thesis has great potential. Speaking with the engineers and other staff at Skandinaviska Glassystem, it seems like this kind of software can be very helpful and make it possible to save a lot of working hours in the future. However, the software will need some testing and further

development before it can be seen as a finished product and reach its full potential.

## 7.1 Suggested further development

There is a lot happening in the construction industry when it comes to automation and other technological improvements. It is hard to predict exactly what will happen next. Today Rhinoceros/Grasshopper seems to be the most suitable environment for geometry related problem solving of the type handled in this thesis but what about the future?

The program Dynamo is similar to Grasshopper, a visual programming environment, that can be linked to and work with the same objects as in the industry standard program Revit. Dynamo is currently far behind grasshopper in development but it looks promising. If Revit continues to grow as an industrial standard Dynamo might very well be a better choice for this kind of automation in the future.

To reduce the risk of spending hours and hours on developing software that suddenly gets outdated it should be considered to write a platform independent source code. The extra job needed to implement the software to a new platform could then potentially be much less. However, to write the code platform independent would take a lot of extra time and might not be worth it at least not at this point. But if the software continues to grow there will be more and more reason to take this into consideration.



# Bibliography

- AECMagazine (2009). *Rhino Grasshopper*. URL: <http://aecmag.com/software-mainmenu-32/293-rhino-grasshopper> (visited on 05/10/2017) (cit. on p. 11).
- Building Sciences, N. I. of (2016). *What is BIM?* URL: <https://www.nationalbimstandard.org/faqs> (visited on 05/10/2017) (cit. on p. 8).
- Dathan, B. and S. Ramnath (2015). *Object-Oriented Analysis, Design and Implementation: An Integrated Approach, Second Edition*. Springer (cit. on p. 10).
- Glassystem, S. (2011). *Victoria Tower, Stockholm*. URL: <http://www.skandglas.se/references/victoria-tower/> (visited on 05/12/2017) (cit. on p. 26).
- Groover, M. P. (2017). *Automation*. URL: <http://academic.eb.com.proxy.lib.chalmers.se/levels/collegiate/article/automation/109393#> (visited on 02/21/2017) (cit. on pp. 6, 7).
- Kensek, K. M. and D. Noble (2014). *Building Information Modeling: BIM in Current and Future Practice*. John Wiley and sons (cit. on p. 8).
- Lamb, F. (2013). *Industrial Automation: Hands on*. McGraw Hill Professional (cit. on p. 8).
- LLC, N. (2017). *Rhino 5 for Windows*. URL: <https://novedge.com/products/2217> (visited on 05/10/2017) (cit. on p. 10).
- Magazin, S. (2012). *Victoria Tower, Stockholm*. URL: <https://www.stylepark.com/en/news/scandic-victoria-tower-in-stockholm-by-wingardh-arkitektkontor> (visited on 05/12/2017) (cit. on p. 26).
- McCool, D. and B. Hardin (2015). *BIM and Construction Management*. John Wiley and sons, Incorporated (cit. on p. 8).
- McNeel, R. and Associates (2017). *Rhinoceros Resources*. URL: <http://www.rhino3d.com/resources/> (visited on 05/10/2017) (cit. on p. 10).
- Tedeschi, A. (2011). *Intervista a David Rutten*. URL: <http://content.yudu.com/Library/A1qies/mixexperiencetoolsnu/> (visited on 05/10/2017) (cit. on p. 11).
- Tolio, T. (2009). *Design of Flexible Production Systems*. Springer Berlin Heidelberg (cit. on p. 9).
- Wadman, J. and C. Hansen (2012). “Glasbibeln”. Design rules and recommendations for glass dimensioning at Skandinaviska Glassystem (cit. on pp. 4, 5, 33).

## A Appendix A

### A.1 Intended practice for the software - an example

In the following chapter an example of how the software, developed in this thesis, can be used will be presented. The example will show a very simple case where manual modifications or additional grasshopper definition is unnecessary.

In this example the different steps explained in algorithm 2 (displayed in chapter 5.2) will be followed.

An overview of the whole process (as it looks on the Grasshopper canvas) can be seen in figure A.1, it shows the grasshopper definition with all the steps in the process grouped and numbered according to algorithm 2. In the sub chapters below the different steps will be described a bit more in detail. There will also be additional illustrations, screen captures from Rhinoceros, to help describing the process.

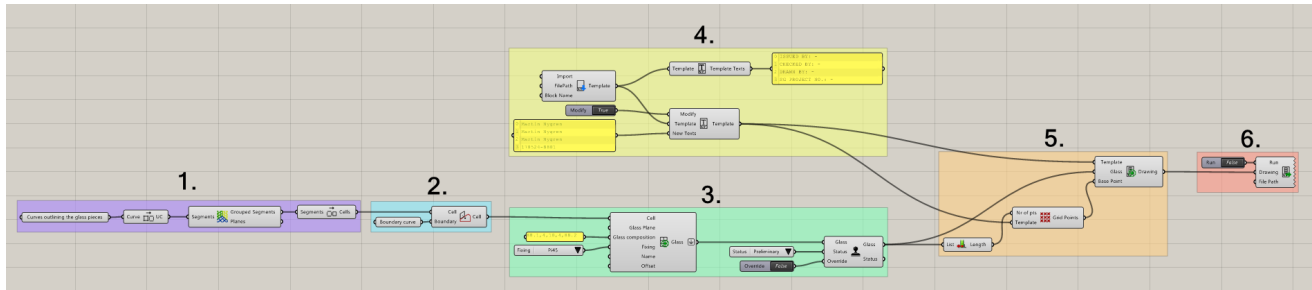
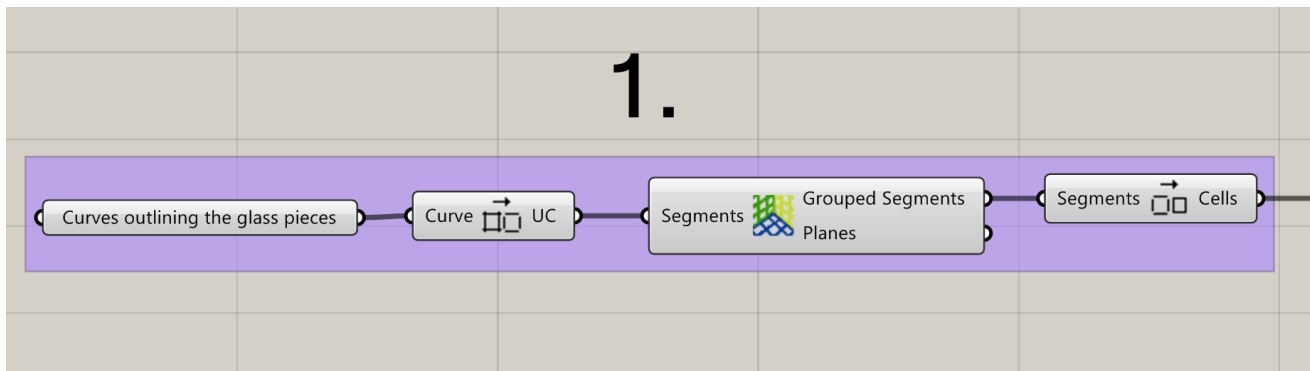


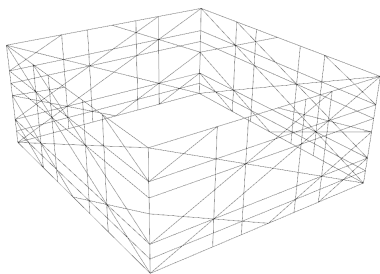
Figure A.1: Showing an example of what the grasshopper definition could look like when the developed tool-kit is used for the whole process of glass dimensioning creating glass drawings. The different parts of the definition is grouped and numbered according to the steps described in algorithm 2.

### A.1.1 Step 1

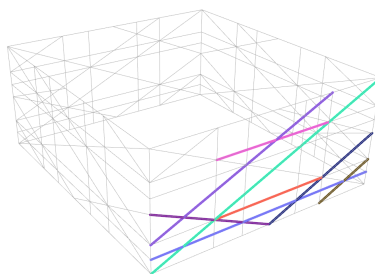
The 3d-model provided as input in this example can be seen in figure A.2b. Lets assume that the model represent 4 walls consisting of irregular edge to edge glass panes. What is desired is to have closed planar curves outlining every glass piece separately. In this case the curves in the model needs to be modified. The provided curves are in this case overlapping, intersecting and angled (as can be seen in figure A.2c). Three different components can be used in combination to create the desired closed curves from this input geometry. The first one is the component “Clean geometry” which creates short curves, segments, from the input curves (see figure A.2d). In this case it divides all input curves in all intersection points and after that removes all duplicate curves. The segments created from this component is then used as the input to the second component “Group segments”. As the name implies the second component sorts the segments in groups, every group contains segments that lie in the same plane. The output contains one list of segments per group created by the component. The four different groups created in this example can be seen in figure A.2e. The segments need to be grouped by plane in order for the third component “Build cells” to work properly. This third component will handle one group of segments at a time and from them build the desired closed planar curves (“cells”). The cells created in this example are illustrated in figure A.2f.



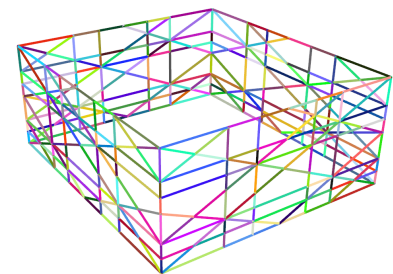
(a) Part of a grasshopper definition performing tasks included in step 1 described in algorithm 2.



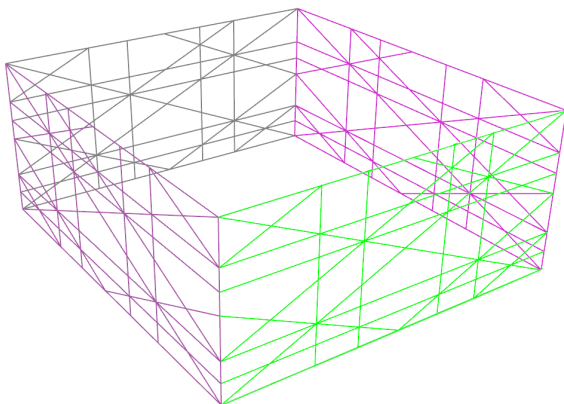
(b) Example of a 3d-model that can be provided as input.



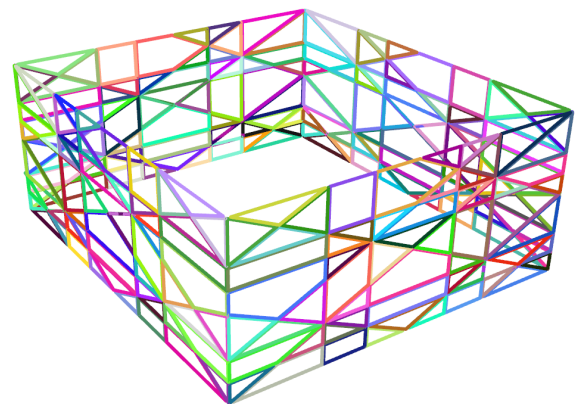
(c) A few curve objects in the 3d-model, highlighted in different colours.



(d) Short segments reaching between intersections have replaced the old curves. All segments are highlighted in different colours.



(e) Segments are sorted by plane.

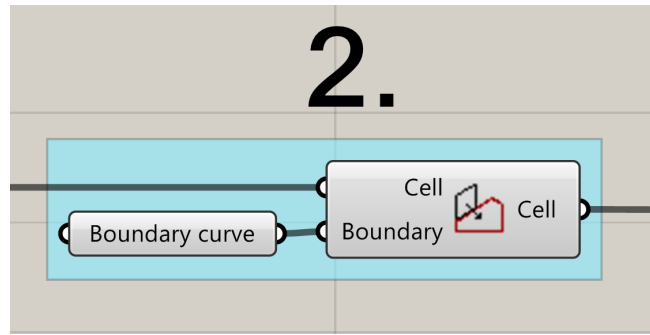


(f) The “Build cell” component have been used to define cells, that is a closed curve representing the module line or outer boundary of a glass piece.

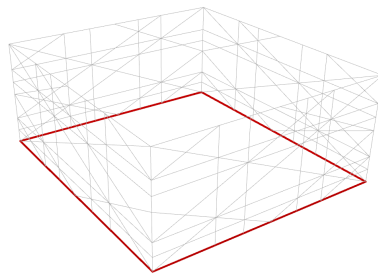
Figure A.2: Step 1

### A.1.2 Step 2

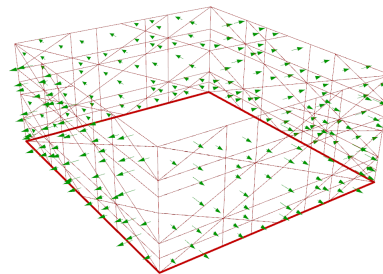
The second step in the process is not always needed, it depends on how the user choose to define the orientation of the glasses in step 3. In this example the glass planes will be defined using the cells (as described in chapter 5.1.2). The glass planes should always have the normal pointing outwards, towards the outside of a building. This means that the directions of many of the curves defining the cells might need to change. In this example this is done using the component “Orient cell (boundary)”. The component needs two inputs, the cell to modify and a boundary curve. The boundary curve should be a closed curve representing the outer boundaries of a building. The output will be cells of the very same shape as the input cells, but now the direction of the curves defining the cells will be so that the planes defined from them will all have normals pointing outwards. Figure A.3b and A.3c shows what this step looks like in Rhino and an example of the same procedure but with a different shaped building can be seen in figure A.3d and A.3e.



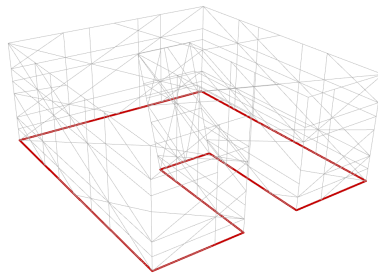
(a) Part of a grasshopper definition performing tasks included in step 2 described in algorithm 2.



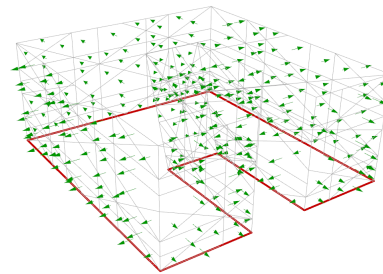
(b) Example of a 3d-model where an example of a possible boundary curve input is highlighted.



(c) Showing cell orientations after the use of the “Orient cell (boundary)” component. Cell normals are shown as green arrows.



(d) Example of a 3d-model where an example of a possible boundary curve input is highlighted.



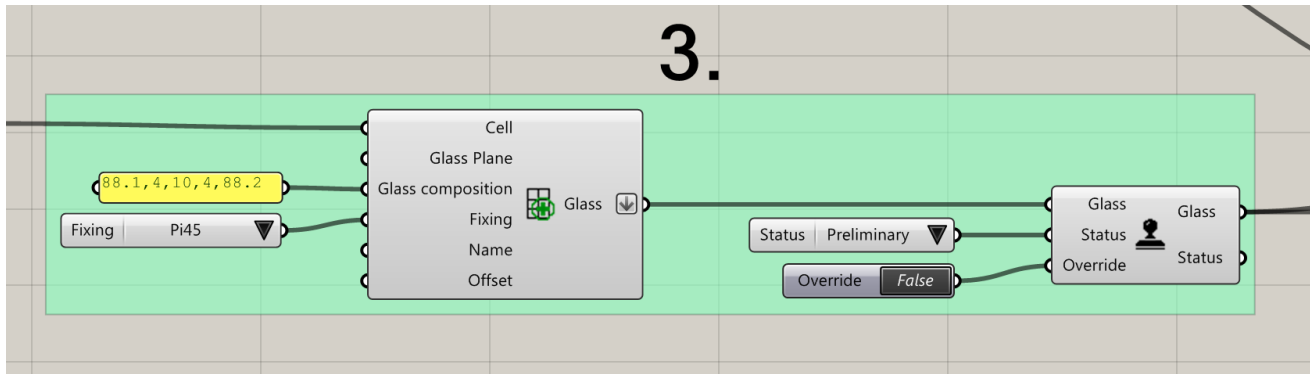
(e) Showing cell orientations after the use of the “Orient cell (boundary)” component. Cell normals are shown as green arrows.

Figure A.3: Step 2

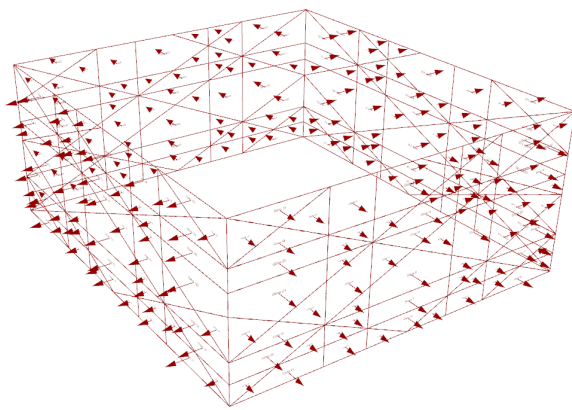
### A.1.3 Step 3

The third step is to create glass objects. This is done using the component “Create Glass”. The component needs some glass specific information as input, such as the type of fixing that is to be used and the glass composition. In this case the user chose to not provide some of the possible inputs (see figure

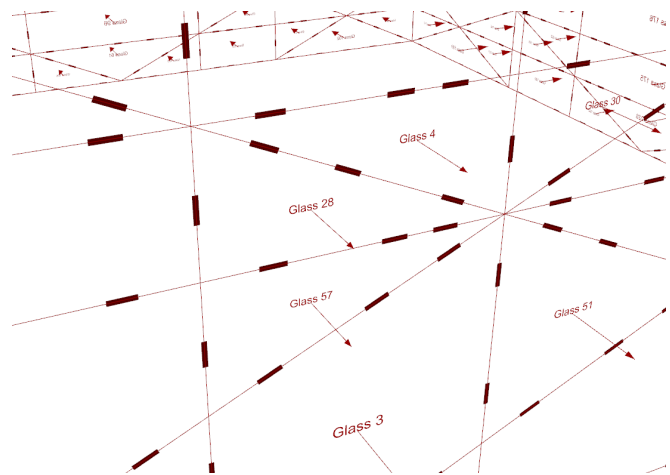
A.4a). Since the input “Glass plane” is missing the glass plane will be defined using the provided cell (as mentioned in chapter A.1.2) and due to the lack of provided glass name the glass objects will be named automatically. The output of the component is the glass objects, they are also as default previewed in Rhino (see figure A.4b and figure A.4c). The preview of a glass object will display the boundary of the glass piece, the normal of the glass plane as an arrow, the fixings along the edges and the name of the glass object. When a glass object is created the user can choose to use the component “Status” to change the status of the glass object. The status is something often specified in the finished glass drawings, saying whether the drawing is for example “preliminary” or a “Working drawing”.



(a) Part of a grasshopper definition performing tasks included in step 3 described in algorithm 2.



(b) Example of how it can look when glass objects, created using the “Create glass” component, are previewed in a Rhino viewport.



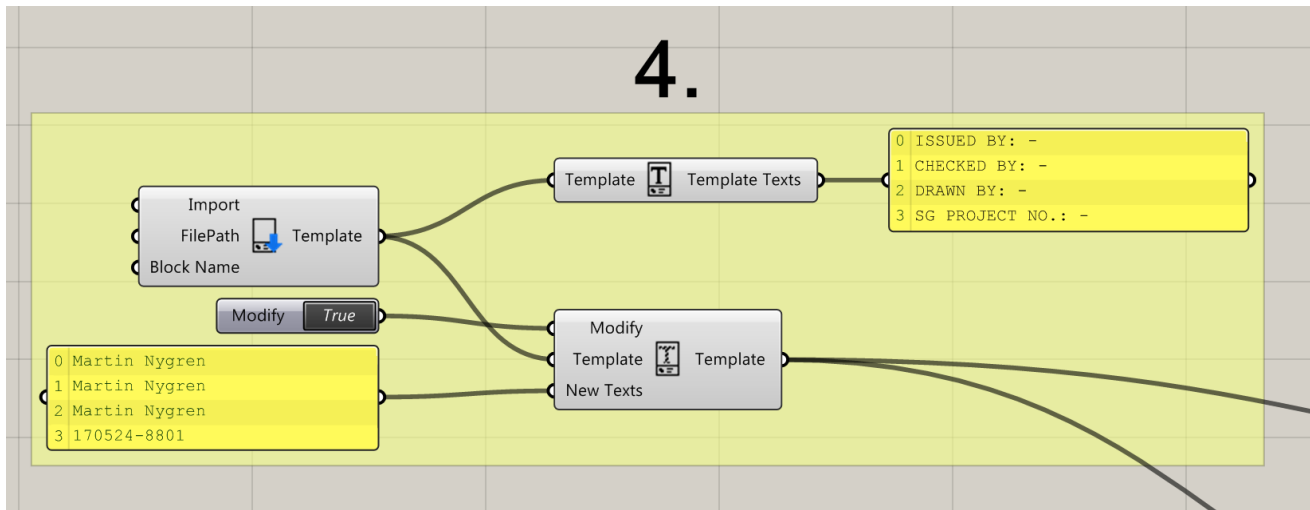
(c) A more zoomed in view of figure A.4b.

Figure A.4: Step 3

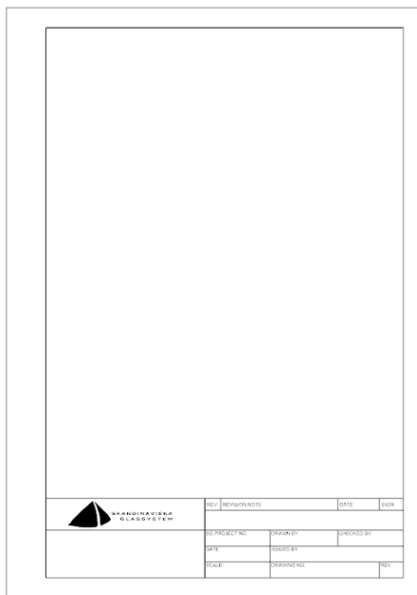
#### A.1.4 Step 4

Step 4 is about creating the template or templates needed for the drawings. In this example only one template will be used for all the drawings created. The component “Import template” is used to import

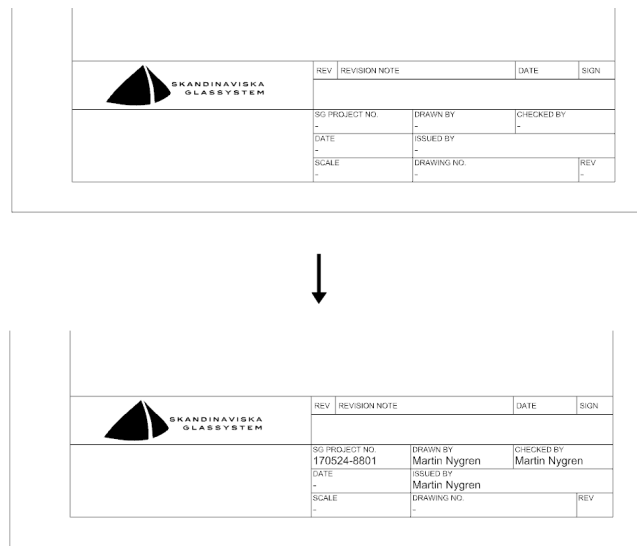
a default template read from a .3dm-file somewhere on the user's computer. The imported template, in this example, can be seen in figure A.5b. In order to update the template with some project specific information (see figure A.5c), that the user wants to be displayed in the drawings, the components “Extract template texts” and “Modify template” are used. All text objects in the template that have custom object names can be modified using this technique.



(a) Part of a grasshopper definition performing tasks included in step 4 described in algorithm 2.



(b) Example of what a template object could look like.

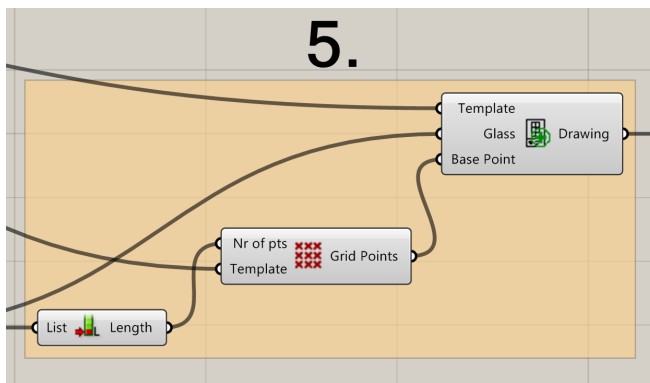


(c) Example of how the template look before and after the texts are changed using the component “Modify template texts”.

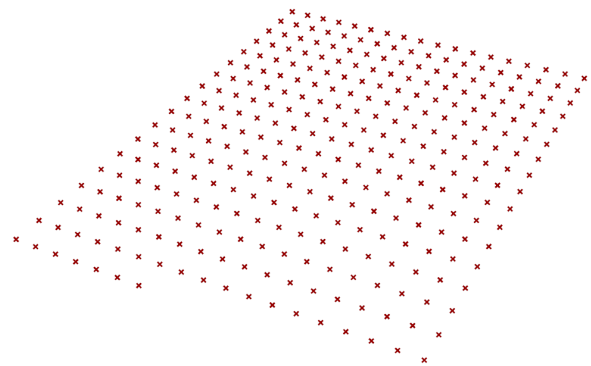
Figure A.5: Step 4

### A.1.5 Step 5

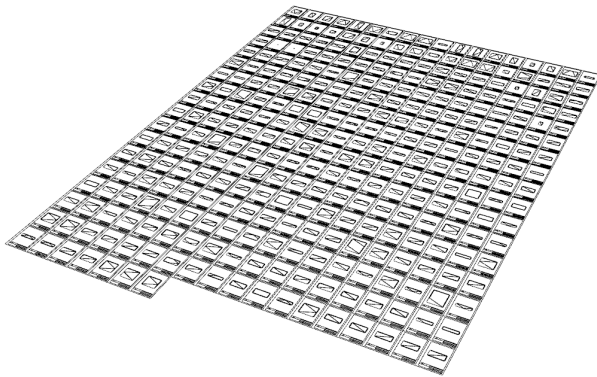
When a suitable template have been imported and all glass objects have been created the next step is to create the drawing objects. This can be done using the component “Create Drawing”. The component needs a template and a glass object to run. A base point can also be provided as an input is the user wants to be able to preview the drawings in Rhino. In this example the user chose to create a grid of points (see figure A.6a) using the component “Grid points” and then use these points together with the glass objects and the template object to create the drawings. The drawings are, as default, preview in Rhino as can be seen in figure A.6c and figure A.6d.



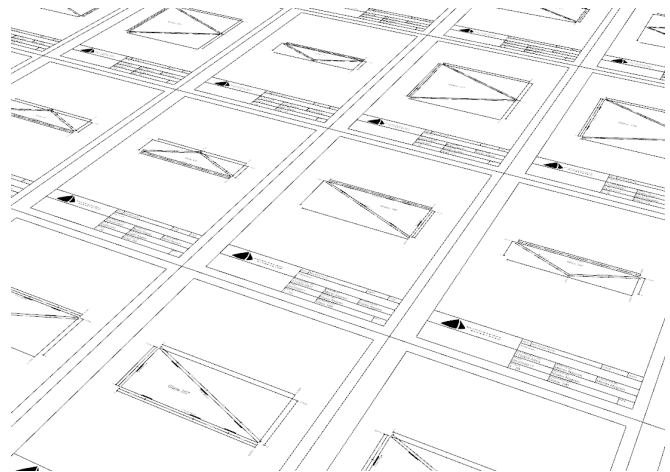
(a) Part of a grasshopper definition performing tasks included in step 5 described in algorithm 2.



(b) Example of grid of points created using the component “Grid points”.



(c) A preview of drawings created using the “Create drawing” component.



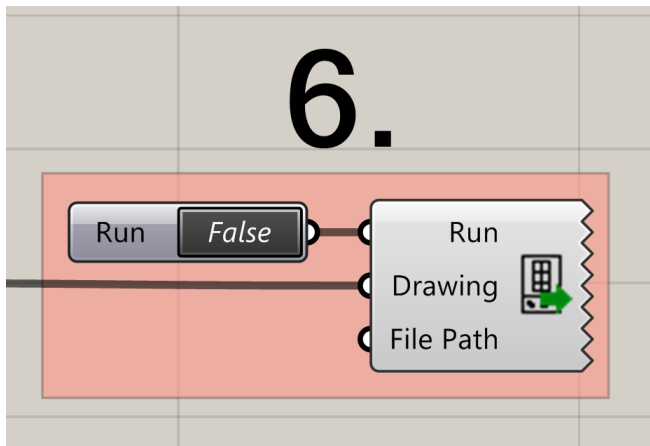
(d) A more zoomed in view of figure A.6c.

Figure A.6: Step 5

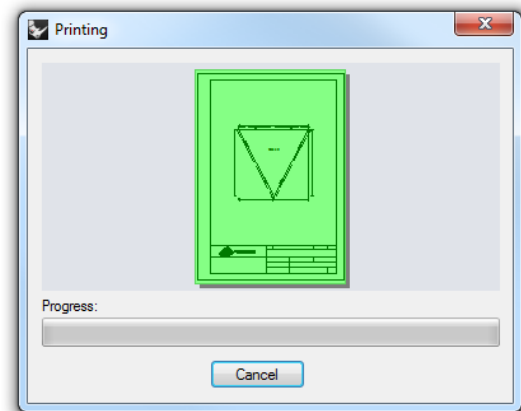


### **A.1.6 Step 6**

The last step in this example is to print the drawings to PDF. This is done using the component “Export drawing”. The drawings to print and a file path specifying a folder to print to is needed as inputs to the component. A boolean data type should also be provided to the input called “Run”. The component will only run, export the drawings that is, if this boolean (a data type with only two possible values: true or false) is set to true. When the printing of a drawing is in progress a window, as the one seen in figure A.7b, will be shown. The drawings, printed to PDF, will then show up in the folder specified as input (see figure A.7c). Figure A.7d displays one of the drawings created in this example.



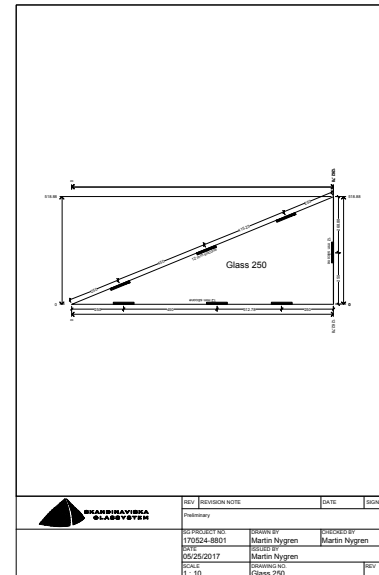
(a) Part of a grasshopper definition performing tasks included in step 6 described in algorithm 2.



(b) Window showing the printing progress, displayed while printing drawings.

Drawing (19)	2017-05-25 14:48	Foxit Reader PDF ...	34 kB
Drawing (20)	2017-05-25 14:48	Foxit Reader PDF ...	34 kB
Drawing (21)	2017-05-25 14:48	Foxit Reader PDF ...	34 kB
Drawing (22)	2017-05-25 14:48	Foxit Reader PDF ...	35 kB
Drawing (23)	2017-05-25 14:48	Foxit Reader PDF ...	36 kB
Drawing (24)	2017-05-25 14:48	Foxit Reader PDF ...	34 kB
Drawing (25)	2017-05-25 14:48	Foxit Reader PDF ...	35 kB
Drawing (26)	2017-05-25 14:48	Foxit Reader PDF ...	35 kB
Drawing (27)	2017-05-25 14:48	Foxit Reader PDF ...	36 kB
Drawing (28)	2017-05-25 14:48	Foxit Reader PDF ...	36 kB
Drawing (29)	2017-05-25 14:48	Foxit Reader PDF ...	34 kB
Drawing (30)	2017-05-25 14:48	Foxit Reader PDF ...	36 kB
Drawing (31)	2017-05-25 14:49	Foxit Reader PDF ...	34 kB
Drawing (32)	2017-05-25 14:49	Foxit Reader PDF ...	38 kB
Drawing (33)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB
Drawing (34)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB
Drawing (35)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB
Drawing (36)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB
Drawing (37)	2017-05-25 14:49	Foxit Reader PDF ...	37 kB
Drawing (38)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB
Drawing (39)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB
Drawing (40)	2017-05-25 14:49	Foxit Reader PDF ...	37 kB
Drawing (41)	2017-05-25 14:49	Foxit Reader PDF ...	37 kB
Drawing (42)	2017-05-25 14:49	Foxit Reader PDF ...	35 kB
Drawing (43)	2017-05-25 14:49	Foxit Reader PDF ...	36 kB

(c) The printed drawings end up in a folder specified by the user.



(d) Example of what a printed drawing could look like. This figure is not in scale.

Figure A.7: Step 6