TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Bauingenieurwesen Institut für Bauinformatik

DIPLOMARBEIT

Entwicklung einer Methode zum Einsatz der digitalen Fertigung im Bauwesen am Beispiel eines parametrischen Fassadenknotens

Development of a methodology for the application of digital manufacturing in the construction industry using the example of a parametric façade node

eingereicht von cand. ing. Adrian Schubert
geb. am 30.05.1996 in Immenstadt i. Allgäu

Betreuer/in:

- Prof. Dr.-Ing. habil. Karsten Menzel
- M.Eng. Suraj Sunil Kumar Shetty

Dresden, den 30.09.2019

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich reiche sie erstmals als Prüfungsleistung ein. Mir ist bekannt, dass ein Betrugsversuch mit der Note „nicht ausreichend“ (5,0) geahndet wird und im Wiederholungsfall zum Ausschluss von der Erbringung weiterer Prüfungsleistungen führen kann.

Name: Schubert
Vorname: Adrian
Matrikelnummer: 4045092

Dresden, den 30.09.2019

_____
Unterschrift cand. ing. A. Schubert

Danksagung

An dieser Stelle möchte ich mich bei den Mitarbeitern des Instituts für Bauinformatik bedanken, die mir trotz meiner Vertiefung „Gebäude-Energie-Management“ während meines Studiums und der Ausarbeitung dieser Diplomarbeit unterstützend zur Seite standen. Ein besonderer Dank gilt dabei:

- Prof. Dr.-Ing. habil. Karsten Menzel danke ich für die Schaffung der Rahmenbedingungen meiner Projekt- und meiner Diplomarbeit – insbesondere für die Unterstützung bei den bürokratischen Voraussetzungen. Er stand mir mit Diskussionen und Ratschlägen zur Seite. Darüber hinaus bedanke ich mich für die Zusammenarbeit im Rahmen meiner Tätigkeit als studentische Hilfskraft.
- M.Eng. Suraj Sunil Kumar Shetty danke ich für die Betreuung bei der Ausarbeitung und die Themenfindung der Projekt- und der Diplomarbeit. Er hat viel Zeit für meine offenen Fragen und Probleme aufgewendet, die dank seiner Hilfe gelöst wurden.
  A special thanks belongs to M.Eng. Suraj Sunil Kumar Shetty for the support during the elaboration and finding the topic of the project work and the diploma thesis. He spent a lot of time on my open questions and problems, which were always solved due to his help.

Ich möchte mich auch bei meiner Familie und meinen Freunden bedanken, die mich während meines Studiums und insbesondere während der Bearbeitungsphase dieser Arbeit unterstützt haben. Ein besonderes Dankeschön gebührt meiner Freundin Nicole Willer, die mir während der Ausarbeitung meiner Diplomarbeit emotional und mit konstruktiven Ratschlägen zur Seite stand. Außerdem möchte ich ihr, meinem Vater Bernd Schubert und meiner Großmutter Brigitte Kaden für das Korrekturlesen danken.

I Abstract

Diese Diplomarbeit untersucht die Anwendbarkeit von digitalen Produktionsmethoden im Bauwesen. Dazu soll beispielhaft ein parametrischer Fassadenknoten ausgearbeitet werden. Dieser wird mithilfe des Parametric Designs entwickelt und in Pfosten-Riegel-Bauweise ausgeführt. Der parametrische Entwurf wurde hierfür in einer grafischen Programmierumgebung umgesetzt. Zu diesem Zweck wird eine geeignete Datenstruktur für den parametrischen Fassadenknoten erarbeitet. Diese Arbeit soll zeigen, wie digitale Produktionsmethoden effektiv im Bauwesen eingesetzt und wie digitale Daten zur Errichtung von realen Fassaden genutzt werden können. Des Weiteren werden in dieser Arbeit Erläuterungen zum aktuellen Stand der Technik von verschiedenen Fassadensystemen aufgeführt. Diese Diplomarbeit ist daher sowohl für Architekten und Bauingenieure, aus den Bereichen der Fassadenkonstruktion und der Bauinformatik, als auch für Informatiker, die sich mit Datenmodellen auseinandersetzen, relevant.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Abkürzungsverzeichnis

3-D	dreidimensional
AM	additive Fertigung (engl. additive Manufacturing)
BIM	Building Information Modeling
CAD	Computer Aided Drafting beziehungsweise Computer Aided Design
CAM	Computer Aided Manufacturing
ESG	Einscheibensicherheitsglas
LBM	Laser-Strahlschmelzen (engl. Laser Beam Melting)
LOD	Level of Development
LoD	Level of Detail
SSG	Structural Sealant Glazing
SZR	Scheibenzwischenraum
VSG	Verbundsicherheitsglas
WDVS	Wärme-Dämm-Verbundsystem
WOP	werkstattorientierte Programmierung

1 Einleitung

1.1 Motivation

Das derzeitige Bauwesen steht vor der großen Herausforderung der Digitalisierung. Ein zentrales Thema stellt dabei das Building Information Modeling (BIM) dar. Auch die deutsche Politik und Wirtschaft nehmen sich dieser Problematik an und setzten Hoffnung in die neuen Technologien der Industrie 4.0. In diesen wird die Möglichkeit gesehen, Bauprojekte effizienter und vor allem innerhalb des gegebenen Kosten- und Zeitrahmens zu realisieren. Die deutsche Politik sieht daher eine schrittweise Einführung vor, die im Stufenplan des Bundesministeriums für Verkehr und digitale Infrastruktur (BMVI) dargestellt ist.

Mit BIM kann ein ganzheitlicher Ansatz der Bauwerksplanung geschaffen werden, der den gesamten Lebenszyklus berücksichtigt. So können Daten nicht nur für den Neubau genutzt werden, sondern auch beispielsweise für Umbauten oder den Abriss. Über den Lebenszyklus hinweg, können so mehrere Modelle desselben Bauwerks entstehen, die die verschiedenen Phasen darstellen. Ohne die Nutzung von BIM gehen die Daten meist zwischen den verschiedenen Phasen verloren, sodass frühere Erkenntnisse neu erarbeitet werden müssen. Des Weiteren liegen von einer Planungsphase meist verschiedene Modelle, beziehungsweise Pläne, vor, da jedes Gewerk in einem eigenen Modell oder Plan arbeitet. Durch die Zusammenführung der Planungen der verschiedenen Gewerke, wird die Speicherung und Nutzung der Modelle für eine spätere Verwendung deutlich vereinfacht.

Aber nicht nur in BIM ist ein großes Potenzial zur Effizienzsteigerung gegeben. So sind auch die Bereiche der digitalen Produktion, der Smart Factory und die Einbindung von Robotern nennenswert. Durch diese Automatisierungsprozesse des Bauablaufs können so Kosten- und Zeitersparnisse erzielt werden und darüber hinaus die Arbeitsbedingungen auf der Baustelle sowie der Arbeitsschutz verbessert werden. Eine große Herausforderung, vor der Bauingenieure stehen, sind die Entwicklungen moderner Architektur. Es ist ein Trend zu erkennen, immer komplexeren Formen der Gebäudehülle zu entwerfen. Mithilfe der neuen Technologien wird dem Bauingenieurwesen die Möglichkeit gegeben, die Wünsche der Architekten zu erfüllen.

Vor allem in der Vorfertigung können die digitale Produktion und Roboter eingesetzt werden. Im Vergleich zu baustellenseitigen Montagen wird die Präzision der Roboter deutlich erhöht. Wie eine Arbeit von Samiee (2019) zeigt, können parametrische Elemente einer Elementfassade von Robotern vorgefertigt werden. Hierfür kann ein Roboterarm verwendet werden, der die einzelnen Profile mit Knotenpunkten verbindet. Die Fassadenelemente wurden in einer vorangestellten Projektarbeit des Autors im Parametric Design erzeugt, sodass die Fassadengeometrie mithilfe einer graphischen Programmierumgebung modifiziert werden kann. Der Bewegungsablauf des Roboterarms aus der Arbeit von Samiee (2019) passt sich automatisch an Änderungen an. Innerhalb dieser Arbeit konnte die Anwendbarkeit von Robotik zur Fertigung von Elementfassaden gezeigt werden. Dieser Ansatz ermöglicht die Produktion von verschiedensten Elementgeometrien, sodass die gestalterischen Möglichkeiten innerhalb eines Entwurfes stark erhöht werden.

1.2 Zielstellung

Ziel dieser Arbeit ist es aufzuzeigen, dass die Methoden der digitalen Produktion für die Errichtung von Gebäuden angewendet werden können. Hierfür soll das notwendige Expertenwissen im Bereich Fassadenknoten ermittelt werden. Die konstruktiven Fertigungsdetails eines Fassadenknotens sind zu entwickeln. Zudem soll ein Algorithmus zur Erstellung von digitalen Fassadenknoten mithilfe des Konzepts „Parametric Design“ entwickelt werden. Hierfür wird eine grafische Programmierumgebung verwendet. Anschließend sind die Fertigungsdetails zur Erstellung eines realen Fassadenknotens mit digitalen Produktionsverfahren auszuarbeiten und eine vergleichende Untersuchung mit konventionellen Methoden zu erstellen.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich zunächst in die Erläuterung des benötigten Wissens über Fassadensysteme und parametrischen Entwerfen. Darauf aufbauend soll dem Leser die Wahl und Modifizierung der Bauteilkomponenten eines Profilquerschnitts einer Pfosten-Riegel-Fassade sowie deren Implementierung innerhalb eines parametrischen Entwurfs näher gebracht werden. Nach der Erstellung der Querschnitte wird die Erzeugung des dreidimensionalen Fassadenknotens detailliert betrachtet. Abschließend werden Erläuterungen zur digitalen Produktion des Knotens gegeben und eine vergleichende Untersuchung mit konventionellen Methoden dargestellt.

2 Grundlagen von Fassaden

Fassaden stellen einen wesentlichen Teil von Gebäuden dar und nehmen dabei eine besondere Rolle ein. Knaack et al. (2014: S. 9) beschreiben Fassaden als Konstruktion, die nicht auf den tatsächlichen Raum, der von der Konstruktion eingenommen wird, begrenzt ist. Damit ist gemeint, dass Fassaden eine gewisse Wirkung sowohl auf die Außenwelt, als auch auf den Innenraum ausüben. Eine Fassade beeinflusst maßgeblich die Repräsentation eines Gebäudes und ist selbst innerhalb des Bauwerks spürbar.

Im Folgenden sollen architektonische, historische sowie bauphysikalische und konstruktive Grundlagen von Fassaden erläutert werden.

2.1 Aufgaben und Anforderungen

Als Bauteil, das den Außen- vom Innenraum abtrennt, besitzen Fassaden eine besondere Rolle in Bauwerken und übernehmen somit besondere Aufgaben. Nach Knaack et al. (2007: S. 9 u. 36) werden folgende Aufgaben für Fassaden beschrieben:

- Architektonische Erscheinung des Gebäudes
- Abgrenzung von Innenraum und Außenwelt
- Schutzfunktionen
  - Vor Wetter
    - Sonnenschutz
    - Wärmeschutz
    - Schutz vor Feuchtigkeit
  - Vor Außenwelt
    - Schallimmissionen
    - Schutz der Privatsphäre
- Sichtbeziehung zwischen Innenraum und Außenwelt
- Krafteinflüsse
  - Eigengewicht
  - Druck- und Sogkräfte aus Windlasten

  - Zum Teil tragende Funktion für andere Bauteile

- Belichtung
- Belüftung

Herzog, Krippner und Lang (2004: S. 18 f.) ergänzen hierzu die Funktion der einbruchhemmenden Wirkung.

Neben den genannten Funktionen beschreiben sowohl Knaack et al. (2014: S. 36) als auch Herzog, Krippner und Lang (2004: S. 19), dass eine Fassade in der Lage sein muss, sich an wechselnde Bedingungen anpassen zu können. Hierbei sind Regelfunktionen, wie etwa variable Verschattungsvorrichtungen – beispielsweise Jalousien – gemeint, die bei Sonneneinstrahlung heruntergefahren werden können. Herzog, Krippner und Lang (2004) verstehen die Fassade dabei als „‚dritte Haut' des Menschen" und verdeutlichen damit die Funktion der Fassade, Schwankungen von äußeren Bedingungen ausgleichen zu können, sodass im Gebäudeinneren ein für den Menschen behagliches Klima erreicht werden kann. Ferner ist damit gemeint, dass eine Fassade extreme Außenbedingungen abfangen muss, um weiterhin funktionsfähig zu sein.

Die Funktionen, die eine Fassade übernehmen muss, hat sich im Laufe der Geschichte maßgeblich gewandelt. Das folgende Kapitel 2.2 soll diesen Wandel aufzeigen und einen kleinen Überblick über die Geschichte der Fassade geben.

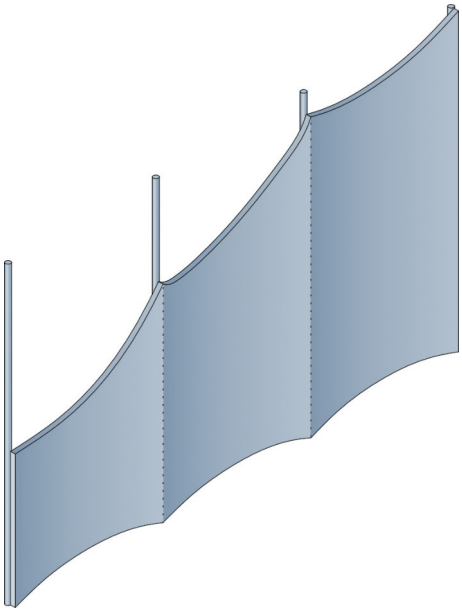
2.2 Historischer Überblick

Für einen historischen Überblick über Fassadensysteme ist es zunächst wichtig, sich die Ursprünge des menschlichen Wohnens zu verdeutlichen. Hierbei kann in zwei grundlegende Tendenzen unterschieden werden. Zum einen gibt es die nomadische und zum anderen die sesshafte Lebensweise. Diese Differenzierung ist maßgeblich an der Entwicklung von verschiedenen Konstruktionsweisen von Bauwerken beteiligt. Die sesshafte Lebensweise führte zu massiven Konstruktionen, die ortsfest verbaut waren. Die Haupteinflüsse für Gebäude waren dabei die Dauerhaftigkeit und die Schutzfunktion, insbesondere vor der Witterung. Andererseits entwickelten Nomaden flexible und mobil einsetzbare Behausungen in Form von Zelten. Der Fokus dieser zweiten Bauweise lag weniger auf der Schutzfunktion, sondern vielmehr auf der Transportfunktion. Dies ist damit zu begründen, dass für nomadische Völker der Transport der Zelte eine alltägliche Herausforderung darstellte. (Knaack et al. 2007: S. 14, 22)

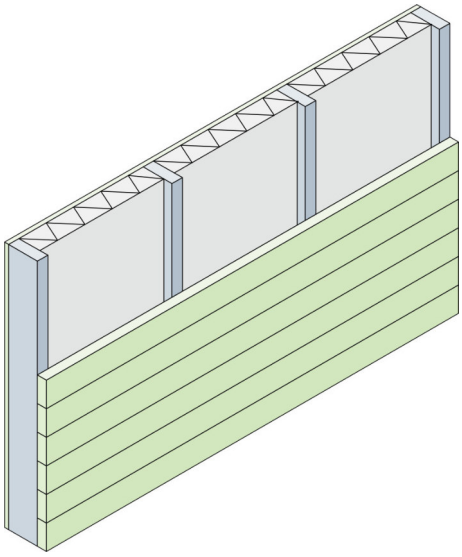
2.2.1 Leichtbau

Nomadische Kulturen entwickelten eine Leichtbauweise. Bei den anfänglichen Strukturen handelte es sich um Skelettstrukturen, die zeltartig umhüllt wurden. Die Grundlage hierfür war das Transportproblem der Behausungen. Aus dieser Problematik folgte die Trennung der Stützkonstruktion von der schützenden Hüllfläche. (Knaack et al. 2007: S. 22) Das Konzept der Zeltkonstruktion ist schematisch in Abbildung 1 (a) dargestellt.

In Europa entwickelte sich später der Fachwerkbau. Hierbei wird die massive Wand vollständig durch eine Holzrahmenkonstruktion mit Ausfachungen aus Geflecht, Lehm oder Ziegel ersetzt. Der gewählte Baustoff ist dabei von der Region abhängig. Beim Fachwerk kann zwischen dem normannischen Fachwerkbau mit in die Wandkonstruktion eingelegter Geschossdecke und dem alemannischen Fachwerkbau mit auf die Wandkonstruktion aufgelegter Geschossdecke unterschieden werden. In Amerika bildete sich eine zum Fachwerk vergleichbare Bautechnik, dem sogenannten Holzskelettbau. Wie in Abbildung 1 (b) dargestellt, besteht die Tragstruktur ebenfalls aus einer Holzkonstruktion. Im Gegensatz zum Fachwerk werden die Ausfachungen aus Holzwerkstoffplatten hergestellt. Diese sorgen für die notwendige Aussteifung der Wandkonstruktion. Bei dieser Bauweise erweist es sich als problematisch, dass nur eine kleine bauphysikalisch wirksame Masse vorhanden ist und somit nicht vom Effekt der Wärmespeicherung profitiert werden kann. (Knaack et al. 2007: S. 22 f.)



(a) Zeltkonstruktion



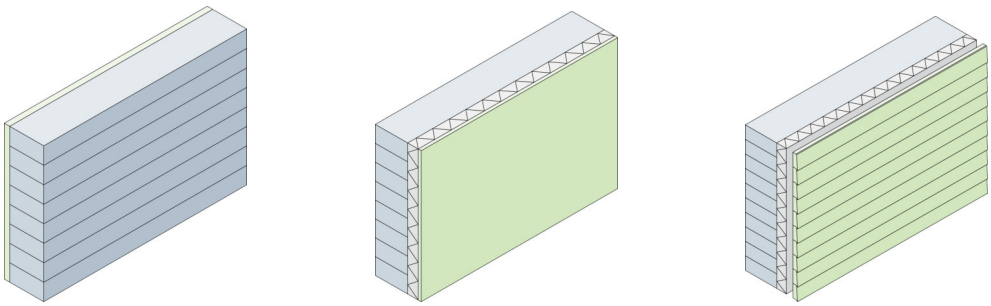
(b) Holzskelettbau

Abbildung 1: Historischer und moderner Leichtbau
Quelle: In Anlehnung an Knaack et al. (2014: S. 22 f.)

2.2.2 Massive Wandkonstruktion

Parallel zu den Entwicklungen der Leichtbauweise wurden massive Wandkonstruktionen, vor allem bei sesshaften Volksgruppen in kalten Klimaregionen, entwickelt. Die Hauptfunktionen waren die Dauerhaftigkeit und der Schutz vor der Witterung. Massive Konstruktionen wurden vor allem aus regionalen Ressourcen hergestellt. So bestanden die Bauteile anfangs aus gefundenen oder selbst hergestellten Materialien, wie etwa Findlinge, Bruchsteine oder gebrannten Ziegel. (Knaack et al. 2007: S. 14) Die Massivbauweise kann sich von der Leichtbauweise dadurch differenzieren, dass alle in Kapitel 2.1 erläuterten Anforderungen von einem einzigen Bauteil übernommen werden. (Cheret et al. 2015: S. 48) In Abbildung 2 (a) ist der schematische Aufbau einer Massivwandkonstruktion gegeben.

Für die Errichtung von Fassaden an massiven Wandkonstruktionen haben sich zwei Bauweisen durchgesetzt. Knaack et al. (2007: S. 14) unterscheidet dabei in Warm- und Kaltfassaden. Als Warmfassaden werden Fassaden verstanden, bei denen sich die wärmedämmende Schicht direkt an der Fassadenkonstruktion befindet. Dieses Schema findet sich in Abbildung 2 (b) wieder. Hierfür spielt es keine Rolle, ob diese Schicht nach außen hin, wie beispielsweise bei einem Wärme-Dämm-Verbundsystem (WDVS), oder nach innen hin, wie beispielsweise bei einer Innendämmung, positioniert ist. Kaltfassaden, wie sie in Abbildung 2 (c) dargestellt sind, werden von den Warmfassaden dadurch abgegrenzt, dass die Wetterschutzschicht durch eine Luftschicht vom Rest der Konstruktion abgekoppelt ist. Dies hat den Vorteil gegenüber Warmfassaden, dass aufgrund von Luftzirkulation eine Trocknungsfunktion an der Wärmedämmung entsteht. Knaack et al. (2007: S. 14) spricht dabei von Wasser, das durch einen Schaden eingedrungen ist. Hier ist zu ergänzen, dass aufgrund der wärmedämmenden Funktion der Fassade Wasserkondensat entstehen kann. Der physikalische Hintergrund ist, dass Luft, die unter den Taupunkt abgekühlt wird, Kondensat bildet. Da sich innerhalb einer Wand ein starkes Temperaturgefälle bildet, um die Differenz von



(a) Massivwand (b) Warmfassade (c) Kaltfassade

Abbildung 2: Massive Wandkonstruktionen
Quelle: In Anlehnung an Knaack et al. (2014: S. 14)

Innen- und Außentemperatur auszugleichen, kann gegebenenfalls eine Kondensatbildung innerhalb der Wandkonstruktion stattfinden (Cheret et al. 2015: S. 278 ff.). In Abbildung 3 ist ein beispielhaftes Glaserdiagramm gegeben. Dieses zeigt die Veränderung des Dampfdruckes über einen kerngedämmten Wandaufbau. Der Dampfdruck ist hellblau dargestellt. Er ändert sich von p_i zu p_e. Gleichzeitig ist dunkelblau der Sättigungsdampfdruck p_s gegeben. Dessen Verlauf ist von der rot dargestellten Temperatur abhängig. Dabei ist zu erkennen, dass der Sättigungsdampfdruck nach außen hin abnimmt. Analog dazu sinkt auch die Temperatur. Der Sättigungsdampfdruck fällt besonders stark im Bereich der Kerndämmung. Dies ist in dem starken Temperaturgefälle infolge der Dämmwirkung begründet. Sollte der Dampfdruck den Sättigungsdampfdruck übersteigen, bildet sich Tauwasser. In diesem Beispiel ist dies nicht der Fall.

Der Massivbau hat sich im Laufe der Jahre stetig weiterentwickelt. Als erster Entwicklungsschritt waren Öffnungen für Rauchabzüge zu beobachten. Diese wurden in einem zweiten Schritt vergrößert, da sie neben der Belüftungsfunktion auch die Funktion der Belichtung übernehmen sollten. Mit den großen Öffnungen wurden jedoch die Gefügestörungen so relevant, dass eine Lösung gefunden werden musste, um das statische Gleichgewicht zu erhalten. Anfangs wurde dieses Problem durch das Einfügen eines horizontalen Balkens als Sturz gelöst. (Knaack et al. 2007: S. 16) Erst mit den bautechnischen Entwicklungen der Romanik, wie etwa dem Strebwerk, dem Rippengewölbe oder dem Spitzbogen, der islamischen Ursprungs ist (Koch 2014: S. 149), konnte sich in der Gotik die Auflösung der Wand entwickeln. Hierfür war eine Errungenschaft der Gotik die Kombination des Skelett- und des Mauerprinzips und somit ein erstes sinnvolles Zusammenführen bisheriger Konstruktionsprinzipien (Fouad 2013: S. 30). Dies spiegelt sich vor allem im Sakralbau der Gotik wider.

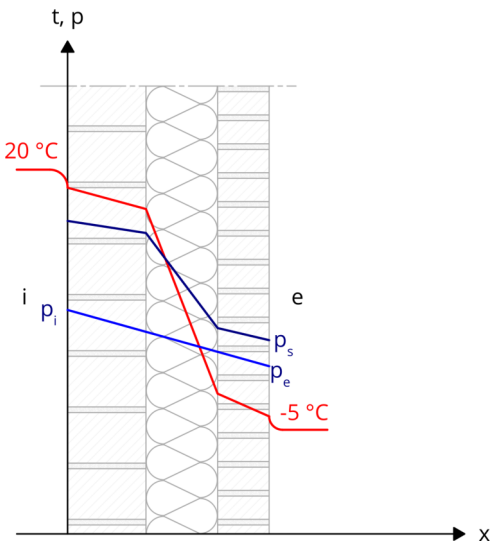
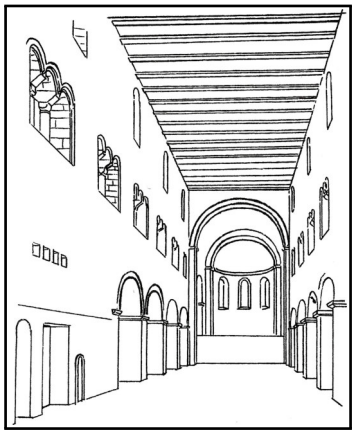
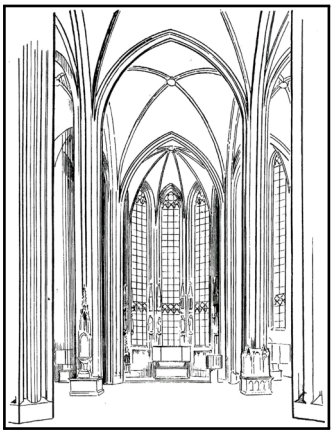


Abbildung 3: Schematisches Glaserdiagramm
Quelle: In Anlehnung an Sascha Pöschl (2008; URL: https://commons.wikimedia.org/wiki/File:Glaserdiagramm.svg (besucht am 31.08.2019))



(a) Prag, St. Georgskirche auf dem Hradschin, 1142



(b) Soest, Maria zur Wiese, um 1400

Abbildung 4: Vergleich zwischen romanischen (a) und gotischen (b) Sakralbauten
Quelle: Koch (2014: S. 138, 174)

Durch Rippen werden die Lasten der Gewölbe auf Pfeiler geleitet und von dort nach unten geführt. Nach und nach konnten durch diesen Fortschritt die von Kräften weitgehend befreiten Wände aufgelöst und große Fensterflächen entworfen werden. (Koch 2014: S. 155) Die Umlenkung der vertikalen Eigenlasten in die Stützen und Pfeiler erzeugt Horizontalkräfte, die zusätzlich abgefangen und nach unten geleitet werden müssen. Um dies umzusetzen, wurde die Bauwerksgeometrie geschickt aneinander gereiht und außenliegende Auflasten eingeführt. (Knaack et al. 2007: S. 16) Durch doppelte und dreifache Strebebögen, die zusätzlich von Fialen beschwert sind, konnten besonders hohe Strukturen errichtet werden (Koch 2014: S. 155). In der Abbildung 4 wird der Vergleich zwischen einem romanischen und einem gotischen Sakralbau verdeutlicht. Es ist zu erkennen, dass die Fensterfläche vergrößert und die Tragstruktur, bedingt durch die Stützen, filigraner wirkt. In der Gotik zeigt sich für den Massivbau eine erste Trennung der Funktionen. Die Wand wird nicht weiter als Haupttragstruktur verstanden, sondern vielmehr als gestalterisches Element, das Funktionen wie Belichtung und Sichtbeziehungen übernimmt.

2.2.3 Entwicklung von Verglasungen

Mit zunehmenden Wandöffnungen nimmt auch der Bedarf an transluzenten beziehungsweise transparenten Baustoffen zu. So kann die Wärmeenergie eines Raumes erhalten werden, ohne dabei auf die Belichtung verzichten zu müssen. Bereits in römischen Thermen wurden Marmortafeln verwendet, die so dünn geschnitten waren, dass sie transluzente Eigenschaften aufwiesen. Mit der Erfindung des Glases konnte zusätzlich zur Belichtung eine Sichtbeziehung zwischen Außen- und Innenraum hergestellt werden. Die Verglasungen wurden zunächst einlagig ausgeführt und technisch konnten anfangs nur kleine Glasschei-

ben gefertigt werden, sodass nur kleine Öffnungen mit Glas verschlossen werden konnten. Mit dem Einzug der Bleiverglasungen war es möglich größere verglaste Flächen zu erzeugen. Besonders die Fenster in Sakralbauten stechen durch ihre Farbigkeit hervor. (Knaack et al. 2007: S. 19) Bereits die Römer entwickelten ein Gießverfahren, das erstmals für die Produktion von flachem, aber kaum transparentem Glas geeignet war. Die Glasherstellung konnte durch das Zylinderstreckverfahren und dem darauf folgenden Mondglasverfahren so optimiert werden, dass flache, durchsichtige und klare Glasscheiben mit sehr glatten Oberflächen gefertigt werden konnten. Diese Produktionsverfahren waren bis Ende des 19. Jhdts. maßgebend. Die heutige Glasherstellung erfolgt über das „Floatglasverfahren" das 1959 von Pilkington erfunden wurde. Dabei wird die Glasschmelze bei circa 1000 °C auf ein Zinnbad geleitet, worauf hin sich das Glas gleichmäßig auf diesem ausbreiten kann. Dieses Verfahren ermöglicht eine kostengünstige Produktion von qualitativ hochwertigem Flachglas. (Herzog, Krippner und Lang 2004: S. 183; Heinze GmbH 2019b)

Neben den Einfachverglasungen entwickelten sich Kastenfenster. Diese bestehen aus zwei hintereinanderliegenden Glasschichten, sodass im Zwischenraum ein zusätzlicher Puffer entsteht. Der Scheibenzwischenraum ist dabei nicht hermetisch abgeriegelt, sodass ein Feuchtigkeitstransport von innen nach außen stattfinden kann. Kastenfenster sind so konzipiert, dass die innere Verglasung demontiert werden kann. Dies ermöglicht je nach Jahreszeit eine Anpassung an die Witterungsbedingungen. Aus dem Konzept des Kastenfensters entwickelte sich die Isolierverglasung. Hierbei wird der Scheibenzwischenraum stark reduziert, sodass eine hermetisch abgeriegelt Luft- oder Edelgasschicht entsteht. Diese Schicht hat aufgrund der eingeschränkten Luftzirkulation eine deutlich bessere Dämmwirkung als bei einem Kastenfenster. Heutzutage werden die beiden Glasscheiben mit Silikon auf eine Aluminiumschiene aufgeklebt, sodass ein fester Randverbund entsteht. (Knaack et al. 2007: S. 20 f.)

2.2.4 Fassaden der klassischen Moderne

Alle bisherigen Vorgänge beschreiben einen über mehrere Jahrtausende andauernden Prozess. Im Folgenden wird ein Zeitraum von circa 100 Jahren behandelt. Vor der klassischen Moderne konnten die Funktionen einer Wand weiter getrennt werden. Dies betrifft vor allem die Funktionen Tragen, Dichten und Durchblick. Jedoch konnte eine vollständige Trennung nicht vorgenommen werden, da weiterhin massive Wandkonstruktionen zur Errichtung von Bauwerken notwendig waren. Erst mit neuen Bautechniken in der klassischen Moderne war es möglich einzelnen Funktionen weiter zu trennen. Daraus resultiert die Entkoppelung der Gebäudehülle von der tragenden Struktur, sodass sich die Wand in die Fassade auflöst. Statisch relevante Stützen werden, wenn möglich nach innen, hinter die Fassaden

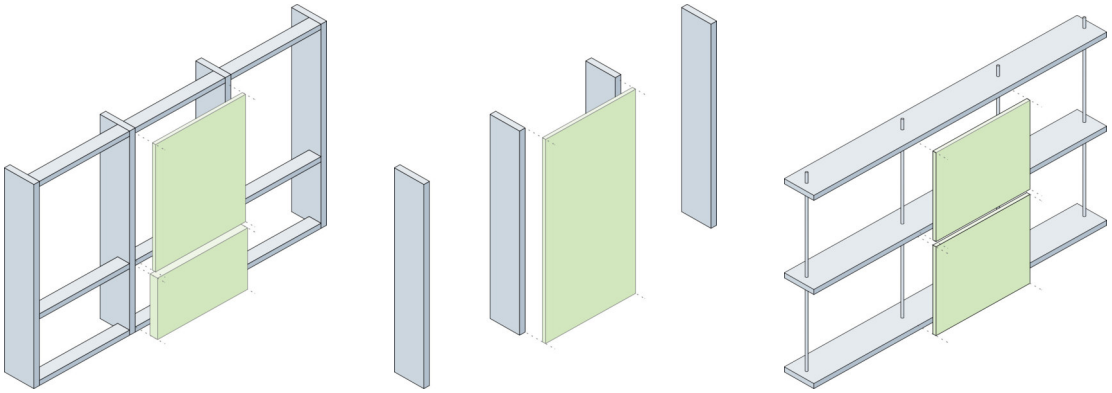
verschoben. So wurden Bauwerke wie das Haus Farnsworth von Ludwig Mies van der Rohe realisiert. Bei diesem Bauwerk besteht die komplette, hinter der Tragstruktur positionierte Hüllfläche aus Glas. (Knaack et al. 2007: S. 24) Die Fenster wurden in der klassischen Moderne meist als Einfachverglasungen ausgeführt. Diese finden sich beispielsweise in den Bauwerken der Weißenhofsiedlung in Stuttgart wieder, in der Architekten wie Ludwig Mies van der Rohe, Le Corbusier oder Walther Grophius mitgewirkt haben. Die Verglasungen befinden sich dabei bündig an der Außenkante der Gebäude und werden durch Stahlprofile eingefasst. (Knaack et al. 2007: S. 19)

2.3 Überblick über aktuelle Fassadensysteme

In diesem Kapitel sollen aktuell verwendete Fassadensysteme beschrieben werden. Bei der heutigen Entwicklung stehen gestiegene Anforderungen an die natürliche Belichtung und die Raumbezüge zwischen Innen- und Außenraum im Vordergrund. Damit geht eine weitere Vergrößerung der Fensterflächenanteile einher. Die massive Lochfassade mit zwischen Sturz und Brüstung integrierten Einzelfenstern stößt dabei schnell an ihre konstruktiven Grenzen und genügt nicht den höheren Anforderungen. (Hestermann und Rongen 2018: S. 573) Daher ist heute der Bedarf an Glasfassaden gestiegen, die hauptsächlich beim Leichtbau beziehungsweise bei der Skelettbauweise verwendet werden. Hierbei wird auf Grundlage der Aufgabenstellung ausschließlich Bezug auf diese Fassadenbauweisen genommen. Dies bedeutet, dass massive Fassadenkonstruktionen, wie etwa ein zweischaliges Mauerwerk mit Sichtmauerwerk oder ein WDVS nicht Bestandteil dieser Diplomarbeit sind.

2.3.1 Pfosten-Riegel-Fassaden

Knaack et al. (2007: S. 25) beschreiben Pfosten-Riegel-Fassaden als ein System von geschosshohen Pfosten und eingesetzten Riegeln. Durch diese Struktur entstehen Flächen, die von Pfosten und Riegeln eingeschlossen werden. Dieses System ist in Abbildung 5 (a) dargestellt. Den entstehenden Flächen können verschiedene Funktionen zugeordnet werden. Beispielsweise übernimmt eine Festverglasung die Funktion der Belichtung. Im Grunde können nach Knaack et al. (2007: S. 25) durch eine Pfosten-Riegel-Fassade vor allem die Funktionen Ausfachen, Belichten, Lüften und der Lastabtrag von Windlasten und Eigenlasten durch die Pfosten übernommen werden. Dem ist zu ergänzen, dass weitere speziellere Aufgaben wie etwa die Erzeugung von elektrischem Strom mittels gedämmten Paneelen mit integrierter Photovoltaik denkbar sind (Hestermann und Rongen 2018: S. 585; vgl. Weller und Fischer 2013).



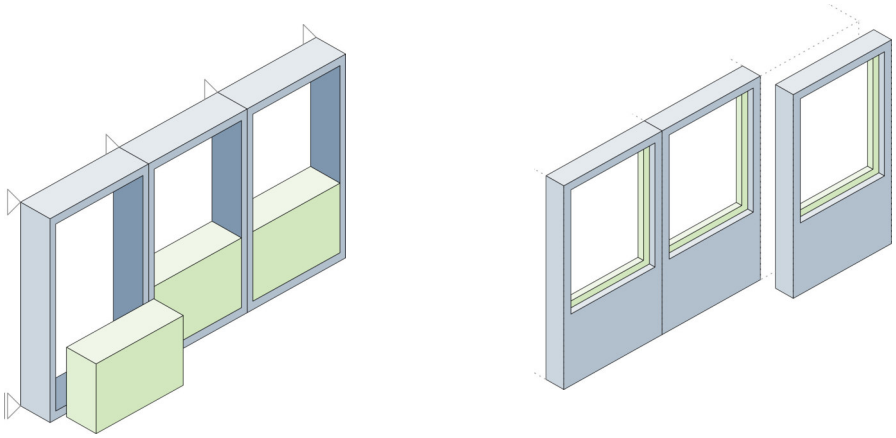
(a) Pfosten-Riegel-Fassade (b) Pfosten-Fassade (c) Riegel-Fassade

Abbildung 5: Pfosten-Riegel-Fassade (a) und Sonderformen (b,c)
Quelle: In Anlehnung an Knaack et al. (2014: S. 25 f.)

Eine Sonderform der Pfosten-Riegel-Fassade ist zum einen die in Abbildung 5 (b) dargestellte Pfosten-Fassade. Bei dieser Konstruktion werden ausschließlich die vertikalen Pfosten verwendet. Dadurch wirkt die Konstruktion filigraner als bei Pfosten-Riegel-Fassaden. Eine weitere Abwandlung ist die Riegel-Fassade. In Abbildung 5 (c) ist eine Darstellung dieses Fassadensystems gegeben. Diese zeichnet sich dadurch aus, dass die lastabtragenden Pfosten durch Zugstangen ersetzt werden. Da die Zugstangen nicht auf Knicken beansprucht werden können, werden im Gegensatz zur Pfosten-Riegel- und Pfosten-Fassade die Fassaden hängend und nicht stehend verbaut. Horizontale Lasten, wie etwa Wind- oder Verkehrslasten, werden dabei durch die Riegel abgetragen. Das Entwicklungsziel dieser Sonderformen ist es, den Anteil der Verglasung im Bezug zur Konstruktionsfläche zu erhöhen, um eine größere Transparenz zu erzielen. (Knaack et al. 2014: S. 26)

2.3.2 Vorhangfassaden

Nach DIN EN 13830:2015-07 (S. 11) werden Vorhangfassadenkonstruktionen als „[...] Teil der Gebäudehülle, die [...] aus miteinander verbundenen horizontalen und vertikalen Profilen besteht, mit der tragenden Konstruktion des Baukörpers verankert ist und mit fest eingebauten und/oder zu öffnenden Ausfachungen ausgestattet ist [...]“ definiert. Dabei soll gelten, dass die Vorhangfassade die Funktionen einer Innen- oder Außenwand vollständig oder zumindest teilweise übernimmt, jedoch nicht zur Statik des Gebäudes beiträgt. Somit sind diese Fassaden selbsttragend konstruiert. Nur die Kräfte aus Eigengewicht, Nutzlasten, Umgebungslasten (Wind, Schnee, etc.) sowie seismische Lasten werden von der Konstruktion aufgenommen und an das Haupttragwerk des Bauwerks weitergeleitet. (DIN EN 13830:2015-07: S. 11) Die Lastabtragung dieser Fassaden erfolgt meist geschossweise als hängendes System. Dadurch werden im Vergleich zu den stehenden Systemen, wie etwa



(a) Vorhangfassade (b) Elementfassade

Abbildung 6: Vorhangfassaden
Quelle: In Anlehnung an Knaack et al. (2014: S. 27 f.)

einer Pfosten-Riegel-Fassade, Probleme mit Knicken umgangen. Darüber hinaus besteht keine konstruktive Abhängigkeit zum restlichen Tragwerk des Gebäudes, sodass die Fassade freier gegliedert und gestaltet werden kann. In Abbildung 6 (a) ist das Konzept der hängend konstruierten Vorhangfassade dargestellt. Es ist zu erkennen, das die einzelnen Felder mit verschiedenen Komponenten gefüllt werden können. (Knaack et al. 2014: S. 27 f.)

2.3.3 Elementfassaden

Elementfassaden sind eine Unterkategorie der Vorhangfassaden. Dieses Fassadensystem bietet die Möglichkeit zur teilweisen Vorfertigung und der endgültigen baustellenseitigen Montage oder der vollständigen Vorfertigung, sodass die Fassadenelemente direkt installiert werden können. Durch die werksseitige Vorfertigung kann eine konstante Produktqualität garantiert und eine zeit- und personalintensive Montage vermieden werden. Dieser Vorteil kann vor allem bei Gebäuden genutzt werden, bei denen die gleichen Elemente möglichst häufig verbaut werden. Dadurch haben sich Elementfassaden vor allem für das Errichten von Hochhausfassaden durchgesetzt. Bei abweichenden Gegebenheiten, wie flache Gebäude mit guter Zugänglichkeit um das Gebäude, sind die Vorteile einer Elementfassade zu prüfen. In Abbildung 6 (b) sind die vorgefertigten Fassadenelemente schematisch dargestellt. (Knaack et al. 2014: S. 28)

2.3.4 Doppelfassaden

Durch den Wunsch, weitere Funktionen an die Fassade auszulagern, entstehen die Doppelfassaden. Dadurch ist es möglich, beispielsweise die Belüftung nicht zwischen Innen- und Außenraum, sondern innerhalb der Fassadenebene vorzunehmen. Durch die kontrollierten Bedingungen im Fassadenzwischenraum kann die entstehende natürliche Thermik zur gezielten Ent- und Belüftung genutzt werden. Bei diesem Konzept haben sich geschosshohe und sich über mehrere Geschosse erstreckende Lüftungssysteme entwickelt. (Knaack et al. 2014: S. 29) In Abbildung 7 (a) - (d) sind vier verschiedene Doppelfassaden dargestellt.

Mittlerweile haben erste Erfahrungen gezeigt, dass der Einsatz von Doppelfassaden nicht zwangsläufig in alle Himmelsrichtungen notwendig oder gar sinnvoll ist. Für einige Anwendungsfälle, wie etwa bei hohen Lärmimmissionen oder großen Gebäudehöhen, kann eine Doppelfassade wirtschaftlich sinnvoll sein. (Knaack et al. 2014: S. 29)

2.3.4.1 Zweite-Haut-Fassaden

Die Gliederung von Doppelfassaden kann in vier Hauptprinzipien unterteilt werden. Als einfachste Lösung gilt dabei die Zweite-Haut-Fassade. Diese ist schematisch in Abbildung 7 (a) dargestellt. Hierbei wird der isolierenden Gebäudehülle eine zweite einfachverglaste Ebene vorgesetzt, die nur im Fuß- und Kopfbereich belüftet wird. Gegenüber den anderen Gliederungsmöglichkeiten besitzen Zweite-Haut-Fassaden einige Nachteile. Nennenswert sind hierbei die geringen Steuerungsmöglichkeiten im Gebäudebetrieb und das damit einhergehende Überhitzungsrisiko. (Knaack et al. 2014: S. 30)

2.3.4.2 Kastenfenster-Fassaden

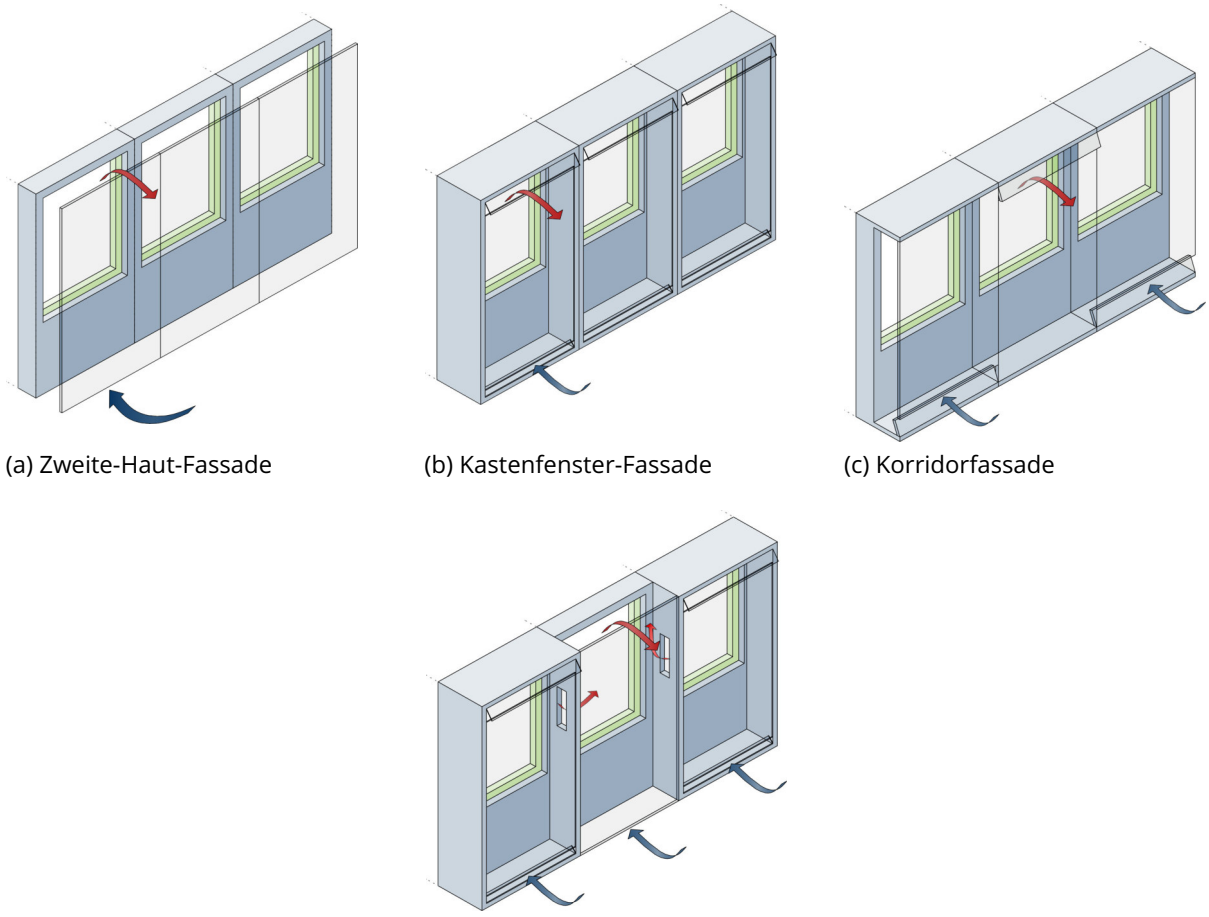
Eine weitere Gliederungsform ist das in Abbildung 7 (b) dargestellte Prinzip des Kastenfensters. Hierbei wird die vorgesetzte Ebene horizontal und vertikal unterbrochen. Die Unterteilungen ergeben sich dabei durch die Größe der Fassadenelemente, die in der Regel geschosshoch sind. Belüftungsmöglichkeiten sind am oberen (Abluft) und unteren (Zuluft) Rand vorgesehen. Je nach Gebäudegrundriss entsteht so eine raumweise Belüftung, die individuell geregelt werden kann. Als Nachteil dieser Methode wird vor allem die Positionierung der Zuluftöffnung direkt über der Abluftöffnung des darunter liegenden Raumes genannt. Dadurch kann die Zuluftqualität einzelner Räume oder ganzer Geschosse stark beeinträchtigt werden. Ein Lösungsansatz für dieses Problem ist eine seitlich versetzte Anordnung der Zu- und Abluftöffnungen. (Knaack et al. 2014: S. 30)

2.3.4.3 Korridorfassaden

Bei der Korridorfassade erfolgt ausschließlich eine geschosshohe horizontale Gliederung, sodass nebeneinanderliegende Räume ein gemeinsames Belüftungssystem bilden. Durch versetzte Zu- und Abluftöffnungen können Lüftungskurzschlüsse wie bei der Kastenfenster-Fassade vermieden werden. Nachteilig ist bei diesem Konzept, dass es zu störenden Schallreflexionen kommen kann. Das Konzept dieser Fassade ist in Abbildung 7 (c) gegeben. (Knaack et al. 2014: S. 31)

2.3.4.4 Schacht-Kasten-Fassaden

Die in Abbildung 7 (d) dargestellte Schacht-Kasten-Fassade stellt die effizienteste, aber steuertechnisch und konstruktiv aufwendigste Variante der Doppelfassaden dar. Bei diesem Konzept werden raumweise Kastenfenster angeordnet und in einen vertikalen Fassadenschacht entlüftet. Dieser Schacht ist dabei geschossübergreifend und aufgrund der Höhe



(a) Zweite-Haut-Fassade (b) Kastenfenster-Fassade (c) Korridorfassade

(d) Schacht-Kasten-Fassade

Abbildung 7: Doppelfassaden
Quelle: In Anlehnung an Knaack et al. (2014: S. 29-32 f.)

des Schachtes bildet sich ein natürlicher Kamineffekt, sodass eine vertikale Luftströmung entsteht. Damit die Belüftung jedes einzelnen Raumes steuerbar ist, werden für jedes Kastenfenster Steuerklappen zum Schacht hin benötigt. (Knaack et al. 2014: S. 32)

2.3.4.5 Weitere Doppelfassaden

Neben den vier Hauptprinzipien haben sich weitere Formen von Doppelfassaden entwickelt. Nennenswert sind hierbei die Wechselfassade und die Komponentenfassade. Entwicklungsziel dieser Konstruktion ist es, entweder die Probleme in den bestehenden Systemen zu kompensieren oder weitere Funktionen an die Fassade zu übertragen. Im Rahmen dieser Arbeit wird nicht weiter auf diese Konstruktionen eingegangen, es sei jedoch auf die Literatur von Knaack et al. (2014: S. 33 ff.) verwiesen.

2.4 Konstruktionsdetails

Nachdem in Kapitel 2.3 ein grundlegender Überblick über verschiedene Fassadensysteme gegeben wurde, soll im Folgenden näher auf einschalig konstruierte Fassadensysteme eingegangen werden, da diese potenziell zur Errichtung eines parametrischen Fassadensystems geeignet sind. Dabei wird der Fokus dieser Arbeit auf die Pfosten-Riegel-Fassaden und die Elementfassaden gesetzt, da alle weiteren Fassadenarten Spezialfälle darstellen.

2.4.1 Pfosten-Riegel-Fassaden

Wie bereits in Kapitel 2.3.1 beschrieben sind Pfosten-Riegel-Fassaden ein aus Pfosten und Riegeln bestehendes Stabsystem. Die Pfosten sind vertikal, meist geschosshoch angeordnet und werden durch horizontale Riegel miteinander verbunden. Dabei entstehen einzelne Felder. Diese Felder können mit verschiedenen Ausfachungen, wie etwa einer festen Verglasung, geschlossenen Elementen oder Fenster- beziehungsweise Türöffnungen versehen werden. (Hestermann und Rongen 2018: S. 573)

2.4.1.1 Fixierung der Fassadenelemente

Eine mechanische Verbindung zwischen den ausfachenden Elementen und den Tragprofilen wird in den meisten Fällen über lineare Metall-Pressleisten erreicht. Diese werden punktuell verschraubt, sodass die Ausfachungen mechanisch gehalten werden. (Hestermann und Rongen 2018: S. 573) Alternative Fixierungskonzepte sind die Glashalteleiste, die Structural Sealant Glazing (SSG)-Fassade sowie Ganzglasfassadensysteme (Hestermann und Rongen 2018: S. 586). Diese drei Konzepte sind in Abbildung 8 (b - d) dargestellt. Bei

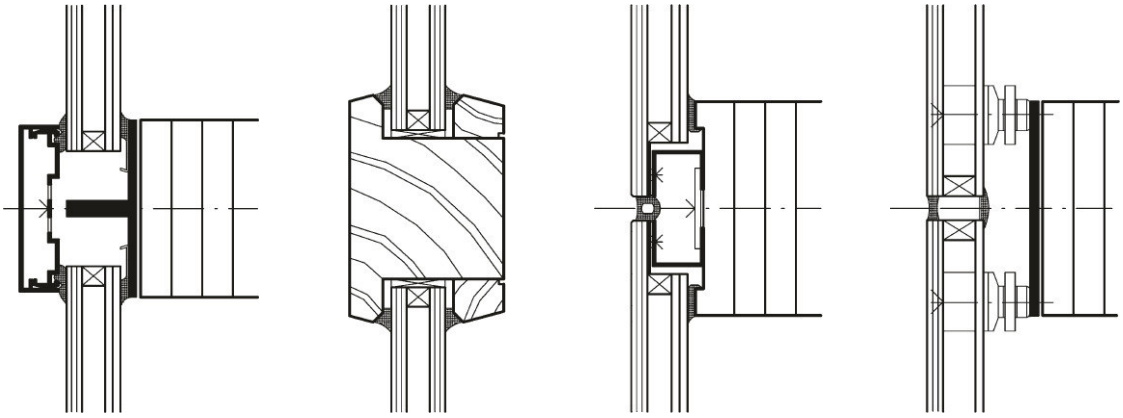


Abbildung 8: Fixierungskonzepte von Glasfassaden
Quelle: Hestermann und Rongen (2018: S. 586)

dem SSG-System, dargestellt in Abbildung 8 (c), erfolgt die Befestigung der Glaselemente ohne die Verwendung von Press- oder Abdeckleisten. Dadurch können homogene, großflächige Glasoberflächen erzeugt werden, die nur durch sehr schmale Fugen unterbrochen werden. Hierbei erfolgt die Fixierung über Klebung der äußeren Scheibe an das Tragprofil. Somit ist die SSG-Fassade nicht mechanisch gehalten. In Deutschland ist eine bauaufsichtliche Zulassung zur Errichtung solcher Fassaden nötig und erfordert das Anbringen von mechanischen Sicherungen, die die Glasscheiben beim Versagen der Klebung halten. Für Gebäude unter 8,00 m Höhe kann die Fassade ohne diese zusätzliche mechanische Sicherung errichtet werden. (Hestermann und Rongen 2018: S. 587 ff.) In Abbildung 8 (d) wird der Profilquerschnitt einer Ganzglasfassade mit Pfosten-Riegel-Tragkonstruktion dargestellt. Der Verbund zwischen Verglasung und Tragprofil wird in diesem Fall über Punkthalter hergestellt. (Hestermann und Rongen 2018: S. 586) Wie bereits erwähnt, werden Pfosten-Riegel-Fassaden meist mit Pressleisten errichtet. Dieses Konzept ist in Abbildung 8 (a) dargestellt. Es liegt darin begründet, dass nicht nur Glaselemente, sondern auch weitere Füllelemente – wie etwa Sandwichpaneele – mithilfe der Pressleisten montiert werden können. (Hestermann und Rongen 2018: S. 586) So können verschiedene Elemente in einer Fassade kombiniert werden. Im weiteren Verlauf dieser Arbeit wird, aufgrund des hohen Verbreitungsgrades, der Fokus auf die Pfosten-Riegel-Fassaden mit Pressleisten gesetzt.

2.4.1.2 Abdichtung und Entwässerung

Die Kombination der Ausfachungen in Verbindung mit den Pressleisten stellt in der Pfosten-Riegel-Fassade die Wetterschutzschicht dar (Knaack et al. 2007: S. 59) und sollte somit zwischen allen Pressleisten und allen verbauten Ausfachungselementen wind- und schlagregendicht ausgeführt werden (Hestermann und Rongen 2018: S. 586). Dazu werden die Pressleisten mit Abdichtungen versehen, sodass das Eindringen von Wasser in das Gebäudeinnere verhindert wird. Hierbei gilt zu beachten, dass bei den meisten Fassadensystemen ein mehrstufiges Abdichtungssystem vorliegt. Dies bedeutet, dass ein Wassereintritt durch die äußeren Abdichtungen der Pressleisten vorgesehen ist. Ein inneres Dichtungssystem an den Tragprofilen verhindert, dass das Wasser den Innenraum erreicht. (Knaack et al. 2007: S. 45) Nach Hestermann und Rongen (2018: S. 586) kann das Abdichtungssystem entweder diffusionsoffen oder diffusionsdicht ausgebildet werden. Bei diffusionsoffenen Systemen ist ein Dampfdruckausgleich zwischen innen und außen möglich. Diffusionsdichte Systeme verhindern dies. In jedem Fall sollte das Abdichtungssystem so ausgebildet werden, dass ein störungsfreier Dampfdruckausgleich sowie die Entwässerung der Profil-

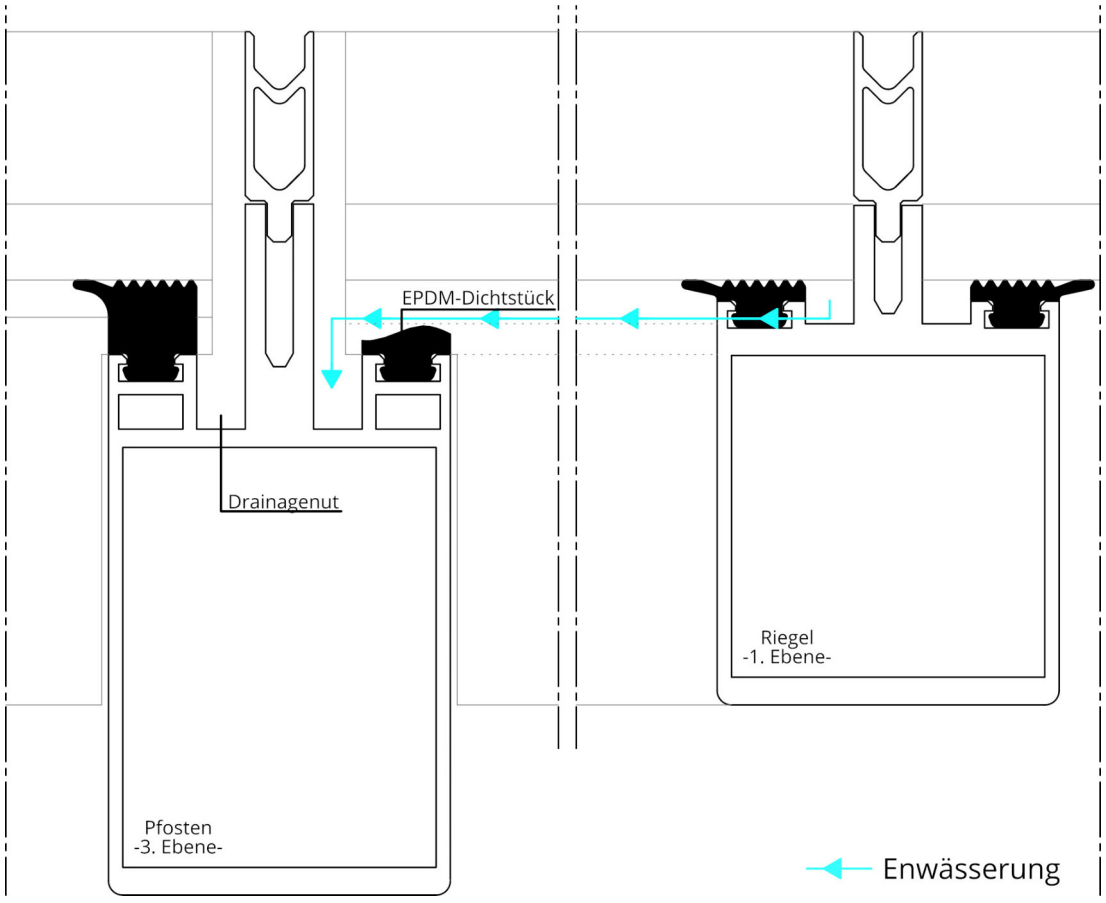
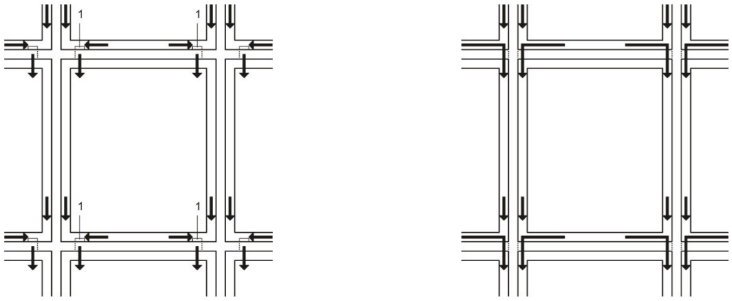


Abbildung 9: Entwässerungskonzept des Fassadensystems „FW 50+“
Quelle: In Anlehnung an Schüco International KG (2011: S. 48)

hohlräume gewährleistet wird. Um das Eindringen von warmer, feuchter Innenraumluft und das entgegengesetzte Eindringen von Regenwasser an die Tragstruktur zu vermeiden, sollte die innere Abdichtungsebene dampfdicht ausgeführt werden. Diese Ansprüche an das Dichtungssystem sind insbesondere im Bereich der Knotenpunkte einzuhalten. (Hestermann und Rongen 2018: S. 586) Durch das mehrstufige Abdichtungssystem sammeln sich entstehendes Kondensat und eindringendes Regenwasser innerhalb der Profile. Die Entwässerung der Pfostenprofile erfolgt, indem das Wasser nach unten zum Fußpunkt geleitet wird. Dort wird es aus der Fassade geführt, beispielsweise in einer dafür vorgesehene Drainageleitung. In Abbildung 9 ist beispielhaft das vorgesehene Entwässerungskonzept des Pfosten-Riegel-Fassadensystems „FW 50+" der Firma Schüco dargestellt. In diesem System bilden sich mehrere Entwässerungsebenen, die das Wasser durchlaufen muss. Die erste Ebene wird von nicht bis in den Fußpunkt durchgängige vertikale Riegelprofile gebildet. Diese kommen beispielsweise oberhalb von Türöffnungen vor. Das Wasser der ersten Ebene wird in die zweite Ebene, die Ebene der Riegel, geleitet. Die dritte Ebene stellen die durchgängigen Pfosten dar. Die Höhe der Drainagekanäle nimmt bei jeder Ebene zu, sodass alle Profile der einzelnen Ebenen auf einer anderen Höhe liegen. Dieser Höhenunterschied wird mithilfe von verschieden hohen Dichtungen ausgeglichen. (Schüco International KG 2011: S. 48) Andererseits sind auch Systeme auf dem Markt, bei denen die Ebenen in den Dichtungen integriert sind. Beispielhaft kann dies an den Aluminiumfassadensystemen der Firma Stabalux GmbH gezeigt werden. Die Dichtungen sind dabei so konzipiert, dass die Dichtungen aller Ebenen die gleichen Einbauhöhen aufweisen. Die Tragprofile werden so auf der gleichen Höhe verbaut. (vgl. Stabalux GmbH 2019: S. 9 f.) Damit die Entwässerung von Pfosten-Riegel-Fassaden funktionieren kann, muss im Fußpunkt der Fassade eine Zuluft- und im Kopfpunkt eine Abluftöffnung vorgesehen werden. Zusätzlich sind alle 6,00 m weitere Zuluftöffnungen vorzusehen. Dichtungen in vertikaler Richtung sind bestenfalls durchgängig auszuführen. Die Entwässerung der Riegelprofile kann auf zwei Arten erfolgen. Die ersten Variante, die in Abbildung 10 (a) dargestellt ist, entwässert feldweise. Dies bedeutet, dass vor jedem Knotenpunkt die äußere Abdichtung der Riegel nach unten hin geöffnet ist, sodass anfallendes Wasser in jedem Feld einzeln das Profilsystem verlässt. Die zweite, in Abbildung 10 (b) dargestellte Möglichkeit besteht darin, das gesamte Wasser der Riegel in die Pfosten und von dort aus nach unten zu leiten. In beiden Fällen sind die Knotenbereiche durch Dicht- und Klebstoffe oder durch Stauchdichtungen abzudichten. Die Dichtungen werden in diesem Bereich zudem überdimensioniert, da es zu Schrumpfungsprozessen kommen kann. (Hestermann und Rongen 2018: S. 586 f.)



(a) feldweise (b) über den Pfosten

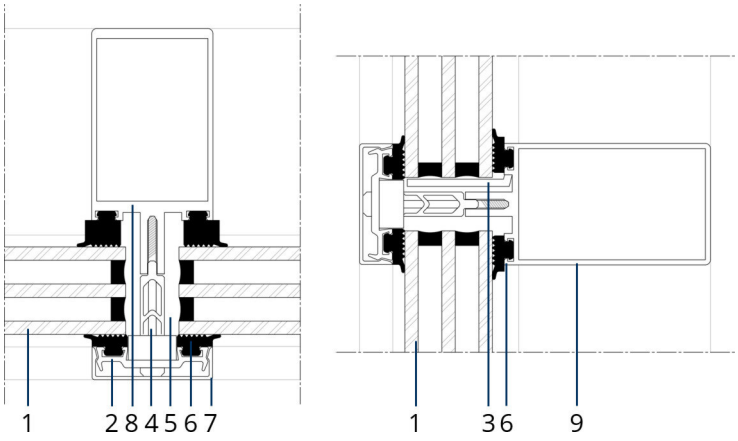
1 Entwässerungsöffnung

Abbildung 10: Entwässerung einer Pfosten-Riegel-Fassade
Quelle: Hestermann und Rongen (2018: S. 588)

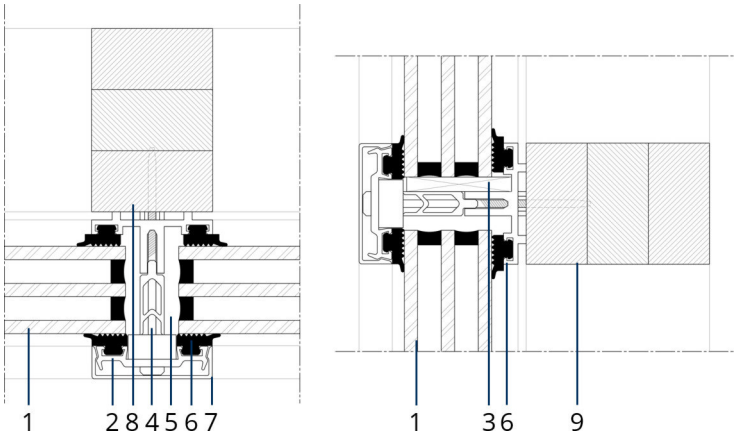
2.4.1.3 Bauteilkomponenten

In Abbildung 11 (a-c) sind exemplarische Querschnitte verschiedener Materialien gegeben. Die Tragprofile der Pfosten (9) und der Riegel (8) befinden sich an der Innenseite der Fassade. Diese Profile sind hauptsächlich für den Lastabtrag verantwortlich. Die Ausfachungselemente (in diesem Fall: Verglasung) (1) werden mithilfe eines Druckprofils (auch: Pressleiste) (2) durch eine Schraube (4) an die Tragprofile gepresst, sodass ein linearer Anpressdruck entsteht, der die Elemente mechanisch befestigt. Damit die Schrauben nicht sichtbar sind und um die Schmutzanfälligkeit zu reduzieren, können Deckprofile (auch: Blendleisten) (7) auf die Druckprofile gesteckt werden. Dabei ist zu beachten, dass wie in Abbildung 11 (b) die Druck- und Deckleiste unabhängig von der Unterkonstruktion wählbar ist. Zwischen den Ausfachungselementen und den Profilen wird zur Erzeugung der erforderlichen Dichtigkeit und Elastizität eine Dichtung (6) verbaut. Bei Verwendung einer Isolierverglasung sollte ein Isolierprofil (auch: Isolator) (5) vorgesehen werden, um die Tragprofile von der Pressleiste thermisch zu trennen, wie in Abbildung 11 (a) und (b) zu sehen ist. Zur Lagerung der Verglasung ist in vertikaler Richtung eine Verklotzung (3) anzuordnen. (Schittich et al. 2006: S. 164 f.)

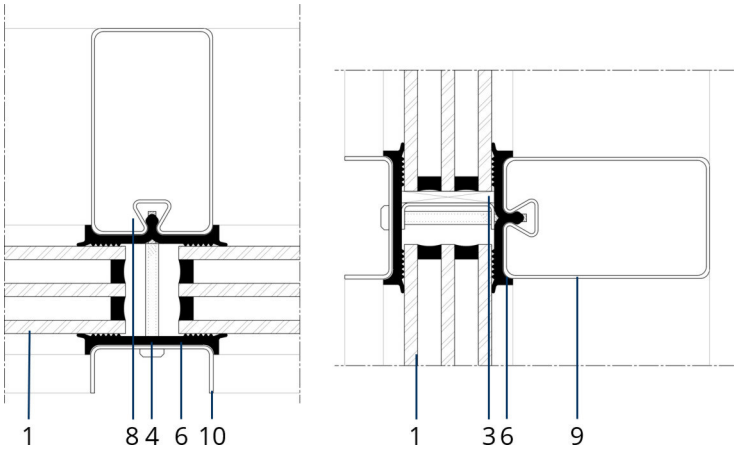
Die Profile der Pfosten-Riegel-Fassade sind in der Regel zwischen 35 und 80 mm breit. Als Standardwerte werden hierbei die Breiten 50 und 60 mm betrachtet. Die Profiltiefe ist hauptsächlich von der statischen Belastung abhängig, können aber auch nach gestalterischen Anforderungen gewählt werden, solange das statische Gleichgewicht eingehalten wird. Gefertigt werden die Profile aus Aluminium, Stahl oder Holz. Es stehen einige standardisierte Profilsysteme verschiedener Hersteller zur Verfügung. Sollten diese den Anforderungen nicht genügen, können auch Sonderformen aus Metall oder Holz entwickelt werden. So kann die Fassade auch weitere Zusatzfunktionen übernehmen, wie etwa integrierte Installationskanäle für Elektroleitungen zur Steuerung von beweglichen Sonnschutzvorrichtungen. (Hestermann und Rongen 2018: S. 573, 580, 582)



(a) Aluminium



(b) Holz/Aluminium



(c) Stahl

1	Verglasung	4	Schraube	7	Deckprofil (Blendleiste)
2	Druckprofil (Pressleiste)	5	Isolierprofil (Isolator)	8	Riegel
3	Verklotzung	6	Dichtung	9	Pfosten

Abbildung 11: Querschnitte einer Pfosten-Riegel-Fassade aus verschiedenen Materialien (Links: Riegelschnitt, Rechts: Pfostenschnitt)
Quelle: In Anlehnung an Schittich et al. (2006: S. 164 f.)

Die Dichtungen sind meist aus Silikon oder EPDM/APTK hergestellt (Schittich et al. 2006: S. 164 f.). Je nach gewählter, beziehungsweise aus Vorgaben hinsichtlich des Wärmeschutzes resultierenden Verglasungen können unterschiedliche Dichtungen verbaut werden. (Knaack et al. 2014: S. 45). Dies hat den Vorteil, dass unterschiedliche Elementdicken innerhalb einer Fassade verbaut werden können. Durch verschiedene Dichtungsgrößen können die entstehenden Differenzen ausgeglichen werden. Zudem stehen weitere Einsetzprofile zur Verfügung, sollten die verschiedenen Dichtungsgrößen nicht ausreichen. (Schüco International KG 2011: S. 48)

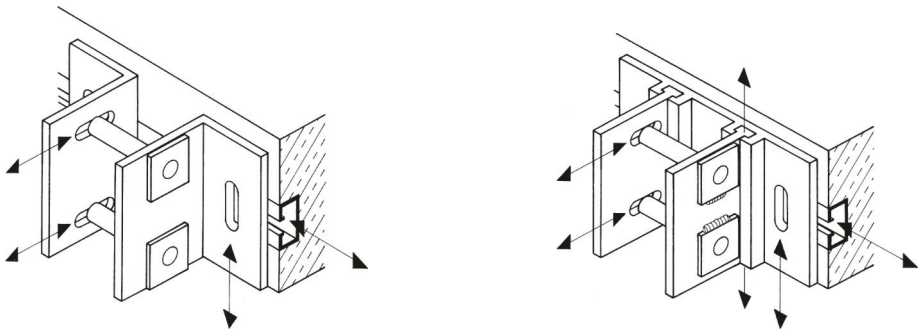
Die verbauten Verglasungen, beziehungsweise andere Ausfachungselemente, werden nach statischen, bauphysikalischen, konstruktiven und sicherheitstechnischen Aspekten gewählt. Dabei kommen heutzutage Floatglasscheiben zum Einsatz. Dieses Verfahren wurde in Kapitel 2.2.3 erläutert. Ausgeführt werden diese meist als Einscheibensicherheitsglas (ESG) oder Verbundsicherheitsglas (VSG). Aufgrund der Anforderungen an den Wärmeschutz werden die Scheiben in den meisten Fällen zu Isolierverglasungen zusammengefügt. Dabei entsteht ein Scheibenzwischenraum (SZR) unterschiedlicher Dicke, der je nach Füllung verschiedene Dämmeigenschaften aufweist. Als Füllgase können hier normale Luft oder Edelgase wie Argon, Krypton oder Xenon verwendet werden. Die Verglasung kann zudem mittels Beschichtungen weiter verbessert werden. (Hestermann und Rongen 2018: S. 585) Hierbei sind die Low-E-Beschichtungen und Sonnenschutzbeschichtungen nennenswert. Eine Low-E-Beschichtung verbessert den U-Wert einer Scheibe und reduziert damit den Wärmeverlust durch die Verglasung (Heinze GmbH 2019a). Eine Sonnenschutzbeschichtung hingegen reduziert den Energieeintrag aufgrund solarer Einstrahlung durch transparente Bauteile, indem mehr Solarstrahlung reflektiert wird (Heinze GmbH o.D.). Die Ausfachungselemente können dabei mit unterschiedlichen gestalterischen Eigenschaften gewählt werden. Hestermann und Rongen (2018: S. 585) nennen hierbei folgende Varianten:

- Transparente Verglasung
- Transluzente Verglasung
- Opake Verglasung
- Sandwichpaneel
- Fenster / Tür
- Weitere Funktionen – beispielsweise PV-Modul

Die Größe für Scheiben liegt für Pfosten-Riegel-Fassaden in der Regel bei einer Seitenlänge von 1,20 bis 2,40 m (Hestermann und Rongen 2018: S. 585). Es können auch größere Scheiben verbaut werden, dies führt allerdings zu einer Transportproblematik. Scheibengrößen bis 3,21 x 6,00 m sind problemlos herstellbar. Die Länge kann bis über 12 m erhöht werden (Heinze GmbH 2019b). Für die Breite gilt, dass Sonderfertigungen über 3,21 m kostenintensiv sind, da die Maschinengröße der meisten Glashersteller auf diese Größe beschränkt ist. Da das Glas am Stück hergestellt wird, ist eine Verlängerung nicht ganz so kostenintensiv. Gelagert werden die Glasscheiben innerhalb der Fassade auf zwei Verklotzungen, in der Regel bestehend aus Polyamid (Hestermann und Rongen 2018: S. 585).

2.4.1.4 Bauwerksanschluss

Eine Pfosten-Riegel-Fassade besteht aus werksseitig vorgefertigten Halbzeugen. Diese werden vor Ort montiert. Der Bauwerksanschluss der Pfosten-Riegel-Fassade wird über Anschlussschuhe ausgeführt. Die Montage erfolgt meist an den Geschossdecken. Damit werden die einzelnen Komponenten der Pfosten-Riegel-Fassade unabhängig vom Haupttragwerk des Gebäudes errichtet. Dadurch können mögliche Toleranzen des Rohbaus ausgeglichen werden. Nachteilig stellt sich dabei jedoch heraus, dass zwischen Geschossdecke und Pfosten-Riegel-Fassade ein Hohlraum entsteht, der nachträglich verschlossen werden muss, um den Brand- und Schallschutz sicher zu stellen. (Knaack et al. 2014: S. 59) Nach Hestermann und Rongen kann die Tragwerksplanung von Pfosten-Riegel-Fassaden sehr vielfältig ausgeführt werden. Hierbei wird in zwei Prinzipien unterschieden. Zum einen können die Lasten aus der Fassade direkt in das Haupttragwerk abgeleitet werden. Bei dieser Variante ist nachteilig, dass der Abstand zwischen Primärtragwerk und der Fassade nur sehr gering ausfallen kann und dass besondere Anforderungen an die Toleranzen des Rohbaus gestellt werden müssen. Andererseits können selbsttragenden Sekundärstrukturen verwendet werden, die die Lasten an das Haupttragwerk übertragen. Das hat den Vorteil,



(a) Festlager (b) Loslager

Abbildung 12: Lager einer Pfosten-Riegel-Fassade
Quelle: Hestermann und Rongen (2018: S. 579 f.)

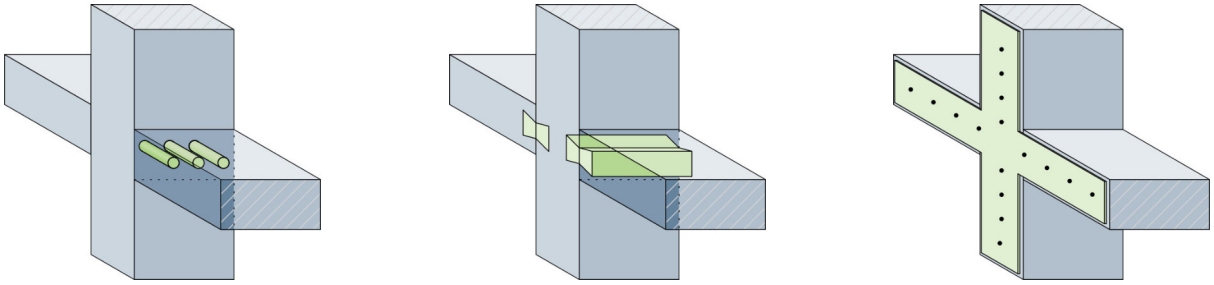
dass durch die Trennung vom Primärtragwerk eine freiere Gestaltung des Fassadenrasters möglich ist und so der Fensterflächenanteil optimiert werden kann. (Hestermann und Rongen 2018: S. 576)

Der Lastabtrag einer Pfosten-Riegel-Fassade erfolgt immer in eine Richtung. Bei stehenden Fassaden ist die Abtragsrichtung nach unten und bei hängenden nach oben. Dazu wird die Fassade oben beziehungsweise unten durch einen Festpunkt gehalten. Um Bewegungen der Fassade zu verhindern, muss diese zusätzlich mindestens an einem Lospunkt befestigt werden. Bei diesem muss sichergestellt werden, dass eine Dilatation ohne die Erzeugung von Spannungen erfolgen kann. Als Dilatation wird dabei die Verschiebung der Bauteile aufgrund von Temperaturschwankungen bezeichnet. Somit muss der Anschluss mit Langlöchern oder Anschlussschienen ausgeführt werden. Wie in Abbildung 12 verdeutlicht, sind die Bauwerksanschlüsse in drei Dimensionen justierbar, sodass Toleranzen ausgeglichen werden können. Es ist in Abbildung 12 (b) auch dargestellt, wie sich das Loslager in Richtung der Pfostenprofile verschieben kann, um Eigenspannungen infolge thermischer Dehnung zu vermeiden. Die Bauwerksanschlüsse werden bestenfalls bereits im Rohbau vorgesehen und entweder Ankerplatten oder Ankerschienen mit Ankerschrauben in die Geschossdecken einbetoniert. Anderenfalls besteht die Möglichkeit einer nachträglichen Montage durch die Verwendung von Schwerlastankern. (Hestermann und Rongen 2018: S. 578 f.)

2.4.1.5 Ebene Fassadenknoten

Im Folgenden sollen die Verbindungsstellen zwischen Pfosten und Riegeln näher erläutert werden. Hierbei muss zwischen Holz und den beiden anderen Werkstoffen Aluminium beziehungsweise Stahl unterschieden werden. Eine kraftschlüssige Verbindung kann bei Tragprofilen aus Holz mit Holzdübeln, Schwalbenschwanzverbindungen, Stahl-Verbindungsmitteln oder in der Verglasungsebene liegenden kreuz- beziehungsweise T-förmigen Flachstählen realisiert werden. In Abbildung 13 (a) - (c) sind einige dieser Verbindungen dargestellt. (Hestermann und Rongen 2018: S. 583)

Für Aluminium und Stahl können generell drei verschiedene Lösungsansätze zum Verbinden der Pfosten und Riegel genutzt werden. Wie Hestermann und Rongen schreiben, können die Profile direkt miteinander oder durch ein zusätzliches T-Stück verschraubt werden. Baitinger, Busse und Mauser ergänzen hierfür die Möglichkeit die beiden Profile miteinander zu verschweißen. Für Letztere, so Baitinger, Busse und Mauser, kann der Nachweis der Tragfähigkeit nach den Regeln der geltenden technischen Bestimmungen durchgeführt werden. (Hestermann und Rongen 2018: S. 584; Baitinger, Busse und Mauser 2006:



(a) Dübel (b) Schwalbenschwanz (c) Flachstahl

Abbildung 13: Knoten einer Pfosten-Riegel-Fassade mit Holztragstruktur
Quelle: In Anlehnung an Hestermann und Rongen (2018: S. 583 f.)

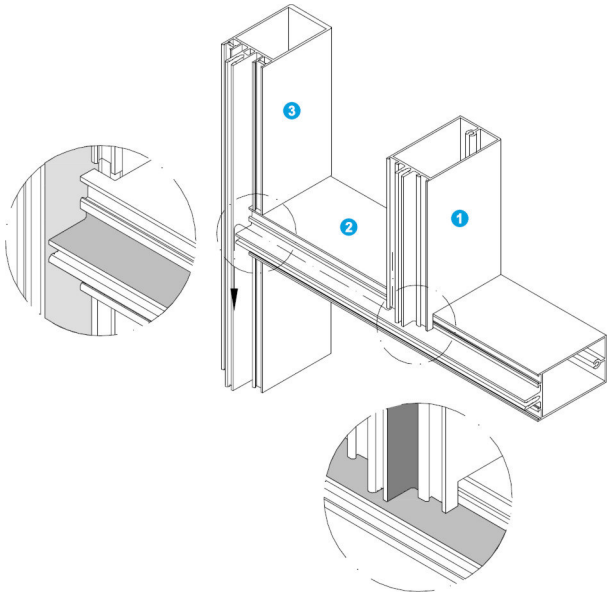


Abbildung 14: Ausbildung von Fassadenknoten
Quelle: Schüco International KG (2011: S. 49)

S. 446 f.) Je nach gewähltem System sind noch weitere Vorkehrungen zu treffen. Beispielsweise beim System „FW 50+“ der Firma Schüco International KG müssen die Riegelprofile ausgeklinkt und an den Pfostenprofilen gegebenenfalls eine entsprechende Ausklinkung errichtet werden. Hintergrund des Ganzen ist, dass bereits erwähnte mehrstufige Dichtungssystem, das in die Tragprofile integriert ist. Das Ausklinken erfolgt dabei wie in Abbildung 14 dargestellt. Somit müssen einerseits die Riegel ausgeklinkt werden, sodass die Dichtungen in den Pfosten hineinragen, um einen dichten Fassadenknoten herstellen zu können. Andererseits werden auch die Nuten für die Dichtungen der Pfosten zwischen Ebene 2 und Ebene 3 ausgeklinkt, damit der Riegel in der richtigen Höhe den Pfosten schneidet.

2.4.1.6 Nicht-ebene Fassadenknoten

Die in Kapitel 2.4.1.5 beschriebenen Erkenntnisse sind nur auf plane Pfosten-Riegel-Fassaden anwendbar. Damit ist gemeint, dass alle Ausfachungen innerhalb einer Ebene liegen. Dadurch werden alle Profile orthogonal zueinander angeordnet. Folgend soll exemplarisch gezeigt werden, in welchem Rahmen das Fassadensystem „FW 50+“ der Firma Schüco International KG für nicht-ebene Fassaden geeignet ist. Hierbei ist zu beachten, dass auch andere Fassadensysteme prinzipiell für diese Anwendung geeignet sind. Die Ausarbeitung eines digitalen Fassadenknotens im Konzept „Parametric Design“ wird im Rahmen dieser Diplomarbeit auf ein System begrenzt. Dies ist darin begründet, dass die Umstellung auf andere Systeme eine vollständige Überarbeitung des Algorithmus benötigt und daher sehr zeitintensiv ist. Prinzipiell kann solch ein Algorithmus für alle bestehenden Systeme entwickelt werden. Dabei gilt zu beachten, dass alle Systeme einen eigenen Gültigkeitsbereich besitzen und somit nicht in jedem System alle Geometriestellung erfüllt werden können.

Zunächst soll aufgezeigt werden in welchen Bedingungen die Riegelprofile aufeinandertreffen können. Hierfür sind die vorhandenen Pfostenprofile maßgeblich. Einerseits stellt Schüco verschiedene Dichtungen zu Verfügung. In konkaver Richtung kann so um bis zu 10 ° je Seite von der planen Ebene abgewichen werden. In konvexer Richtung sogar bis zu 15 °. Andererseits sind auch variable Pfostenprofile vorhanden. Diese erlauben je Seite eine Abweichung von bis zu 25 ° in konvexer Richtung. (Schüco International KG 2011: S. 60 - 62, 64) Eine Verbindung dieser beiden Möglichkeiten ist nicht explizit möglich, wird aber zur Erweiterung des Wertebereichs im Rahmen dieser Diplomarbeit angenommen. Des Weiteren wird die Annahme getroffen, dass die variablen Pfostenprofile bereits gebogen vorhanden sind. Hierfür stehen die Winkel 0 °, 5 °, 10 °, 15 °, 20 ° und 25 ° zur Verfügung. Der Gültigkeitsbereich beträgt somit je Seite von - 10 ° bis + 40 ° in konvexer Richtung.

Andererseits ist auch eine Abweichung der Pfostenprofile von der planen Ebene möglich. Der Wertebereich wird in diesem Fall von den zur Verfügung stehenden Riegelprofilen begrenzt. Im Gegensatz zu den Pfosten sind für die Riegel keine abgewinkelten Dichtungen vorgesehen. Somit stehen nur variable Riegelprofile zur Verfügung. Diese weisen einen Gültigkeitsbereich von 38 ° bis 90 ° in konvexer Richtung auf. (Schüco International KG 2011: S. 76) Der Bereich zwischen 0 ° und 38 ° kann in der Realität durch das Fassadensystem nicht dargestellt werden. Im Rahmen dieser Arbeit wird dennoch die Annahme getroffen, dass die variablen Riegelprofile auch für diesen Wertebereich geeignet sind. Dies hat den Hintergrund, dass für den digitalen Fassadenknoten keine Unterbrechung des Gültigkeits-

bereiches vorliegt. Für die Entwicklung eines realen Fassadenknotens bedeutet dies, dass ein weiteres Profil, das den Bereich schließen kann, oder mehrere Profile in Kombination mit variablen Dichtungen entwickelt werden müssen.

Um Toleranzen im realen Fassadenknoten zu berücksichtigen, wird außerdem die Annahme getroffen, dass die Standarddichtungen und Standardriegelprofile auch für kleine Winkel geeignet sind. Für die Standarddichtungen wird ein Gültigkeitsbereich von - 1,25 ° bis 1,25 ° angenommen. Der Wertebreich der Standardriegelprofile wird auf - 2 ° bis 2 ° gewählt. Dies hat den Hintergrund, dass anderenfalls auch bei sehr kleinen Winkeln die abgewinkelten Dichtungen, beziehungsweise variablen Riegelprofile, verwendet werden, da im Programmablauf das exakte Treffen des Wertes 0,00 sehr unwahrscheinlich ist.

2.4.2 Elementfassade

Hestermann und Rongen (2018: S. 573) und auch Knaack et al. (2014: S. 46) grenzen die Elementfassade von der Pfosten-Riegel-Fassade durch einen gewissen Grad an werksseitiger Vorfertigung ab. Dadurch können die Montagekosten und -zeiten vor Ort sowie die Abhängigkeit vom Wetter deutlich reduziert werden. Zudem kann eine Elementfassade mit erhöhter Genauigkeit gegenüber der Pfosten-Riegel-Fassade kalkuliert werden.

Die Elementfassade zeichnet sich durch einen tragenden Rahmen aus. Dieser muss steif ausgeführt werden. Die Verglasungen beziehungsweise andere Ausfachungselemente und weitere Haustechnikkomponenten werden wenn möglich werksseitig montiert. Die Montage erfolgt über Anschlusswinkel, die im Bereich der Decke montiert werden. Es ist zu beachten, dass diese Befestigungen exakt in der Rohbauphase ausgerichtet werden müssen. Die Montage der einzelnen Elemente kann anschließend nur von unten nach oben erfolgen, da die oberen Module auf die unteren aufgesetzt werden. Die Elemente werden dabei hängend montiert, sodass am oberen Punkt ein Festlager und unten ein Loslager verbaut werden. Dieses Loslager wird als vertikal verschieblicher Bolzen realisiert, vergleichbar mit dem Bauwerksanschluss einer Pfosten-Riegel-Fassade in Abbildung 12 (zu finden auf Seite 22. (Knaack et al. 2014: S. 46, 60)

Nachteilig ist, dass im Gegensatz zu den Pfosten-Riegel-Fassaden die Randprofile der Elemente einer Element-Fassade doppelt ausgeführt werden müssen. Somit ergeben sich dickere Ränder, was sich negativ auf die filigrane und transparente Beschaffenheit sowie auf die daraus resultierende natürliche Belichtung auswirkt. In der Regel ist das Randprofil eines Elementes circa 4,00 cm stark. Da es doppelt verbaut werden muss, sind die Abstände zwischen den Glasscheiben circa 8,00 cm groß. Im Vergleich dazu sind die Profile einer Pfosten-Riegel-Fassade circa 5,00 - 6,00 cm dick. Die Größe der einzelnen Elemente wird

hauptsächlich durch den Transport begrenzt. In der Regel werden Elemente verbaut, die geschosshoch und 1,20 - 2,70 m breit sind. (Knaack et al. 2014: S. 46)

Das Dichtungssystem einer Elementfassade wird innerhalb der Felder werksseitig hergestellt, sodass die Elemente vor Ort untereinander abgedichtet werden müssen. Da keine dichtenden Pressleisten zwischen den Feldern vorhanden sind, werden die Dichtungen in vertikaler Richtung als Steckdichtungen ausgeführt. In horizontaler Richtung wird ein Dichtband verlegt, auf dem die einzelnen Fassadenelemente montiert werden. In der Regel sind in den einzelnen Elementen drei Nuten vorgesehen, in denen die Dichtungen positioniert werden. (Knaack et al. 2014: S. 46, 60)

3 Grundlagen des parametrischen Fassadenknotens

In diesem Kapitel werden Grundlagen erläutert, die zur Produktion eines parametrischen Fassadenknotens notwendig sind. Zunächst soll hierzu der Begriff der digitalen Fertigung definiert werden und in kurzen Fallbeispielen aufgezeigt werden, wie die digitale Fertigung bereits im Bauwesen eingesetzt wird. Anschließend wird eine kurze Einführung in die Themen Parametric Design und Datenstruktur gegeben. Darauf aufbauend soll eine Datenstruktur für einen parametrischen Fassadenknoten einer Pfosten-Riegel-Konstruktion aufgebaut werden. Hierfür wird ein bestehendes System, wie es in Kapitel 2.4.1 beschrieben wird, als Ausgangspunkt genutzt.

3.1 Digitale Produktion zur Errichtung von Fassaden

In diesem Kapitel soll erläutert werden, wie die digitale Produktion[1] zur Fertigung und Errichtung von Fassaden genutzt wird. Hierfür werden zunächst die Grundlagen der digitalen Produktion erläutert. Anschließend wird die aktuelle Umsetzung anhand von Fallstudien aufgezeigt.

3.1.1 Grundlagen der digitalen Produktion

Bereits in der vorangestellten Projektarbeit konnte der Begriff der digitalen Produktion definiert werden. Hierunter wird eine Produktionsweise verstanden, „in der Informationen durchweg in digitaler Form erzeugt, verwaltet, gespeichert, verteilt, verarbeitet und kommuniziert werden." (Westkämper et al. 2013: S. V) Somit beschreibt die digitale Produktion die Verwendung von digitalen Technologien und Methoden zur Herstellung von realen Produkten. Nach Hofmann (2017: S. 1) setzt sich die digitale Produktion das Ziel die konventionelle Fertigung in drei Gesichtspunkten zu optimieren:

1. Auftragsplanung: Einbeziehung von Echtzeitdaten aus der realen Produktion
2. Auftragsvorbereitung: Visualisierung und Simulation von Prozessen und Produkten zur Fehlervermeidung
3. Auftragsdurchführung: durch intelligent vernetzte Prozessketten

[1] Der Begriff der digitalen Produktion entspricht dem Begriff der digitalen Fertigung. Da im deutschen Sprachgebrauch die digitale Produktion verbreiteter ist, wird folgend von digitaler Produktion gesprochen.

Um die digitale Produktion umzusetzen, wird sowohl ein digitales Produktmodell als auch ein digitales Modell der fertigenden Fabrik benötigt. Dieses Modell umfasst alle Prozesse, Produkte und Ressourcen einer Fabrik. Dabei wird sowohl eine statische Darstellung der Fabrik und der Produktionsanlagen, als auch eine dynamische Darstellung der ablaufenden Prozesse verwendet. (Schuh, Klocke und Ripp (2002) nach Westkämper et al. (2013: S 110)) So liegen einerseits alle relevanten Informationen zum zu fertigenden Produkt vor, als auch alle Informationen über die ausführbaren Methoden innerhalb eines Werkes. Die Erzeugung der Produktmodelle wird durch Engineering-Umgebungen, wie etwa CAD- oder FEM-Systeme realisiert. Innerhalb dieser Umgebungen können zudem Berechnungen und Optimierungen stattfinden. Mithilfe von administrativen Managementsystemen kann der Arbeitsablauf um betriebswirtschaftliche und logistische Funktionen erweitert werden. Die Fertigung der realen Produkte wird mit computergestützten Systemen und mechatronischen Elementen (Sensoren und Aktoren) ausgeführt. Dabei sind die ausführenden Systeme über maschineninterne IT-Systeme miteinander vernetzt, sodass diese miteinander kommunizieren können. Durch diesen Arbeitsablauf sind in der digitalen Produktion sämtliche technischen, organisatorischen und administrativen Prozesse eines Unternehmens enthalten. (Westkämper et al. 2013: S. 11)

Im Rahmen der Projektarbeit konnte außerdem die Struktur der digitalen Produktion aufgezeigt werden. Wie in Abbildung 15 dargestellt ist, eignen sich vor allem additive und konventionelle CNC-gesteuerte Fertigungsverfahren zur digitalen Produktion. CNC-gesteuerte konventionelle Verfahren gliedern sich wiederum in mechanische, thermische und forma-

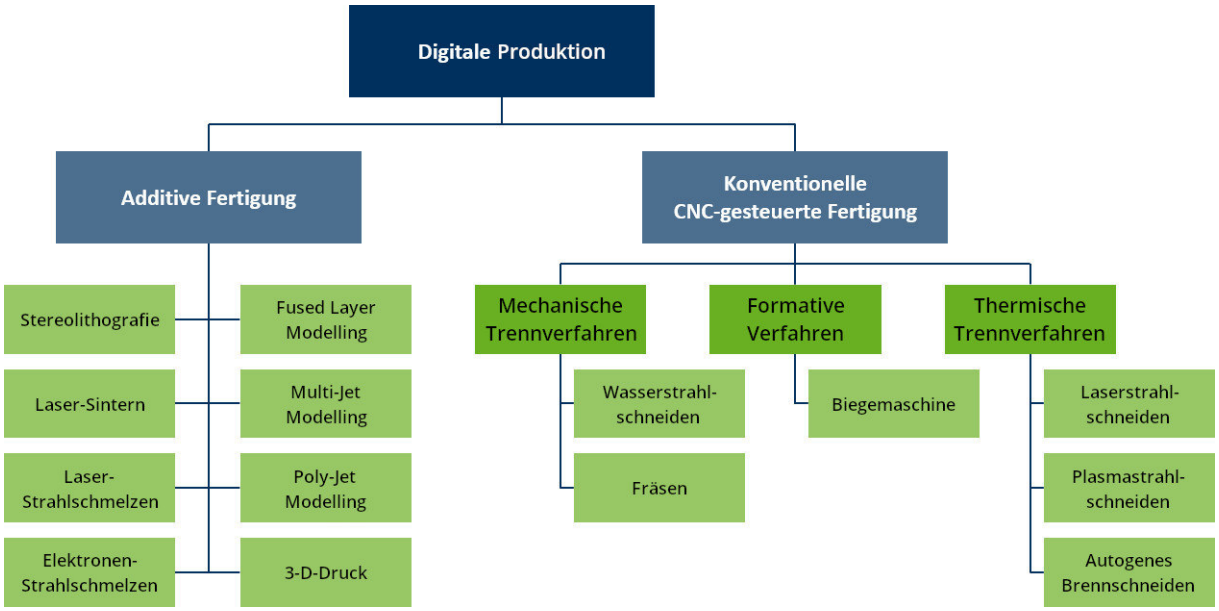


Abbildung 15: Fertigungsverfahren der digitalen Produktion
Quelle: Eigene Darstellung

tive Verfahren. Zudem ist eine kleine Auswahl an verfügbaren Verfahren gegeben. Alle diese Verfahren haben ihre eigenen Vor- und Nachteile. Beispielsweise eignet sich das autogene Brennschneiden nur für Stähle mit niedrigem Legierungsgrad. In Kapitel 3.2 „Additive Fertigung“ und 3.3 „CNC-Fertigung“ sind die Verfahren der Abbildung beschrieben. Daher soll in dieser Arbeit nicht weiter auf die Fertigungsverfahren eingegangen werden.

3.1.2 Schüco Parametric System

Elementfassaden sind für die Errichtung von parametrischen Fassaden geeignet. Auf dem Markt sind bereits erste Systeme erhältlich, wie beispielsweise das „Schüco Parametric System“ der Firma Schüco International KG. Das Konzept dieser Elementfassade ist in Abbildung 16 verdeutlicht. Jedes der pyramiedenartigen Felder stellt dabei ein Fassadenelement dar. Diese bestehen aus einem tragenden Elementrahmen. Auf den Rahmen wird eine dreidimensionale Rohrrahmenstruktur aufgebracht. Diese stellt die Verbindung zwischen den ausfachenden Elementen und dem Elementrahmen dar. (Schüco International KG 2016b: S. 17) Dadurch muss nur der Rohrrahmen an den Entwurf angepasst werden. So können wiederholend die gleichen Elementrahmen verwendet werden. Der große Vorteil des Rohrrahmens besteht darin, dass das Verschneiden der Rohre aufgrund der Geometrie eines Zylinders sehr einfach erfolgen kann. Die Fertigung der Fassadenelemente erfolgt in einem 5-Achsen-Bearbeitungszentrum wie etwa der Schüco Maschine DC 500. Zur Steuerung der Anlage werden die 3-D-Daten eines digitalen Detailmodells genutzt. (Schüco International KG 2016b: S. 22 f.) Dieses Beispiel zeigt auf, wie die digitale Produktion bereits im Bauwesen angewendet wird. Um weitere gestalterische Möglichkeiten zu ermöglichen, wird der Fokus der praktischen Ausarbeitung auf Pfosten-Riegel-Fassaden gesetzt. So können die Vorteile dieses Fassadensystems mit den Vorteilen des Parametric Designs kombiniert werden. Dabei gilt jedoch, dass die Erkenntnisse auch auf Elementfassaden übertragen werden können.

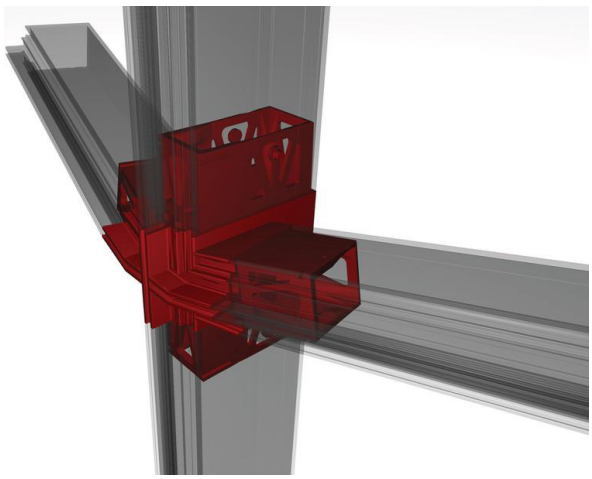
Abbildung 16: Konzept einer parametrischen Elementfassade
Quelle: Eigene Darstellung

3.1.3 Nematox Façade Node

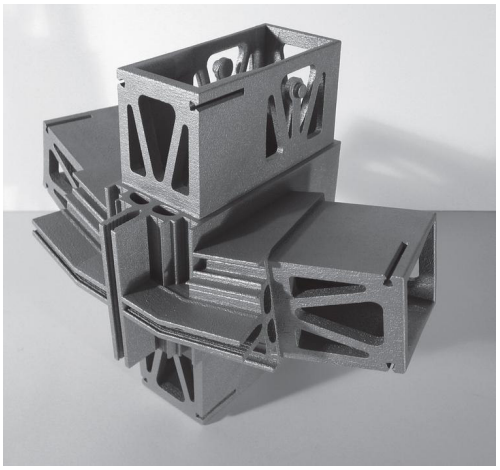
Bereits im Rahmen der Projektarbeit des Autors wurde das Beispiel des Nematox Façade Nodes herausgearbeitet. Hierbei handelt es sich um einen hochgradig anpassbaren Fassadenknoten für Pfosten-Riegel-Fassaden. Wie in Abbildung 17 zu sehen, können die einzelnen Pfosten und Riegel in bestimmten Winkeln zueinander montiert werden. Durch diese Winkel entstehen nicht nur plane Oberflächen, sondern dreidimensionale Fassaden, die beliebig geformt sein können.

Die Ausbildung von nicht-rechtwinkeligen Fassadenanschlüssen stellt in der Praxis immer ein Problem dar, da hierfür Schnitte durch Freiformwinkel geführt werden müssen. Da diese nur sehr ungenau ausgeführt werden können, ergeben sich daraus Leckagen in der Fassade. Dieses Problem wird baustellenseitig durch die Verwendung von Silikon gelöst. In den meisten Fällen ist solch eine Lösung jedoch unzureichend. (Strauß 2013: S. 114)

Daher wurde zwischen 2008 und 2010 der Nematox Façade Node entwickelt. Das Produktdesign erfolgt rein digital. Die Winkel können als Parameter in eine Software eingegeben werden, sodass diese automatisch das Knotenmodell auf diese Angaben anpasst. Da aufgrund von Weiterentwicklungen in der Direct Metal Fabrication – beispielsweise LBM (vgl. Kapitel 3.2) – die Produktionszeiten reduziert und die Bauteilqualität verbessert werden konnten und zeitgleich die Preise für additive Fertigungssysteme und deren Materialien gesunken sind, wurde die Fertigung der Knoten mittels additiver Fertigung realisiert. (Strauß 2013: S. 114 f. Strauß, Emmer Pfenninger Partner AG und Knaack 2015: S. 229 ff.)



(a) Rendering des Nematox Façade Nodes



(b) Prototyp aus Aluminium

Abbildung 17: Nematox Façade Node
Quelle: Strauß (2013: S. 116, 119)

3.2 Fertigung einer parametrischen Pfosten-Riegel-Fassade

Ein Ansatz, um den Knoten zu fertigen, ist die additive Fertigung (engl. additive Manufacturing) (AM). Darunter wird „[die] direkte Herstellung von Bauteilen aus elektronischen Daten ohne formgebende Werkzeuge“ (Schmid 2015: S. 3) verstanden. Ein Merkmal der AM ist, dass die Formgebung eines Werkstückes durch das Hinzufügen von Rohmaterial erzielt wird. (Strauß, Emmer Pfenninger Partner AG und Knaack 2015: S. 226) Für weitere Erläuterungen zur Definition des Begriffs der additiven Feritung ist an dieser Stelle auf Kapitel 3.2.1 „Begriffsdefinition: Additive Fertigung“ in der Projektarbeit des Autors verwiesen.

Additive Fertigungsverfahren bringen einige Vorteile mit sich. So sind etwa ihre Anwendbarkeit auf stets unterschiedliche Bauteilgeometrien und die Herstellung von Geometrien, die mit konventionellen Fertigungsverfahren nicht herstellbar sind, zu nennen. Jedoch entstehen dadurch einige Nachteile. Die Herstellungszeiten für einen einzelnen Fassadenknoten sind höher und es können derzeit nicht dieselben statischen Eigenschaften erzielt werden, wie bei der Verwendung von konventionellen CNC-Verfahren. Bezüglich der Herstellungszeit schreibt Strauß (2013: S. 119), dass die Produktionszeit eines Prototypen des additiv gefertigten Nematox Façade Node 80,5 Stunden beträgt. Davon sind 76,5 Stunden der eigentliche Druckvorgang und der Rest Post-Processing. Der Prototyp wurde dabei mit LaserCusing aus Aluminium gefertigt. LaserCusing beschreibt das gleiche Verfahren wie der Begriff Laser-Strahlschmelzen (engl. Laser Beam Melting) (LBM). Hierbei handelt es sich um ein additives Feritungsverfahren. Bei diesem Verfahren werden pulverförmige Metalle lokal durch einen Laser aufgeschmolzen. Beim Abkühlen des Materials erhärtet dieses und bildet so das Bauteil. Weitere Details sind in der Projektarbeit des Autors in Kapitel 3.2.3.3 Laser-Strahlschmelzen beschrieben.

Um die Produktionszeiten und die strukturellen Eigenschaften zu verbessern, wird der Fokus dieser Arbeit darauf gerichtet, ein bestehendes System zu verwenden und so zu modifizieren, dass es zur Errichtung von komplexen Fassadenschnittpunkten unter Nutzung von digitalen Fertigungsprozessen geeignet ist. Als bestehendes System kommen alle, auf dem Markt verfügbaren Fassadensysteme in Betracht. Die herstellerseitig standardisierten Pfosten- und Riegelprofile sowie die dazugehörige Peripherie, wie etwa spezielle Dichtungen, Pressleisten, Blendleisten, etc., können genutzt und mit einer 3-D-Profilschneideanlage verschnitten werden. Mithilfe von zwei Roboterarmen sollen die Bauteilkomponenten abschließend in Position gebracht und miteinander verbunden werden. Um einen kraftschlüssigen Verbund der Komponenten zu erreichen, können diese beispielsweise miteinander verschweißt werden.

Die praktische Ausarbeitung in Kapitel 4 bis 7 dieser Arbeit, beschäftigt sich hauptsächlich mit dem Fassadensystem "FW 50+"der Firma Schüco International KG. Die gewonnenen Erkenntnisse können auf Fassadensysteme anderer Hersteller übertragen werden. Da die Umstellung auf ein anderes System sehr zeitintensiv ist, wird vom Autor empfohlen, sich bei einer weiteren Ausarbeitung dieser Arbeit auf ein Fassadensystem oder zumindest einen Hersteller zu konzentrieren oder ein eigenes System zu entwickeln.

Zum Verschneiden der Profile kann eine Schneideanlage vergleichbar zur Produktionsanlage „Schüco Maschine DC 500“ der Firma Schüco International KG verwendet werden. Diese Anlage wird von der Firma Schüco International KG zur Fertigung ihres Parametric Systems genutzt. Die Maschinensteuerung basiert auf Grundlage von dreidimensional (3-D)-Daten des Detailmodells des zu fertigenden Fassadenelements. Mithilfe einer CNC-gesteuerten Scheiben- und Vollkernfräsanlage wird der Rohrprofilrahmen der Elementfassaden zur Montage vorbereitet. (Schüco International KG 2016b: S. 22; Schüco International KG 2016a: S. 12 ff.)

Als alternativer Ansatz kann eine Schneideanlage vergleichbar zur Anlage „MPC 450 I 500 – 1200“ der niederländischen Firma HGG Group genutzt werden. Diese Anlage ist für die 3-D-Profilierung von runden und quadratischen Querschnitten geeignet. Für die Anlage stehen wahlweise eine Plasma- oder eine Sauerstoffbrennschneidetechnologie zur Verfügung. (HGG Group 2019a; HGG Group 2019b) Wie bereits in der vorangestellten Projektarbeit gezeigt wurde, eignet sich das Sauerstoffbrennschneiden nicht für Aluminium, sodass die Plasmaschneidetechnologie verwendet werden sollte. Eine Eignung der Anlage für Fassadenprofile ist zu prüfen. Prinzipiell kann eine ähnlich aufgebaute Schneideanlage, die auf die Abmessungen der Profile optimiert wird, genutzt werden.

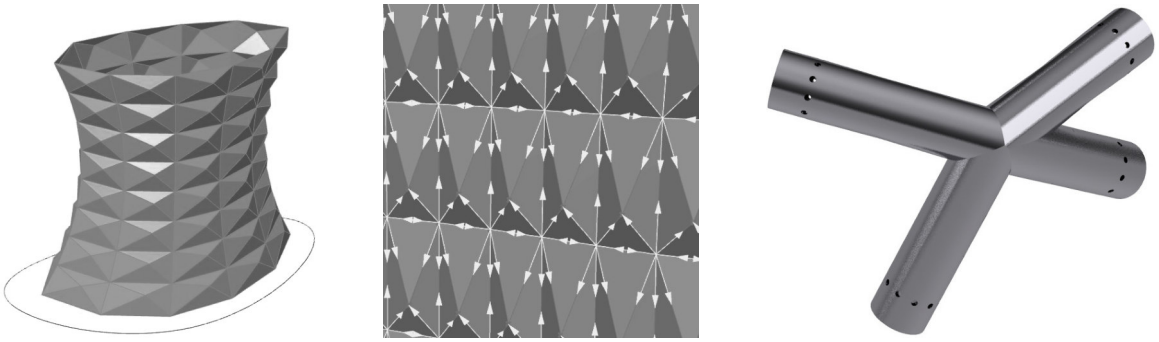
3.3 Parametric Design

Wie bereits in einer vorangestellten Projektarbeit des Autors gezeigt wurde, versteht man unter dem Begriff „Parametric Design“ ein Architekturkonzept, das auf Regeln, Einschränkungen, Features und Assoziationen zwischen Parametern und Objekten innerhalb eines Modells basiert. Das Ziel des Konzepts ist eine nutzerfreundliche Anpassung des Modells durch Eingabeparameter. Die Regeln und Einschränkungen können durch mathematische Formeln beschrieben werden. So beeinflussen die Variablen der Formeln die Eigenschaften eines Modells maßgebend. (Aksamija et al. 2011: S. 33)

Ein großer Vorteil des Parametric Designs ist die Verarbeitbarkeit durch einen Computer. So können die Eingabeparameter nicht nur nach ästhetischen und gestalterischen Grün-

den gewählt, sondern auch mit einer bestimmten Bedeutung verknüpft werden. Dadurch konnte in der Projektarbeit aufgezeigt werden, dass sich Parametric Design für die Errichtung von nachhaltigen Gebäuden nutzen lässt. So können zum Beispiel, wie eine Arbeit von Aksamija et al. (2011: S. 35) beschreibt, die Verschattungselemente eines Gebäudes variabel verschoben werden. Die Positionierung soll auf Grundlage von Gebäudesimulationsberechnungen erfolgen. So kann ein Optimum zwischen winterlichen Wärmegewinnen aufgrund von Solarstrahlung durch transparente Bauteile und dem sommerlichem Energieverbrauch, bedingt durch Klimaanlagen, erzielt werden.

In der Projektarbeit wurde des Weiteren aufgezeigt, wie Parametric Design dazu eingesetzt werden kann, ein gesamtes Gebäude zu modellieren, für die Fassade relevante Daten zu extrahieren und diese zur Konstruktion von parametrischen Fassadenknoten zu verwenden. Bei dieser Betrachtung wurde auf eine vollständige Ausarbeitung aller konstruktiven Details verzichtet. Die Abbildung 18 (a) bis (c) verdeutlichen die Erkenntnisse aus der Projektarbeit. In Abbildung 18 (a) ist das gesamte Entwurfsmodell dargestellt. Dieses wurde auf Grundlage eines vereinfachten Grundrisses, Parametern zur Beschreibung der Fassadengeometrie und verschiedenen Eingangsparametern, wie Geschosshöhe oder Anzahl der Geschosse, erzeugt. Das Grundkonzept der Fassade ist vergleichbar mit Abbildung 16 und besteht aus einer pyramidenartigen Geometrie, die sich aus fünf Punkten zusammensetzt. Hierzu bilden vier Punkte den äußeren Rahmen eines einzelnen Fassadenelementes und der fünfte Punkt befindet sich im Feld. Durch Variablen kann der Punkt im Feld und in Normalenrichtung zur Feldebene verschoben werden. Somit können über die Eingabeparameter beliebig viele verschieden geformte Gebäudegeometrien realisiert werden. Abbildung 18 (b) verdeutlicht den Informationsfluss zwischen dem Entwurfsmodell des Gebäudes und dem Konstruktionsmodell des Fassadenknotens. Im Falle des in der Projektarbeit erstell-



(a) Entwurfsmodell (b) Datenfluss (c) Parametrischer Fassadenknoten

Abbildung 18: Erkenntnisse aus der Projektarbeit
Quelle: Screenshots aus Rhinoceros 6

ten Gebäudemodells sind die wichtigsten Informationen die Vektoren zwischen den zuvor beschriebenen fünf Punkten. Abbildung 18 (c) bildet einen ersten Entwurf für das Design des parametrischen Fassadenknotens ab. Bei diesem Designmodell werden dabei keine konstruktiven Details beachtet, wie etwa die Befestigung der Fassadenelemente an der Tragstruktur. Da der Übergang vom Entwurfsmodell des Gesamtbauwerks zum Konstruktionsmodell eines einzelnen Knotens bereits ausreichend in der vorangestellten Projektarbeit beschrieben wurde, grenzt sich diese Diplomarbeit insofern ab, dass keinerlei Bezug zum gesamten Gebäudemodell hergestellt wird. Dies bedeutet, dass der Fokus auf der Ausarbeitung eines parametrischen Fassadenknotens liegt und grundlegende konstruktive Details, wie etwa die Befestigung von Glaselementen an den Profilen, berücksichtigt werden. Daher und um die Zeit zur Berechnung eines Fassadenknotens auf das Notwendigste zu reduzieren, werden keine Informationen aus einem Entwurfsmodell extrahiert und weiterverarbeitet. Die Eingabe dieser Parameter erfolgt in diesem parametrischen Design durch den Nutzer in Form von Rotationswinkeln für jedes einzelne Bauteil.

Wie bereits beschrieben, wird in dieser Diplomarbeit ein parametrisches Design zur automatischen Erzeugung eines Fassadenknotens ausgearbeitet. Zur besseren Verständlichkeit der Vorgänge innerhalb des Algorithmus werden zunächst in Kapitel 3.4 einige Kenntnisse über Rhinoceros 6 und die seit Version 6 darin enthaltene Erweiterung Grasshopper erläutert. Rhinoceros (auch kurz: Rhino) 6 ist eine kommerziell erhältliche 3-D-Computergrafik- und CAD-Anwendungssoftware der Firma Robert McNeel & Associates. Bei dem Plug-In Grasshopper, das hauptsächlich von David Rutten und Robert McNeel & Associates entwickelt wurde (Mode Lab 2015: S. 4), handelt es sich um eine grafische Programmierumgebung mit der sich unter anderem parametrische Entwürfe erstellen lassen.

3.4 Grundlagen zu Rhinoceros 6 und Grasshopper

In diesem Kapitel soll ein Einblick in Rhinoceros 6 und Grasshopper geben werden. Dies dient zur besseren Übersicht ab Kapitel 4 und gilt anschließend als grundlegend.

Grasshopper ist eine visuelle Programmierumgebung (Mode Lab 2015: S. 6). Schiffer 2019 definiert eine visuelle Programmiersprache als „[f]ormale Sprache mit visueller Syntax oder visueller Semantik zur vollständigen Beschreibung der Eigenschaft von Software“. Somit benötigt es zu einer visuellen Programmierung grafische Elemente, die die Syntax oder die Semantik beschreiben, anstelle des sonst textuellen Quellcodes. In Grasshopper übernehmen dies sogenannte Komponenten, die auf der Arbeitsfläche, dem sogenannten Canvas, platziert werden und dort mithilfe von Kabeln zu einem Algorithmus zusammen gefügt

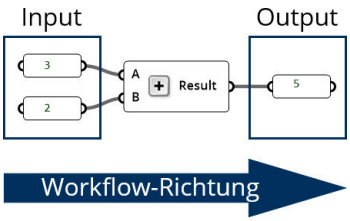


Abbildung 19: Funktion von Komponenten in Grasshopper
Quelle: In Anlehnung an Grasshopper

werden können (Mode Lab 2015: S. 19 f., 40 ff., 57 ff.). Diese Komponenten können als vorgefertigte Programmbausteine verstanden werden, denen, wie in Abbildung 19 dargestellt, meist von der linken Seite kommend Inputwerte gegeben werden. Der Baustein führt die ihm eingespeicherte Operation durch. Im Fall der Abbildung 19 handelt es sich um eine simple Addition. Auf der rechten Seite können die Ergebnisse der Komponente entnommen werden. Der allgemeine Programmfluss findet hierbei von links nach rechts statt (Mode Lab 2015: S. 8, 43). So können die verschiedenen Komponenten zu einem vollständigen Programm zusammengefügt werden.

3.4.1 Wichtige Komponenten in Grasshopper

An dieser Stelle ist auf einen kurzen Überblick über die wichtigsten, vorprogrammierten und in der praktischen Ausarbeitung verwendeten Komponenten verwiesen. Dieser zeigt die Komponenten mit ihrem Namen, ihrer Darstellung in Grasshopper und ihrer grundsätzlichen Funktion. Als Quelle hierfür dienen die Dokumentationen von Mode Lab 2015 und Rodricks und Heumann 2018a. Weitere Informationen zu den aufgelisteten und weiteren Komponenten sind den Dokumentationen zu entnehmen. In Anhang „A1: Tab. 5: Übersicht: wichtige Grasshopperkomponenten“ ist der Überblick gegeben. Dabei gilt zu beachten, dass die In- und Outputs der Komponenten umbenannt werden können und dass es sich nur um eine kleine Auswahl aller Komponenten handelt.

3.4.2 Datenstrukturen

In der Informatik wurden viele verschiedene Datenstrukturen entwickelt. Diese Datenstrukturen dienen dazu Informationen zu speichern. Daten können auf unterschiedlichste Weise in den verschiedenen Strukturen abgelegt werden. Ein Feld, oder auch Array, stellt hierfür eine sehr einfache Möglichkeit dar. In einem Feld der Größe n können n Variablen $A[1], ..., A[n]$ vom gleichen Typ gespeichert werden. Über den Index $[i]$ ist ein direkter Zugriff auf jedes einzelne Element $A[i]$ möglich. (Blum 2013: S 4) Bei verketteten Listen, enthalten alle Knoten den Folgeeintrag. Somit zeigt jeder Knoten in einer einfach verketteten Liste auf den Nachfolger. Bei doppelt verketteten Listen ist zudem der Vorgängerknoten bekannt.

Der Vorteil bei dieser Datenstruktur besteht darin, dass beim Einfügen von Daten in die Mitte der Datenstruktur nur wenige Informationen modifiziert werden müssen. (Dietzfelbinger, Mehlhorn und Sanders 2014: S. 76) Um Elementfolgen, die nur am Anfang oder Ende modifiziert werden dürfen, abzubilden, wurden Schlangen und Keller entwickelt. Bei einem Keller werden die Daten am Ende eingefügt und entnommen. Dieses Prinzip nennt sich „last in first out (LIFO)". Im Kontrast dazu steht das Prinzip „first in first out (FIFO)", das von Schlangen umgesetzt wird. Hierbei werden die Elemente am Ende eingefügt und am Anfang der Folge entnommen. (Blum 2013: S. 5) Dies sind nur einige wenige Beispiele für die Speicherung von Daten. In der Informatik existiert eine Vielzahl weiterer Möglichkeiten, um Informationen abzuspeichern.

3.4.2.1 Datenstrukturen in Grasshopper

Das Entwicklerteam von Grasshopper hat sich bei den Datenstrukturen für die Implementierung einer einzigen Datenstruktur entschieden, um den Benutzer nicht mit verschiedenen Datenstrukturen zu verwirren. Um die Wahl des Konzepts nachvollziehen zu können, hilft ein Blick auf die Funktionsweisen beziehungsweise Kardinalitäten der Komponenten. Viele Komponenten funktionieren nach dem Prinzip „ein Input erzeugt genau einen Output". Die Kardinalität kann in diesem Fall als 1:1 ausgedrückt werden. Beispiel hierfür ist die Addition. Der Input von zwei Zahlen erzeugt genau ein Ergebnis. Häufig ist bei den Grasshopper-Komponenten die Kardinalität 1:N vertreten. Hierbei wird durch einen Input eine Vielzahl von Ergebnissen erzeugt. Beispiel hierfür ist die „Divide Curve"-Komponente. Diese zerlegt eine Input-Kurve in eine unterschiedlich lange Liste von Output-Punkten. Das gegenteilige N:1-Verhalten ist zum Beispiel bei der „Polyline"-Komponente gegeben, bei der eine Liste mit Input-Punkten genau eine Output-Kurve erzeugt. Des Weiteren gibt es noch N:N-Beziehungen. Hier erzeugen Input-Listen Output-Listen. Die meisten Komponenten aus dieser Kategorie gehören zum Bereich des „Datamanagements". Dies bedeutet, dass vielmehr die Art und Weise der Datenspeicherung beeinflusst wird, als die Informationen der Daten. Für diese Beziehung ist die „Reverse List"-Komponente zu nennen. Sie erfüllt den Zweck, dass die Einträge der Listen umgedreht werden, sodass der letzte Eintrag an erster Stelle steht und umgekehrt. Zuletzt gibt es die komplexeren Beziehungen 1:N', N':1 oder N':N'. Es werden ganze Datenbäume als Input benötigt, beziehungsweise als Output erzeugt. Beispielsweise die „Divide Surface"-Komponente erzeugt mehrere Listen mit Output-Punkten. Jede Liste repräsentiert eine Reihe an Punkten. (Rutton 2015)

Neben den Kardinalitäten der Komponenten ist außerdem die „Historie" der Daten maßgeblich für die Entscheidung für Datenbäume als primäre Datenstruktur in Grasshopper

verantwortlich (Rutton 2015). Wie Kapitel 3.4.2.2 zeigen wird, können den Daten so Informationen zugeordnet werden, die gar nicht in den Daten selbst gespeichert werden. Vielmehr ist es die Struktur selbst, die den Daten weitere Bedeutungen zuschreiben kann.

Wie Abbildung 20 zeigt, verfügt Grasshopper über verschiedene Darstellungen für Verbindungen. Um diese angezeigt zu bekommen, muss in Grasshopper unter „Display " die Option „Show Fancy Wires" aktiviert sein. Die Unterscheidung der verschiedenen Kabel erfolgt darin, dass unterschiedliche Datenstrukturen an die nächste Komponente übergeben werden. Im Bereich ① handelt es sich um einzelne Werte. Hier wird die Linie einfarbig dargestellt. Sollte es sich dabei um die leere Menge handeln, so wird das Kabel orange eingefärbt. Im Gegensatz dazu ist der Verbinder im Bereich ② durch eine weiße Linie geteilt. Bei der hier übergebenen Datenstruktur handelt es sich, wie auch in den beiden Panels erkennbar, um eine Liste. Um diese in einen vollständigen Baum umzuwandeln, wird der Pfad {0} der unteren Liste in den Pfad {1} umgewandelt, sodass beim Zusammenfügen der Listen mehrere Pfade vorhanden sind. Wie in Bereich ③ erkennbar ist, wird eine gestrichelte Linie für Datenbäume verwendet. (Mode Lab 2015: S. 57 ff.)

Ein Datenbaum ist nichts anderes als eine „geordnete Ansammlung von Listen" (Rutton 2015). Unter Ansammlung wird eine Liste verstanden, die weitere Listen enthält. So entsteht eine hierarchische Struktur der Listen. Jeder dieser Listen wird eine Nummer zugeordnet, die sie in Verbindung mit der Nummer aller ihr vorangegangen Listen eindeutig identifizieren lässt. Die Nummern müssen hierfür nicht durchgängig sein, sodass in einer Ebene mit zwei Listen die Liste 0 und Liste 5 existieren können. Die Datensätze bekommen einen innerhalb der Liste eindeutigen Schlüssel zugeordnet, wobei dieser bei [0] beginnt und immer um den Wert 1 erhöht wird. Dabei dürfen keine Werte ausgelassen werden, höchstens können Listenplätzen Null-Objekte zugeordnet werden.

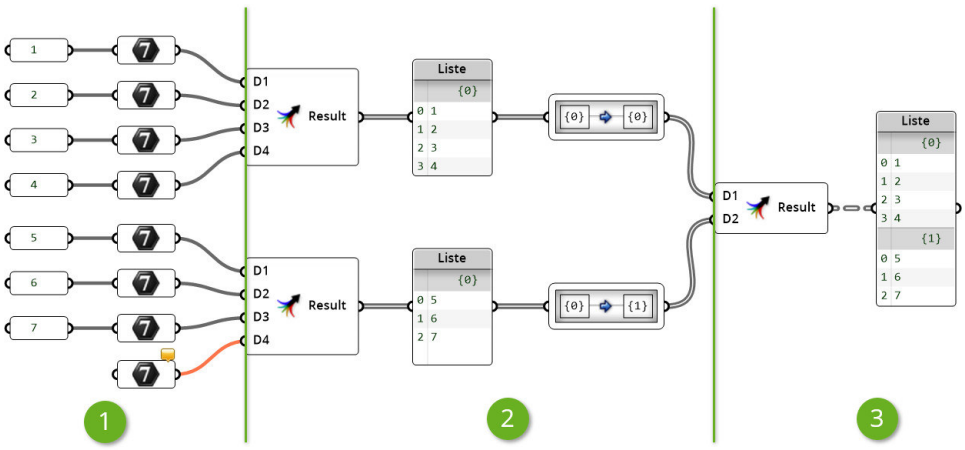


Abbildung 20: Darstellung von verschiedenen Datenstrukturen in Grasshopper
Quelle: In Anlehnung an Grasshopper

Die Nomenklatur von Pfaden sieht in Grasshopper vor, dass der Pfad einer Liste in geschweiften Klammer geschrieben wird und die einzelnen Hierarchieebenen durch Semikola getrennt werden. Die Objektschlüssel innerhalb einer Liste werden in eckigen Klammern geschrieben. Alle Objekte eines Baumes können über die Zusammensetzung von Pfad und Objektschlüssel eindeutig identifiziert werden. Folgend sind einige Beispiele für Objektpfade gegeben: (Rutton 2015)

{0;1;2} ist die dritte Liste aus der zweiten Liste der ersten Liste
{0;2;0;1}[4] ist das fünfte Element aus der Liste {0;2;0;1}

Bei der Nomenklatur für Pfade gilt:

{Hierarchieebene 0;Hierarchieebene 1;...;Hierarchieebene n}.

3.4.2.2 Datenstruktur des parametrischen Fassadenknotens

Die Erkenntnisse aus Kapitel 2.4.1 und Kapitel 3.4.2.1 können nun angewendet werden, um eine sinnvolle, bauteilorientierte Datenstruktur für einen parametrischen Fassadenknoten zu bilden. Bauteilorientiert soll bedeuten, dass der Fokus darin besteht, jederzeit die einzelnen Bauteile und Bauteilkomponenten aufgrund der Struktur, in der sie gespeichert werden, erkennen zu können.

Die erste Gliederungsebene der Datenstruktur wird durch die einzelnen Bauteile gegeben. Im Falle der praktischen Ausarbeitung handelt es sich hierbei um zwei Pfosten und zwei Riegel. Die ersten beiden Positionen werden für die beiden Pfosten belegt, wobei {0} dem nach oben und {1} dem nach unten zeigenden Profil entspricht. Die anderen beiden Zweige werden für die Riegel verwendet. Hier gilt, dass {2} nach rechts zeigt und {3} nach links. Eine Darstellung dieser Situation ist in Abbildung 21 gegeben. Diese Abbildung vereinfacht die Profile in Form von Vektoren.

Die zweite Datenstrukturebene wird durch die in Kapitel 2.4.1 beschrieben Komponenten einer Pfosten-Riegel-Konstruktion definiert. Diese Konstruktion besteht aus mehreren

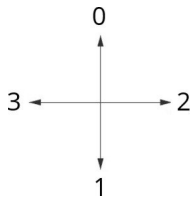
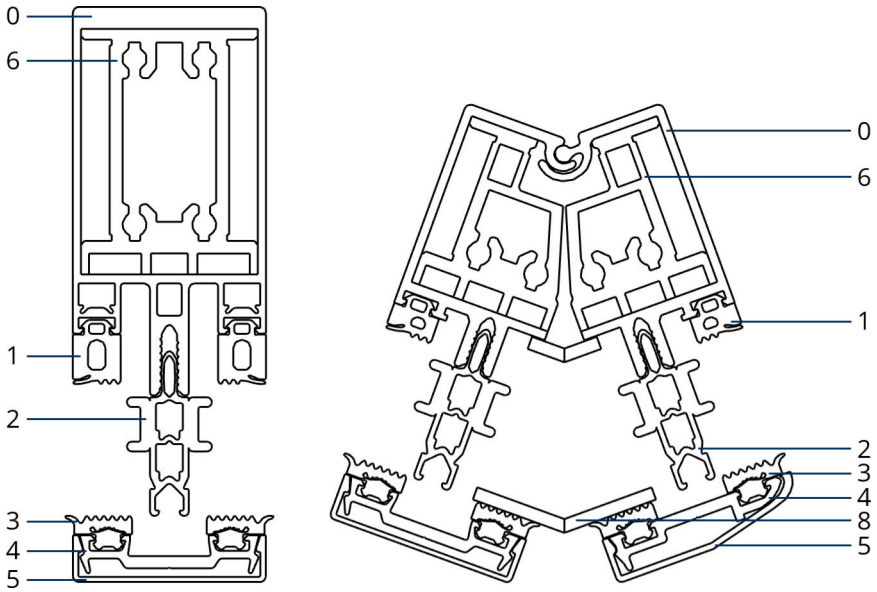


Abbildung 21: Datenstruktur des parametrischen Fassadenknotens
Quelle: Eigene Darstellung

Komponenten, die jeweils eine Funktion übernehmen. Wie bereits in Kapitel 2.4.1.3 beschrieben, besteht das Fassadensystem „FW 50+“ grundlegend aus sechs Komponenten. Abbildung 22 verdeutlicht den Aufbau des Tragprofils (0), der inneren Abdichtungen (1), des Isolators (2), der äußeren Abdichtungen (3), der Pressleiste (4), der Blendleiste (5), des Einschubprofils (6) eines Standardpfostenprofils und eines variablen Riegelprofils. Für das Riegelprofil ist zudem die Mittenverblendung (8) dargestellt. Die hierbei verwendete Nummerierung der Komponenten wird für den späteren Verlauf der praktischen Arbeit konsequent über den gesamten Arbeitsablauf behalten.

Der siebte Punkt wird bewusst „übersprungen“. Dies hat den Hintergrund, dass diese Position für die sogenannten Cutter reserviert wird, da jedes Bauteil mindestens einen Cutter erhält, jedoch die Mittenblenden für variable Riegelprofile nur bei den Riegeln – Bauteilkomponenten {2;7} und {3;7} – vorkommen können. Bei bestimmten Strukturveränderungen, beispielsweise dem in Kapitel 5.2.4 beschriebenen Gruppieren und dem anschließenden Aufheben der Gruppierung, kann anderenfalls die Positionen verrutschen. Dies wird durch den Gruppiervorgang hervorgerufen. Dabei werden alle Elemente eines Pfades in eine Gruppe gespeichert. Anschließend kann mithilfe der „Shift Path“-Komponente eine Hierarchieebene entfernt werden. Da bei Pfaden eine durchgehende Nummerierung der Pfade nicht zwingend erforderlich ist, bei Listen hingegen schon, werden Elemente aus einem Pfad hinter einer solchen Lücke nach oben verschoben und erhalten eine neue Nummerie-



0	Tragprofil	3	äußere Dichtung	6	Einschubprofil
1	innere Dichtung	4	Pressleiste	8	Mittenverblendung
2	Isolator	5	Blendleiste		

Abbildung 22: Komponenten einer Pfosten-Riegel-Fassade in der Datenstruktur
Quelle: In Anlehnung an Rhinoceros 6

rung. Unter Cutter werden in dieser Arbeit sämtliche Geometrien betrachtet, die zum späteren Verschneiden der vier Profile dienen. Diese Geometrien sind grob an die Geometrien einzelner Komponenten angelehnt. Jedoch finden sich zum einen starke Vereinfachungen, sodass die boolschen Operationen vom Computer leichter ausführbar sind. Zum anderen sind sie meist in bestimmte Richtungen größer als ihre verwandte Komponente, um beim Verschneiden Geometrien zu löschen, die möglicherweise über die verwandte Komponente herausragen. Die Cutter stellen dabei keinen positiven Anteil am Produkt dar, sondern sorgen ausschließlich für eine geometrische Subtraktion.

In der dritten Ebene werden Bauteilkomponenten unterschieden, die doppelt innerhalb eines Profils beziehungsweise einer Profilhälfte vorkommen können. Dies betrifft vor allem die Dichtungen. Aufgrund besonderer Geometriestellung des Fassadenknotens können die Dichtungen auf der linken und auf der rechten Seite unterschiedlich sein. Daher dürfen beide Geometrien nicht in einer gemeinsamen Liste gespeichert werden.

Die vierte Ebene nimmt vor allem auf die in Kapitel 2.4.1.6 beschriebenen variablen Riegelprofile Bezug. Hier werden beide Bauteilhälften in jeweils einen eigenen Pfad gespeichert. Dies ermöglicht es, alle Komponenten einer Bauteilhälfte auf einmal anzusprechen und spielt vor allem, wie später zu sehen, beim Aufdrehen der Riegelprofile eine Rolle.

Die gesamte Datenstruktur baut sich somit wie folgt zusammen:

{X;Y;Z;A}[i]

Mit:
- X - Bauteil (0,1 – Pfosten; 2,3 – Riegel)
- Y - Bauteilkomponente (siehe Auflistung oben)
- Z - Bauteilvarianten (v. a. bei Dichtungen (entweder 0 oder 1))
- A - Bauteilhälften (v. a. bei variablen Riegelprofilen (entweder 0 oder 1))
- i - Einzelne Geometrien (in diesem Fall je eine Fläche)

Zwar könnten die Bauteilkomponenten in einer Liste gespeichert werden, jedoch ist es bei Querschnitten mit abweichender Komponentenanzahl – beispielsweise variable Profile – problematisch, die Komponenten eindeutig auseinanderzuhalten.

Eine Darstellung der vollständigen Baustruktur ist in Abbildung 23 gegeben. In der Grafik ist zu erkennen, wie auf allen beschriebenen Hierarchieebenen Äste entstehen. Dabei ist dargestellt, dass vor allem die erste Hierarchieebene, die Ebene der Bauteile, für eine klare Aufteilung des Baumes sorgt.

Zudem erlaubt diese Datenstruktur mit den in Kapitel 3.4.2.3 beschriebenen Datenmanagementoperationen einen sehr komfortablen Zugriff auf einzelne Bauteilelemente.

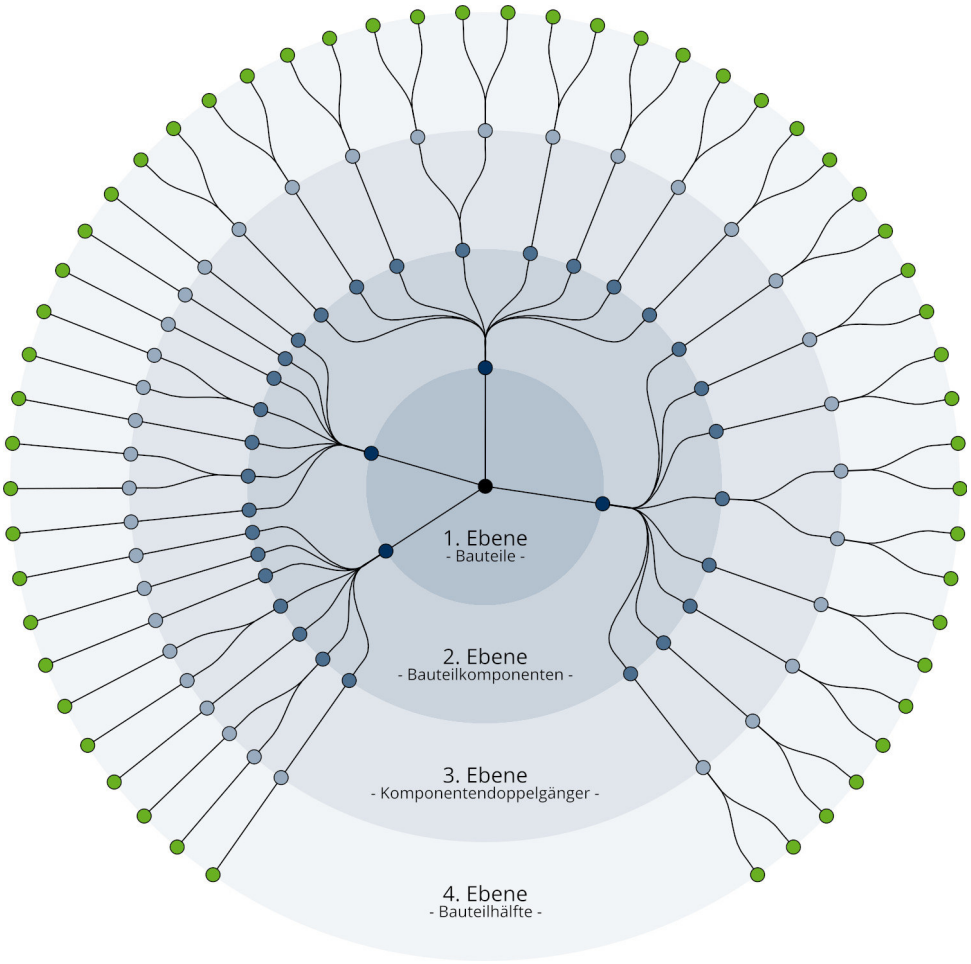


Abbildung 23: Datenstruktur des parametrischen Fassadenknotens
Quelle: Eigene Darstellung

3.4.2.3 Datenmanagement in Datenbäumen

Wie bereits in Kapitel 3.4.2.2 angedeutet, können den Daten innerhalb eines Datenbaumes übergeordnete Informationen zugeordnet werden, die nicht explizit in den Daten gespeichert werden. In Bezug auf die praktische Ausarbeitung lässt sich sagen, dass im Pfad {0;3;0;0} eine äußere Abdichtung des nach oben positionierten Pfostenprofils gespeichert ist. Dieses Wissen kann nur über die zuvor definierte Datenstruktur erzeugt werden, ist aber von essenzieller Bedeutung bei der weiteren Verarbeitung der Daten. Im Folgenden soll aufgezeigt werden, mit welchen Hilfsmitteln Bäume in Grasshopper manipuliert, bearbeitet und gefiltert werden können.

Zur Umstrukturierung der Pfade empfiehlt sich die Nutzung des „Path Mappers“. „Die ‚Path Mapper‘ Komponente (Sets/Tree/Path Mapper) erlaubt es lexikale Operationen auf Datenbäume anzuwenden.“ (Mode Lab 2015: S. 132) Unter lexikalen Operationen werden textbasierte Masken und Muster verstanden, die ein logisches Abbildungsverfahren zwischen einer In- und einer Output-Syntax ermöglichen. (Mode Lab 2015: S. 132)

Dieses Tool erlaubt eine sehr umfassende Syntax, sodass mit dem „Path Mapper“ fast alle Aufgaben des Datenbaummanagements durchgeführt werden können. Die Syntax besteht aus einer Input-Definition und einer Output-Definition. Bei geeigneter Syntax überträgt der „Path Mapper“ die Änderung auf den Datenbaum. (Rodricks und Heumann 2018b; TU Delft 2013) Die Input-Definition ist immer von dem Datenbaum abhängig, der modifiziert werden soll. Die Syntax benötigt genauso viele Pfadvariablen, wie die eingehende Datenstruktur aufweist. Die Pfadvariablen werden in geschweiften Klammern geschrieben. Sollen auch die Objektindizes verändert werden, so ist nach der geschweiften Klammer eine zusätzliche Variable in runden Klammern hinzuzufügen. Eine gültige Input-Syntax wäre beispielhaft:

{A;B;C}(i)

Die Output-Definition ist nach gewünschter Funktion zu wählen. Es sollten die Variablen aus der Input-Syntax verwendet werden. Es müssen dabei nicht alle Variablen vorkommen und es dürfen Ergänzungen mithilfe arabischer Zahlen gemacht werden. In Tab. 1 sind einige Beispiele für Output-Syntaxen und ihre Funktionen gegeben. Dabei wird die oben gezeigte Input-Syntax als Grundlage genutzt.

Eine der wichtigsten Komponenten zum Filtern der Datenstrukturen ist die „Split Tree“-Komponente. Diese nutzt eine Maske, um einen Baum in zwei Teile zu zerteilen. Als „Positive“ wird der Anteil der Daten ausgegeben, dessen Pfade der Maske entsprechen. Der Rest der Datenstruktur befindet sich in Output „Negative“. Um beispielsweise das Tragprofil der rechten Bauteilhälfte des nach links zeigenden variablen Riegels auszuwählen, kann mit der Maske {3;0;0;1} genau dieses Profil aus der Datenstruktur herausgefiltert werden. Dies ist eine sehr einfache Maske. Es wird dadurch aufgezeigt, dass in der gewählten Datenstruktur alle Bauteile herausgefiltert werden können. (Rodricks und Heumann 2018c)

Tab. 1: „Path Mapper“-Syntax

Input	Output	Wirkung
{A;B;C}(i)	{A;B;C}(i)	Dies ist ein sogenanntes „Null Mapping“, da hierbei keine Änderungen an der Datenstruktur vorgenommen werden.
{A;B;C}(i)	{0;A;B;C}(i)	Es wird eine zusätzliche Hierarchieebene eingefügt. Die Position und der Pfadname der Hierarchieebene können dabei frei gewählt werden.
{A;B;C}(i)	{A;B}(i)	Eine Hierarchieebene wird entfernt.
{A;B;C}(i)	{0}	Alle Hierarchieebenen werden entfernt. Dies entspricht der „Flatten Tree“-Komponente.
{A;B;C}(i)	{A;B;C;i}(0)	Alle Objekte werden in einen eigenen Pfad geschrieben. Dies entspricht der „Graft Tree“-Komponente.

Quelle: Eigene Darstellung

Wie Rutton 2013 schreibt, wurden mit Grasshopper Build 0.9.0063 (circa November 2013) weitere, komplexere Abfragemasken implementiert. So können mehrere Pfade mit einer Abfrage ausgewählt werden. Folgende Semantiken stehen dadurch für das „Split Tree"-, „Path Compare"- und „Replace Path"-Tool zur Verfügung: (Rutton 2013; Rodricks und Heumann 2018c)

- {...;(X,Y);...} - wählt alle Pfade mit dem Wert X oder Y aus der entsprechenden Hierarchieebene
- {...;(X to Y);...} - wählt alle Pfade mit Werten aus dem Wertebereich [X;Y] aus der entsprechenden Hierarchieebene
- {...;*;...} - wählt alle Pfade aus der entsprechende Hierarchieebene
- {...;!(x);...} - wählt alle Pfade außer die mit dem Wert X aus der entsprechenden Hierarchieebene
- Regeln können mit einem „or" miteinander verknüpft werden (z. B. {...;(0) or (3) or (2);...}). Alternativ kann auch eine Liste von Masken übergeben werden. Dann werden alle Pfade gewählt, die einer der beiden Masken entsprechen.

Diese Regeln sind auch auf die Objekte innerhalb der Listen anwendbar. Dabei ist die entsprechende Notation von eckigen Klammern umgeben. So können beispielhaft mit der Maske {*}[0 to 3] die ersten drei Objekte aus allen Listen des gesamten Baumes zurückgegeben werden. (Rutton 2013)

Die Funktion weiterer Komponenten des Datenmanagements werden an dieser Stelle nicht erläutert. Eine kurze Beschreibung kann jedoch dem Kapitel 3.4.1 entnommen werden.

3.5 LOD einer parametrischen Pfosten-Riegel-Fassade

Folgend soll eine kurze Einordnung der zu erstellenden parametrischen Pfosten-Riegel-Fassade in das von der BIMForum (2018) spezifizierte LOD vorgenommen werden. Es ist zu beachten, dass nach BIMForum (2018: S. 10) eine Unterscheidung zwischen Level of Development (LOD) und Level of Detail (LoD) vorgenommen werden muss. Demnach beschreibt LOD den Entwicklungsgrad, also den Grad, in dem die Elementgeometrie und daran angehängte Informationen durchdacht wurden. Level of Detail (LoD) hingegen beschreibt hauptsächlich die Menge der Details im Modellelement. Das BIMForum (2018) versteht den LoD als Input für das Element und den LOD als zuverlässigen Output.

Augrund dieser Unterscheidung ist für die praktische Ausarbeitung dieser Arbeit zu beachten, dass es sich um ein Geometriemodell handelt. Dies ist darin begründet, dass die Ausarbeitung mit einer reinen Geometriemodellierungssoftware erzeugt wurde. Somit sind keine nicht modellierten Informationen an die digitalen Bauteile verknüpft. Daher wird das erzeugte Modell besser durch die LoD beschrieben. Wie in Kapitel 3.4.2.2 erläutert wird, ist es jedoch möglich Informationen über die Datenstruktur zu implizieren. Dadurch kann dennoch zwischen den einzelnen Bauteilkomponenten differenziert werden. Für die weitere Entwicklung des parametrischen Fassadenknotens ist es wichtig, eine Schnittstelle zu einer BIM-Software zu schaffen und explizite Informationen mit dem Modell zu vereinen. Im Rahmen dieser Diplomarbeit wurde dieser Schritt allerdings nicht ausgearbeitet.

Eine Einordnung in die LOD wird an dieser Stelle trotzdem vorgenommen. Dabei wird aufgezeigt, welche Informationen implementiert werden müssen, um zwischen LoD und LOD zu unterscheiden. In Tab. 2 ist eine Übersicht über die Stufen der vom BIMForum (2018) spezifizierten LOD gegeben. Die Definitionen der Stufen entsprechen dabei den BIM-Protokoll-Dokumenten des American Institute of Architects (2013).

Nach der Definition des BIMForum (2018) konnte das LOD 100 bereits im Entwurfsmodell der Projektarbeit des Autors erzielt werden. Die einzelnen Fassadenelemente wurde in Form von Flächen im Modell repräsentiert. Die Anzahl der Flächen entspricht dabei, nicht der Anzahl der repräsentierten Bauteile. Diese Stufe der LOD ist in Abbildung 18 (a) und (b) (zu finden auf Seite 35) dargestellt. LOD 200 wurde auch in der Projektarbeit erreicht. Hierfür wurden die einzelnen Bauteilkomponenten eines Fassadenknotens durch Zylinder repräsentiert. Dabei wurde die ungefähre Menge, Größe, Form, Lage und Ausrichtung der Bauteile beachtet. Ein Messen dieser Attribute direkt aus dem Modell ist jedoch nicht möglich. Diese Repräsentation ist in Abbildung 18 (c) dargestellt.

Im Rahmen dieser Diplomarbeit soll das LOD weiter erhöht werden. Durch das Implementieren von realen Fassadenprofilen anstelle von kreisförmigen Querschnitten wird das direkte Messen aus dem Modell ermöglicht. So kann das LOD 300 erreicht werden. Nach der Interpretation des BIMForums ist an dieser Stelle die Implementierung eines Projektursprungs und die Positionierung des Bauteils in Bezug auf diesen notwendig. Da in dieser Arbeit jedoch kein gesamtes Gebäude modelliert wird, ist dieser Schritt nicht möglich. Ansatzweise werden auch die Schnittstellen zu den anderen Gebäudesystemen dargestellt, wie etwa die Pressleisten und die Verbindung der Fassadenknoten mit den Fassadenprofilen. Die Darstellung der Bauwerksanschlüsse ist noch nicht implementiert, sodass das LOD 350 noch nicht erreicht werden konnte.

Tab. 2: Stufen der LOD

Stufe	Beschreibung
LOD 100	– Repräsentation des Elements im Modell mit einem Symbol oder einer anderen generischen Darstellung – Anforderungen der LOD 200 nicht erfüllt – Attribute – bspw. Kosten/m^2 – können aus anderen Elementen abgeleitet werden – *keine geometrische Darstellung* – *zeigt Existenz eines Bauteils an – nicht die Form, Größe, Position* – *Informationen sind als Annäherung zu betrachten*
LOD 200	– Repräsentation des Elements im Modell als generisches System, Objekt oder Baugruppe mit ungefähren Mengen, Größen, Form, Lage und Ausrichtung – nicht-grafische Informationen können an das Element angehängt sein – *Elemente sind generische Platzhalter* – *alle aus dieser LOD abgeleiteten Informationen sind als Annäherung zu betrachten*
LOD 300	– Repräsentation des Elements im Modell als spezifisches System, Objekt oder Baugruppe in Bezug auf Menge, Größe, Form, Lage und Ausrichtung – nicht-grafische Informationen können an das Element angehängt sein – *Menge, Größe, Form, Lage und Ausrichtung des Elements kann direkt aus dem Modell gemessen werden – ohne Verweis auf nicht modellierte Informationen* – *definierter Projektursprung und Positionierung des Elements in Bezug auf den Projektursprung*
LOD 350	– wie LOD 300 – zusätzlich: Darstellung der Schnittstelle zu anderen Gebäudesystemen – *Modellierung von Teilen, die für die Koordination mit benachbarten oder befestigten Elementen notwendig sind* – *Menge, Größe, Form, Lage und Ausrichtung des Elements kann direkt aus dem Modell gemessen werden – ohne Verweis auf nicht modellierte Informationen*
LOD 400	– Repräsentation des Elements im Modell als bestimmtes System, Objekt oder eine bestimmte Baugruppe in Bezug auf Menge, Größe, Form, Lage und Ausrichtung mit Detail-, Fertigungs-, Montage- und Installationsinformationen – nicht-grafische Informationen können an das Element angehängt sein – *Modellierung des dargestellten Bauteils mit ausreichender Detailgenauigkeit für die Herstellung* – *Menge, Größe, Form, Lage und Ausrichtung des Elements kann direkt aus dem Modell gemessen werden – ohne Verweis auf nicht modellierte Informationen*
LOD 500	[nicht in Verwendung] – feldgeprüfte Repräsentation in Bezug auf Menge, Größe, Form, Lage und Ausrichtung – nicht-grafische Informationen können an das Element angehängt sein – *nach BIMForum nicht definiert*

Kursiv: Interpretation nach BIMForum
Quelle: Eigene Darstellung, Inhalt in Anlehnung an BIMForum (2018: S. 12 f.) und American Institute of Architects (2013), zitiert nach BIMForum (2018: S. 12 f.)

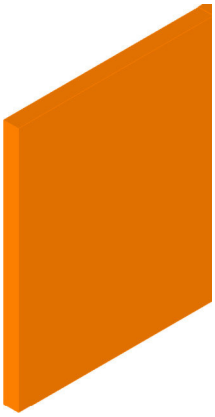
Des Weiteren wird an dieser Stelle untersucht, welche Anforderungen an Pfosten-Riegel-Fassaden notwendig sind. Nach der Spezifikation des BIMForum (2018: S. 77) gelten für Vorhangfassaden die selben Voraussetzungen wie für außenliegende Fensterwände. Die Anforderungen sind in Tab. 3 zusammengefasst. In Abbildung 24 sind die einzelnen LOD Stufen für eine Fensterwand dargestellt.

Die Spezifizierung der Fensterwand und das in Abbildung 24 gegebene Beispiel der verschiedenen LOD-Stufen einer Fensterwand lässt sich auf den Fall des parametrischen Fassadenknotens übertragen. Demnach kann durch die Implementierung der realen Querschnittsprofile ein LoD erzielt werden, der dem der LOD 400 entspricht. Dies ist darin begründet, dass so die einzelnen Bauteilkomponenten detailliert dargestellt und differenziert werden können. Beispielsweise werden in der praktischen Ausarbeitung einzelne Dichtungen und Profile modelliert. Gleichzeitig zeigt die Spezifikation auf, dass zum Erreichen der LOD 400 weitere Hürden – beispielsweise die Schnittstelle zwischen Fassade und Gebäude – zu bewältigen sind und weitere nicht modellierbare Informationen – beispielsweise die einzelnen Funktionen der Bauteile – implementiert werden müssen. Aus diesem Gesichtspunkt kann der ausgearbeitete Fassadenknoten circa in der LOD 300 eingeordnet werden.

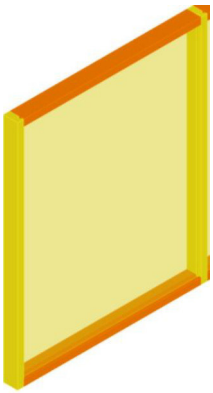
Tab. 3: LOD einer Vorhangfassade

Stufe	**Beschreibung**
LOD 100 (B20)	– solides Massenmodell, das das gesamte Gebäudevolumen darstellt oder schematische Wandelemente (nicht durch Typ oder Material unterscheidbar) – Einbautiefe / Dicke sind noch flexibel
LOD 200	– allgemeine Wandobjekte – gesamte Einbautiefe des Elements – Darstellung durch ein einzelnes Objekt – Layout und Position flexibel
LOD 300	– spezifizierte Position und Ausrichtung der Glasfläche – Nennflächenmaße und Dicke der Verglasung – Abstand, Lage, Grüße und Ausrichtung der Pfosten – bedienbare Komponenten definiert und im Modell integriert – bspw. Fenster, Jalousien und Türen
LOD 350	– Pfostenformen und -geometrien definiert – Verankerungslayouts und -typen definiert und modelliert – aktuelle Abmessungen
LOD 400	– komplette Pfosten-Riegel-Extrusionsprofile – Schnittstellendetails zwischen Wandsystemen (innen) und Wand- und Stützsystemen

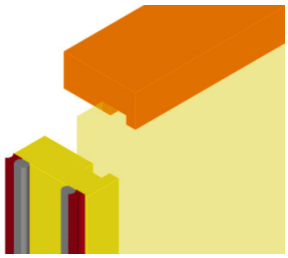
Quelle: In Anlehnung an BIMForum (2018: S. 63, 78 – 81)



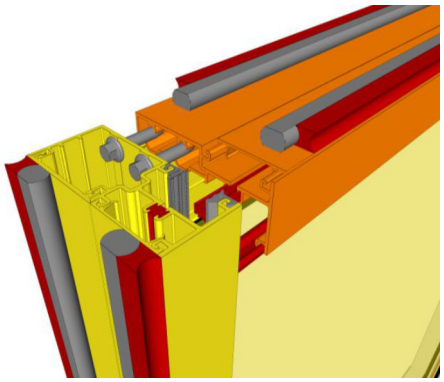
(a) LOD 200



(b) LOD 300



(c) LOD 350



(d) LOD 400

Abbildung 24: Level of Development einer Fensterwand
Quelle: BIMForum (2018: S. 79 ff.)

4 Erzeugung Profilquerschnitte

Nachdem die grundlegenden Konzepte von Grasshopper in Kapitel 3.4 und der Aufbau von Pfosten-Riegel-Fassaden in Kapitel 2.4.1 erklärt wurden, kann in diesem Kapitel mit der Erläuterung der Grasshopper-Definition zur Erzeugung von parametrischen Fassadenknoten begonnen werden.

4.1 Überblick über den Algorithmus

An dieser Stelle soll ein kurzer Überblick über die Wahl der Bauteilkomponenten und dem Erzeugen der dazugehörigen Geometrie gegeben werden. In Abbildung 25 ist der Programmaufbau schematisch dargestellt. Durch die Pfeile wird der grobe Informationsfluss verdeutlicht. In Kapitel 4 werden die dunkelblau dargestellten Komponenten erläutert. Diese sind für die Wahl und Generierung der Profilquerschnitte zuständig. Kapitel 5 erläutert, wie die geladenen Komponenten transformiert werden müssen um einen sinnvollen Querschnitt darzustellen. Abschließend wird in Kapitel 6 gezeigt, wie ein 3-D-Fassadenknoten aus den Querschnitten erzeugt wird.

Als erstes werden Inputdaten in der Algorithmus eingegeben. Die Daten können einerseits aus der Analyse eines digitalen Gebäudemodells extrahiert werden. Dieser Vorgang wurde in der Projektarbeit des Autors gezeigt. Andererseits kann, wie in der praktischen Ausarbeitung dieser Diplomarbeit dargestellt, der Dateninput manuell durch den Nutzer vorgenommen werden. Im Algorithmus wird dies im Bereich „Input“ umgesetzt. Erläuterungen zu diesem befinden sich in Kapitel 4.2

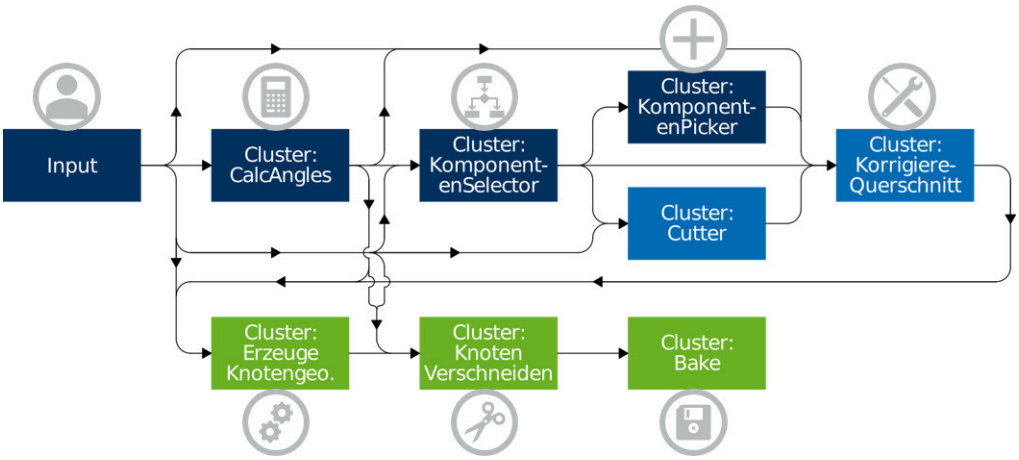


Abbildung 25: Schematischer Aufbau des Algorithmus
Quelle: Eigene Darstellung

Nachdem alle Inputparameter bekannt sind, beginnt der Algorithmus damit, die Winkel zwischen den einzelnen Fassadenprofilen zu berechnen. Zusätzlich wird an dieser Stelle eine Validierung der Eingangsdaten vorgenommen. Die Umsetzung erfolgt in dem in Kapitel 4.3 erläutertem „Cluster: CalcAngles“.

Als dritter Schritt können die verschiedenen Komponenten, die im Fassadenknoten verbaut werden, gewählt werden. Hierzu wird in Kapitel 4.4 das „Cluster: KomponentenSelector“ implementiert.

Abschließend für Kapitel 4 kann die Bauteilgeometrie mit dem „Cluster: KomponentenPicker“ geladen werden. Die Funktionsweise dieses Clusters wird in Kapitel 4.5 erläutert.

Darauffolgend werden in Kapitel 5 Funktionen in den Algorithmus eingefügt, die die erzeugte Geometrie korrigieren und sogenannte Cutter, die zum Verschneiden der Knotengeometrie benötigt werden, einführt. In Abbildung 25 ist dieser Bereich hellblau dargestellt.

Im „Cluster: Cutter“ werden, parallel zur Erzeugung der Komponentengeometrie, die Geometrien der Cutter generiert. Die Definition des Begriffs Cutters und die Vorgehensweise bei deren Generierung sind dem Kapitel 5.1 zu entnehmen.

Anschließend können die Geometrien der Bauteilkomponenten sowie die Geometrien der Cutter korrigiert werden. Damit ist gemeint, dass die Positionen der Komponenten an die verschiedenen Parameter – wie etwa die Elementdicke – angepasst werden. Die Ausführung dieses Vorgangs ist in Kapitel 5.2 erläutert.

An dieser Stelle liegen die Querschnitte der einzelnen Profile fertig vor. In Kapitel 6 kann daher mit der Erzeugung eines 3-D-Fassadenknotens begonnen werden. In Abbildung 25 sind die dafür implementierten Komponenten grün dargestellt.

Zunächst werden mit dem „Cluster: Erzeuge Knotengeometrie“ die 2-D-Querschnitte zu 3-D-Profilen erweitert und entsprechend der Eingabeparameter positioniert. Dieser Vorgang ist in Kapitel 6.1 ausgeführt.

Um die Geometrie des Knotens fertig zu stellen, wird in Kapitel 6.2 das „Cluster: Knoten Verschneiden“ implementiert.

Abschließend wird in Kapitel 6.3 erläutert, wie die Geometrie des digitalen Fassadenknotens mithilfe der Bake-Funktion von Grasshopper an Rhinoceros 6 übertragen werden kann.

4.2 Eingabeparameter

Da in dieser Diplomarbeit keinerlei Daten aus einem Entwurfsmodell generiert werden (vgl. Kapitel 3.3), sollen die Input-Parameter durch den Anwender definiert werden. Hierfür steht der in Abbildung 26 dargestellte Bereich innerhalb der Grasshopper-Definition zur Verfügung. Innerhalb des Bereichs hat der Nutzer zunächst die Möglichkeit fünf Winkel zu definieren. Diese Winkel dienen zur Veränderung der Richtungen, in denen die Fassadenprofile aufeinandertreffen. Die Winkel „MPfosten“ 26.1 und „MRiegel“ 26.2 ermöglichen den Pfosten- und Riegelprofilen eine erste Rotation um die x-Achse für die Pfosten, beziehungsweise um die z-Achse für die Riegel. Dieser Winkel wirkt sich dabei immer in gleicher Weise auf beide Pfosten beziehungsweise beide Riegel aus. Verschieden große Rotationswinkel für die jeweils gegenüberliegenden Profile sind in dieser ersten Rotation nicht notwendig, da mithilfe der zweiten Rotationswinkel alle Geometriestellungen der Profile zueinander erreicht werden können. Eine unterschiedliche Rotation beider Profile kann auch durch die Rotation der gesamten Knotengeometrie ausgedrückt werden. Daher wird von der getrennten Implementierung abgesehen.

Die zweiten Rotationswinkel „SPfosten“ 26.3, „SRiegelR“ 26.4 und „SRiegelL“ 26.5 ermöglichen für die Pfosten und Riegel eine zweite Rotation um die y-Achse. Für die Pfosten findet erneut eine gemeinsame Rotation beider Profile statt. Der Grund dafür ist, wie beim

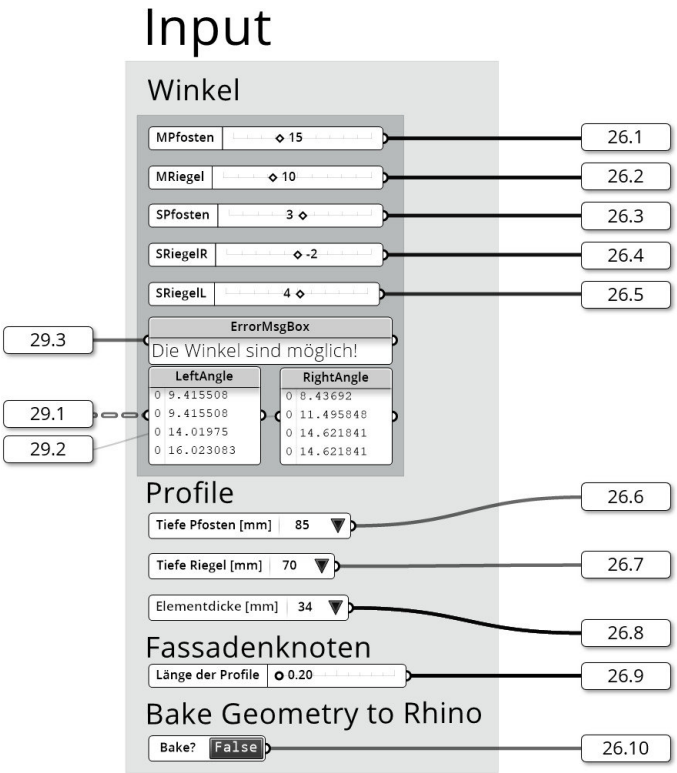


Abbildung 26: Eingabeparameter für den parametrischen Fassadenknoten
Quelle: In Anlehnung an Grasshopper

Rotationswinkel „MPfosten“, dass dies in einer Rotation der Gesamtgeometrie endet. Um nicht nur spiegelsymmetrische Fassadenknoten zu erhalten, wird den beiden Riegeln an dieser Stelle eine unabhängige Rotation ermöglicht. Dabei beschreibt „SRiegelR“ die Rotation des rechten Profils und „SRiegelL“ die des Linken. In Abbildung 27 sind die einzelnen Vektoren und Winkel dargestellt.

Unterhalb der Einstellungsmöglichkeiten für die Winkel befindet sich ein Panel mit dem Namen „ErrorMsgBox“. Der Input für diese Komponenten stammt von 29.3. Dieses Panel soll den Nutzer mit einer ersten Einschätzung versorgen, ob der Fassadenknoten mit den gewählten Parametern erfolgreich erzeugt werden kann. Hierbei erfolgt eine Prüfung, ob die Winkel zwischen den Profilen im Gültigkeitsbereich des Fassadensystems liegen. Die Gültigkeitsbereiche für das gewählte Fassadensystem wurden bereits in Kapitel 2.4.1 beschrieben. Es ist zu beachten, dass nicht alle Fehlerquellen des Algorithmus durch die „ErrorMsgBox“ ausgegeben werden. Zum Beispiel können Geometriestellungen auftreten, bei denen die beiden Winkel zwischen dem Profil und den beiden Fassadenelementen zu stark voneinander abweichen. In diesem Fall kann der Unterschied nicht mehr durch den Einsatz von speziellen Dichtungen ausgeglichen werden. Die Folge ist, dass kein Tragprofil oder keine Pressleiste gefunden werden kann, die die Bedingungen der Winkel erfüllt. Obwohl der Feh-

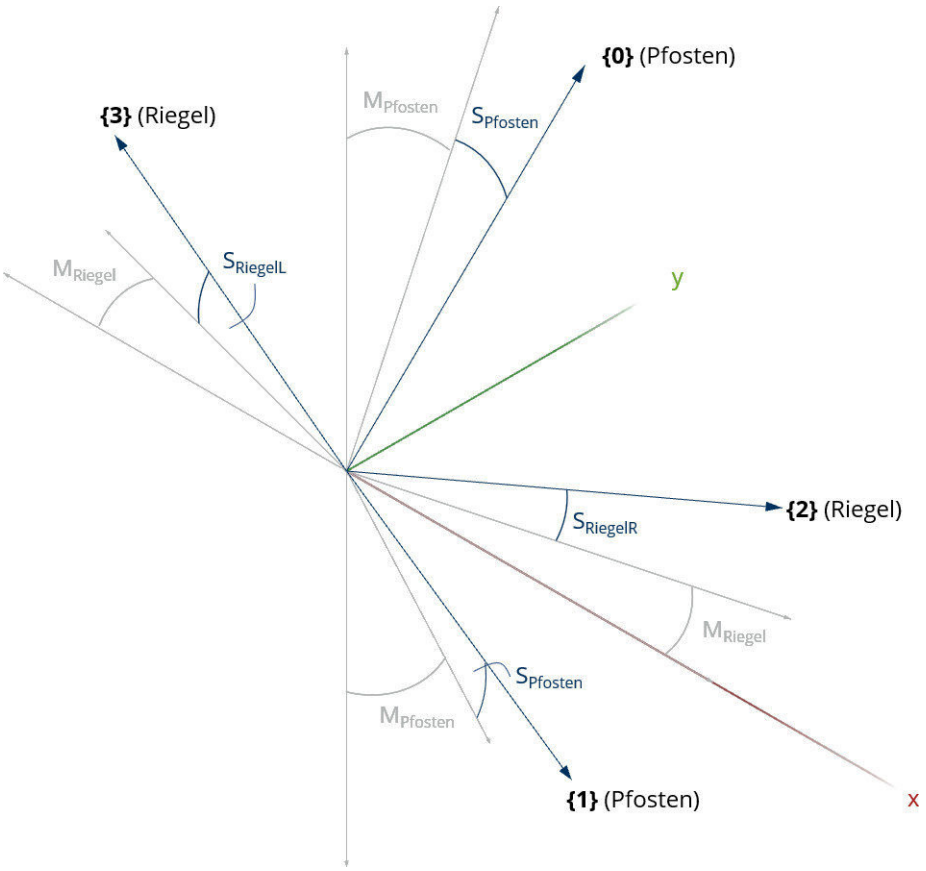


Abbildung 27: Profilvektoren und Inputwinkel
Quelle: In Anlehnung an Rhinoceros 6

ler nicht in der ErrorMsgBox ausgegeben wird, wird er in den eigenen Programmierungen bei der Auswahl der Profilkomponenten berücksichtigt, vgl. Kapitel 4.4.1. Ein Umarbeiten zum Zweck der Ausgabe auf einem Panel ist genauso möglich.

Die beiden Panels „LeftAngle“ und „RightAngle“ geben die im letzten Absatz behandelten links und rechts anliegenden Winkel aus. Die Daten werden von 29.1 beziehungsweise 29.2 bezogen. Mithilfe der Winkel kann der Nutzer eine erneute Prüfung der Funktionsfähigkeit des Algorithmus vornehmen.

Unterhalb der Winkeleinstellungen können vom Anwender verschiedene Profiltiefen für Pfosten 29.6 und Riegel 29.7 gewählt werden. Einige Pfostentiefen sind mit einem * gekennzeichnet. Dies bedeutet, dass für den Pfosten keine Variante eines variablen Profils vorliegt. Für beide Pfosten und beide Riegel ist jeweils nur eine Tiefe wählbar. Eine Implementierung von unterschiedlich tiefen Riegeln wäre technisch umsetzbar. Für die Pfosten könnte dies zu konstruktiven Problemen hinsichtlich der Bauteilgeometrie führen, da die Pfostenprofile im späteren Fassadenknoten durchgängig ausgeführt werden. In Abbildung 28 sind die Parameter der Profilquerschnitte dargestellt. Auch die folgend erläuterte Elementdicke kann aus dieser Abbildung entnommen werden.

Wie bereits in Kapitel 2.4.1 beschrieben, können bei dem Fassadensystem unterschiedliche Elementdicken verwendet werden. Mit dem Dropdownmenü „Elementdicke“ 29.8 können unterschiedliche Elementdicken eingestellt werden. Dies hat zur Folge, dass unterschiedliche Dichtungen und unterschiedliche Isolatoren gewählt werden müssen.

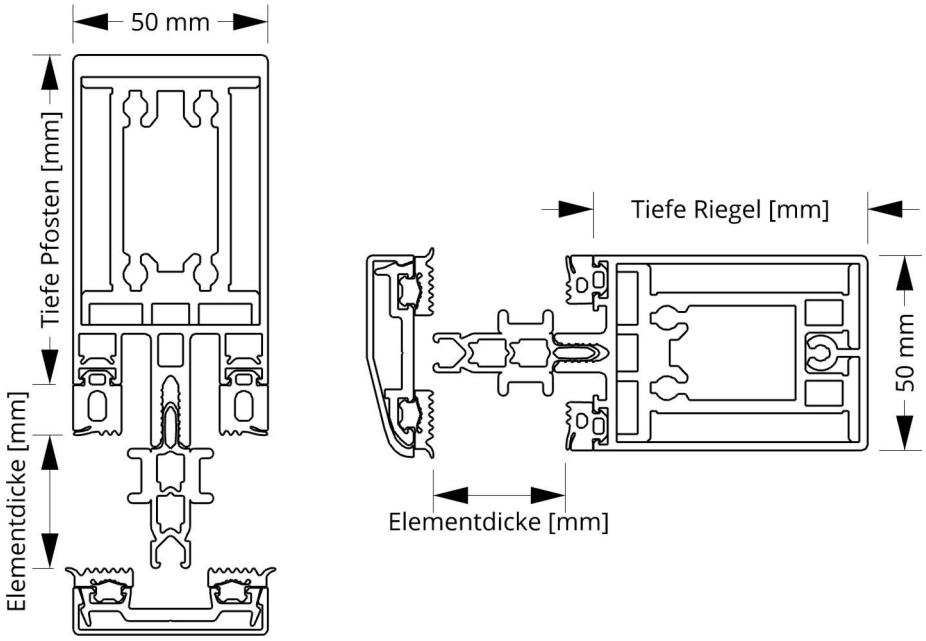


Abbildung 28: Parameter der Profilquerschnitte
Quelle: Screenshot aus Rhinoceros 6

Zuletzt gibt es spezielle Einstellungen für den Fassadenknoten. Hier kann eingestellt werden, welche Länge die Profile 29.9, vom Mittelpunkt des Knotens ausgehend, aufweisen sollen. Außerdem kann die Einstellung aktiviert werden, dass die in Grasshopper erzeugte Geometrie nach Rhinoceros 6 übertragen wird. Hierzu muss die „Toggle Switch"-Komponente „Bake?" auf TRUE gestellt werden. Der boolsche Wert wird in 29.10 weitergegeben.

4.3 Berechnung der Winkel zwischen den Profilen

Um geeignete Komponenten wählen zu können, werden die Winkel zwischen dem Profil und den angeschlossenen Elementen benötigt. Diese Winkel können durch die Stellung der Profile zueinander ermittelt werden. Das in Abbildung 29 dargestellte „Cluster: CalcAngles" übernimmt die Berechnung dieser Winkel.

Zur Berechnung der Winkel benötigt das Cluster die aus Abbildung 26 stammenden Eingabewinkel 26.1 - 26.5. Nach der Berechnung gibt das Cluster die berechneten Werte für den linken Winkel „LeftAngle" 29.1 und den rechten Winkel „RightAngle" 29.2, eine Fehlermeldung „ErrorMsg" 29.3 und die zur Berechnung benötigten Vektoren 29.4 zurück. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/01_ClusterCalcAnglesKomplett.jpg" ist die komplette Darstellung des Clusterinneren gegeben.

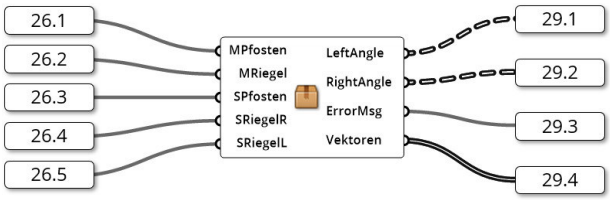


Abbildung 29: Das „Cluster: CalcAngles"
Quelle: In Anlehnung an Grasshopper

4.3.1 Erzeugung der Profilvektoren

Innerhalb des Clusters werden zunächst die in Abbildung 27 dargestellten Vektoren erzeugt. An dieser Stelle werden zunächst die Vektoren vor der Rotation betrachtet. Diese zeigen entlang der Richtung der späteren Profile. Wie in Abbildung 30 ersichtlich ist, erfolgt die Generierung der Vektoren innerhalb eines Panels als Texteingabe. Das Ergebnis davon wird gegraftet[2], sodass jeder Vektor in einem eigenen Pfad enthalten ist. Dadurch wird die erste Hierarchieebene, wie bereits in Kapitel 3.4.2.2 beschrieben, erzeugt. In Abbildung 31 ist die beschriebene Datenstruktur schematisch veranschaulicht. Es ist zu erkennen, dass

[2] unter „graften" wird in dieser Arbeit, in Anlehnung an die Grasshopper-Komponente „Graft", das Hinzufügen einer Hierarchieebene verstanden

die vier Vektoren an der Position der späteren Bauteile gespeichert werden. Zusätzlich soll jedem Vektor eine Ebene gegeben werden, sodass die Ausrichtung der Profile bestimmt werden kann. Diese Ebene soll der Ebene entsprechen, in der die Fassadenelemente liegen würden, wenn diese in einem 180 ° Winkel zueinander stehen. Diese Ebene, im Folgenden Referenzebene genannt, ist in Abbildung 32 als Ebene A-A dargestellt. Bevor die Profile rotiert werden, liegen alle Ebenen innerhalb der x-z-Ebene. Somit können die Ebenen über einen Ursprungspunkt (0|0|0) und eine z-Richtung (0|1|0) definiert werden. Da nur die Beträge der Winkel berechnet werden, ist dabei die Ausrichtung der Ebenennormalen nicht von Bedeutung. Diese Ebene wird benötigt, um die Abweichungen von der

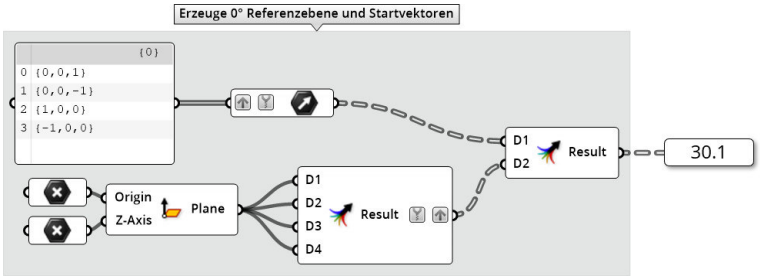


Abbildung 30: Erzeugung der Vektoren und Referenzebenen
Quelle: In Anlehnung an Grasshopper

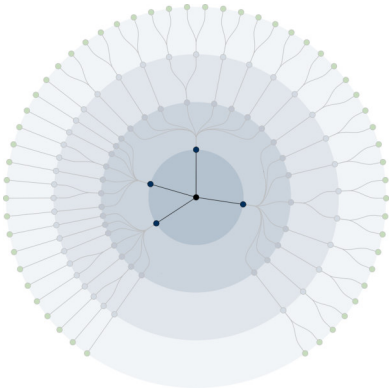


Abbildung 31: Datenstruktur der Profilvektoren
Quelle: Eigene Darstellung

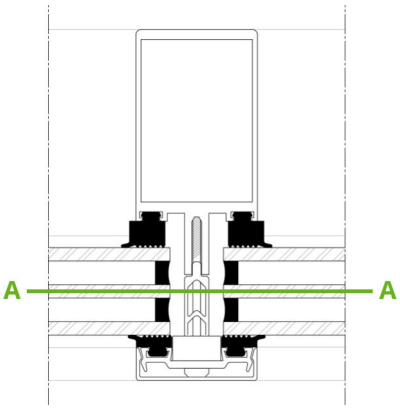


Abbildung 32: Referenzebene
Quelle: In Anlehnung an Schittich et al. (2006: S. 164)

Referenzebene und somit die benötigten Profile und Dichtungen zu bestimmen. Die Referenzebenen werden auch in die bauteilorientierte Datenstruktur gegraftet – vgl. Abbildung 31 – und anschließend mit den Vektoren zusammengeführt. Da die Vektoren zuerst an die „Merge“-Komponente übergeben werden, erhalten sie die Position {X}(0) und die Ebenen {X}(1). Mit X $\in$ [0;3] für das jeweilige Bauteil. Die Reihenfolge der Bauteile entspricht dabei der in Kapitel 3.4.2.2 aufgebauten und in Abbildung 21 (zu finden auf Seite 40) dargestellten Datenstruktur. Sie wird in 30.1 weitergegeben.

4.3.2 Rotation der Profilvektoren

Nachdem die Vektoren und Referenzebenen 30.1 erstellt wurden, werden diese entsprechend der fünf, in Kapitel 4.2 eingeführten Winkel 26.1 bis 26.5 rotiert. Wie Abbildung 33 zeigt, werden die Winkel in eine geeignete Datenstruktur überführt, um die Datenstruktur der Vektoren und Referenzebenen erfolgreich modifizieren zu können. Die Datenstruktur ist dabei analog zu Abbildung 31 aufgebaut und entspricht somit {X}(i), wobei X für die vier Bauteile steht. Hierfür werden die Winkel zunächst in der Datenstruktur {0}(X) abgelegt, indem die Winkel in der richtigen Reihenfolge an die „Merge“-Komponente übergeben werden. Diese Winkeldatenstruktur wird innerhalb der „Rotate Axis“-Komponenten gegraftet, sodass sie der Struktur {X}(0) entsprechen. Dadurch wirken sich die Winkel sowohl auf die Vektoren, als auch auf die Referenzebenen aus. Wie in Kapitel 4.2 beschrieben, werden die Vektoren und Referenzebenen zunächst um die x- beziehungsweise z-Achse rotiert, bevor sie in der zweiten „Rotate Axis“-Komponente um die y-Achse rotiert werden. Da diese Achsen für den parametrischen Fassadenknoten in allen Fällen gelten, werden die Achsen direkt in den beiden Komponenten gespeichert. Dabei wurden die Achsen der gegenüberliegenden Vektoren jeweils in die negative Richtung gewählt. So drehen sich die beiden gegenüberliegenden Bauteile bei gleichen Winkelvorzeichen aufeinander zu. Bei den Rotationen der Pfostenprofile gilt es weiterhin zu beachten, dass sich beide Profile nicht zueinander verdrehen. Dies hat den Hintergrund, dass das Pfostenprofil im fertigen Fassadenknoten durchgängig eingesetzt werden soll. Werden die beiden Profile zueinander verdreht, können sie nicht mehr vollständig verschnitten werden. Die Vektoren, die nach der Rotation vergleichbar zu Abbildung 27 sind, werden in 33.1 weitergegeben.

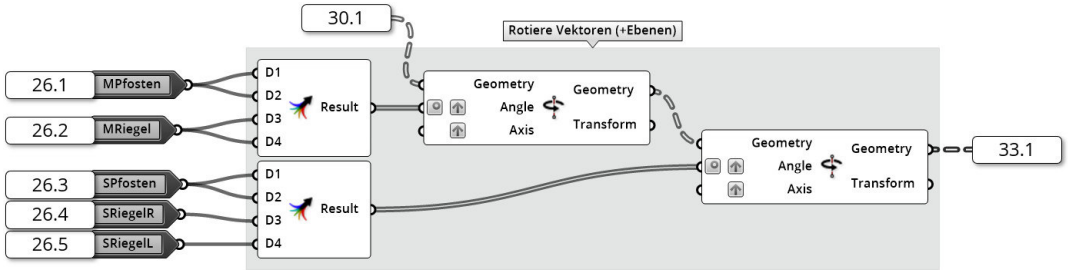


Abbildung 33: Rotation der Vektoren und Referenzebenen
Quelle: In Anlehnung an Grasshopper

4.3.3 Umstrukturierung der Datenstruktur zur Berechnung

Im Folgenden muss die Datenstruktur aus 33.1, die bauteilorientiert vorliegt, zur Berechnung der Winkel optimiert werden. Dieser Vorgang ist in Abbildung 34 dargestellt. Zunächst werden die Vektoren und Referenzebenen voneinander getrennt. Dies wird über die „List Item"-Komponente ermöglicht. Sie gibt alle Objekte mit einem gewissen Objektindex zurück. Mit dem Index [0] wird auf die Vektoren und mit [1] auf die Referenzebenen zugegriffen. Um das Umsortieren der Elemente zu erleichtern, werden die Datenbäume wieder in Listen geflattet[3]. Damit gilt die Datenstruktur {0}(X), mit X als Bauteil. Mit einer weiteren „List Item"-Komponente werden die Listen in die vier einzelnen Elemente zerlegt, sodass ein Umsortieren mit der „Merge"-Komponente stattfinden kann. Dabei sollen die räumlich benachbarten Vektoren auch in der Datenstruktur benachbart werden. Um dies zu ermöglichen, müssen die Elemente [2] und [3] ihre Position tauschen. Dadurch werden die Vektoren dem Uhrzeigersinn nach sortiert. Anschließend werden die Referenzebenen 34.1 an die Berechnung der Winkel übergeben. Die Vektoren 34.2 werden zuvor weiterverarbeitet.

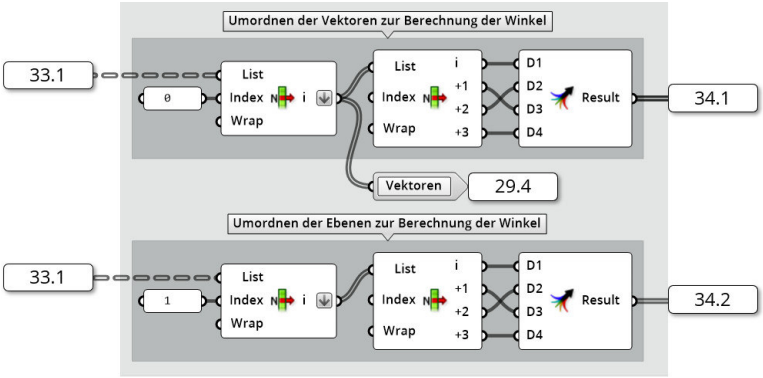


Abbildung 34: Umstellung der bauteilorientierten Struktur in die Vektororientierte
Quelle: In Anlehnung an Grasshopper

4.3.4 Erzeugung der Elementebenen

Da nicht die Winkel zwischen den Profilvektoren und den Referenzebenen gesucht werden, sondern die zwischen den Referenzebenen und den Ebenen der realen Fassadenelemente, folgend Elementebenen genannt, müssen Letztere zunächst gebildet werden. Diese Elementebenen können, wie in Abbildung 35 gezeigt, aus jeweils zwei benachbarten Profilvektoren abgeleitet werden. Dazu wird jeweils ein Vektor aus 34.1 und der Vorgänger dieses Vektors zur Bildung einer Ebene genutzt. Auf den Vorgänger kann man mit einer „Shift List"-Komponente zugreifen. Diese verschiebt die Indizes einer Liste um eine gewisse Anzahl „Shift". Um den Vorgänger zu erreichen, ist „Shift" dabei - 1. So erhält man die Ebene des

[3] unter "flatten wird in dieser Arbeit, in Anlehnung an die Grasshopper-Komponente „Flatten", das Entfernen aller Hierarchieebenen verstanden

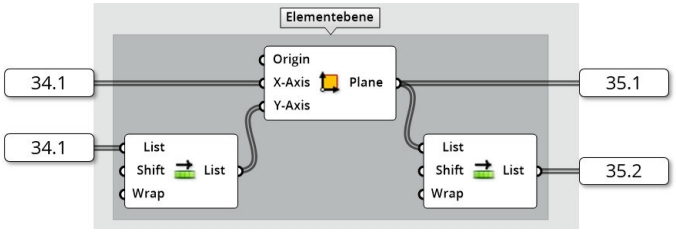


Abbildung 35: Erzeugung der Elementebenen
Quelle: In Anlehnung an Grasshopper

Fassadenelements des Feldes, das im Uhrzeigersinn gesehen, vor dem Vektor vorkommt. In Grasshopper wird die Bildung der Ebene von der „Construct Plane“-Komponente übernommen, der die beiden Vektoren übergeben werden. Zwei der so erzeugten Ebenen sind in Abbildung 36 blau dargestellt. Die gezeigten Elementebenen entsprechen den Ebenen zwischen den Profilen {0} und {2} sowie {0} und {3}. Im Vergleich dazu ist die Referenzebene des Pfostenprofils {0} grau abgebildet. Gesucht wird der Winkel zwischen diesen Elementebenen und der Referenzebene. Hierfür werden einerseits die erzeugten Ebenen in 35.1 weitergegeben. Mit dieser Ebene kann im Beispiel der Abbildung 36 der Winkel zwischen den Profilen {0} und {2} berechnet werden. Andererseits wird auch der Winkel zwischen {0} und {3} benötigt. Um diesen zu bestimmen, wird nicht die Vorgängerelementebene benötigt, sondern die des Nachfolgers. Dazu kann mithilfe einer „Shift“-Komponente auf die Nachfolgerelementebenen zugegriffen werden. Dabei wird der Wert „Shift“ auf 1 gesetzt. Anschließend werden diese Ebenen in 35.2 übergeben.

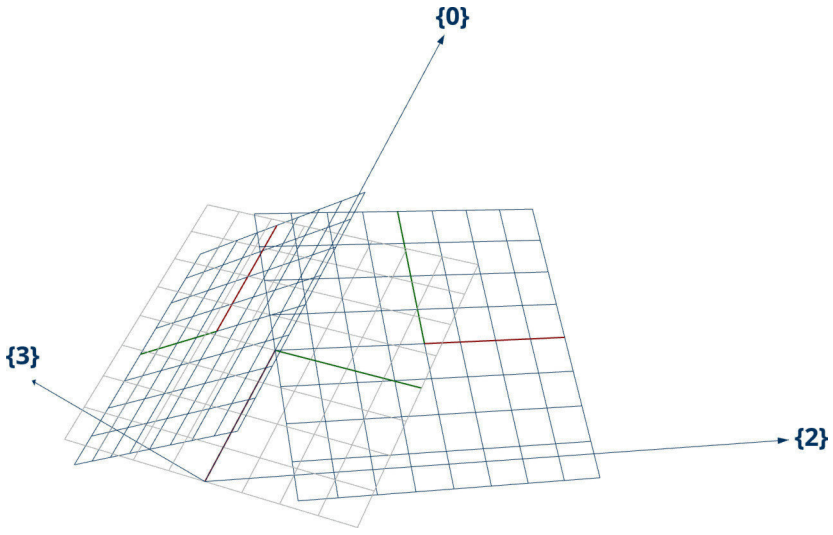


Abbildung 36: Unterschied zwischen Referenz- (grau) und Elementebenen (blau)
Quelle: In Anlehnung an Rhinoceros 6

4.3.5 Berechnung der Winkel

An dieser Stelle sind fast alle Vorbereitungen zur Berechnung der Winkel zwischen der Referenzebene 34.1 und den beiden Elementebenen 35.1 und 35.2 abgeschlossen. Wie Abbildung 37 zeigt, ist der vorletzte Schritt die Umformung der Ebenen in ihre Normalenvektoren. In Grasshopper kann dies umgesetzt werden, indem die Ebenen an eine „Vektor"-Komponente übergeben werden. Anschließend können die Winkel zwischen Normalenvektoren mithilfe der „Angle"-Komponente bestimmt werden. Dabei wird in die Berechnung zwischen der Referenzebene und der Vorgänger- beziehungsweise Nachfolgerelementebene unterschieden. Der Winkel zwischen Referenzebene und dem Vorgänger wird als „LeftAngle" 37.1 weitergeleitet. Der Winkel zum Nachfolger entsprechend „RightAngle" 37.1.

Die Berechnungsergebnisse 37.1 und 37.2 werden wieder in die bauteilorientierte Struktur zurückgeführt. Dieser Vorgang ist in Abbildung 38 dargestellt. Dazu werden, wie zuvor beschrieben, erneut die „List Item"- und „Merge"-Komponente verwendet. Die Ergebnisse werden mit der „Degrees"-Komponente von RAD in GRAD umgewandelt. Zum einen werden die Winkel als Baum aus dem Cluster ausgegeben und zum anderen als 38.1 und 38.2 zur Erzeugung der Fehlermeldung für die in Kapitel 4.2 beschriebenen „ErrorMsgBox"-Panel verwendet. Die Berechnungsergebnisse der Winkel werden, als Tree gegraftet, aus dem Cluster ausgegeben. Die Ausgabe erfolgt dabei an der Stelle 29.1, als „LeftAngle" und 29.2, als „RightAngle".

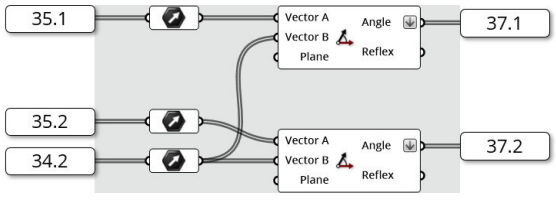


Abbildung 37: Berechnung der Winkel
Quelle: In Anlehnung an Grasshopper

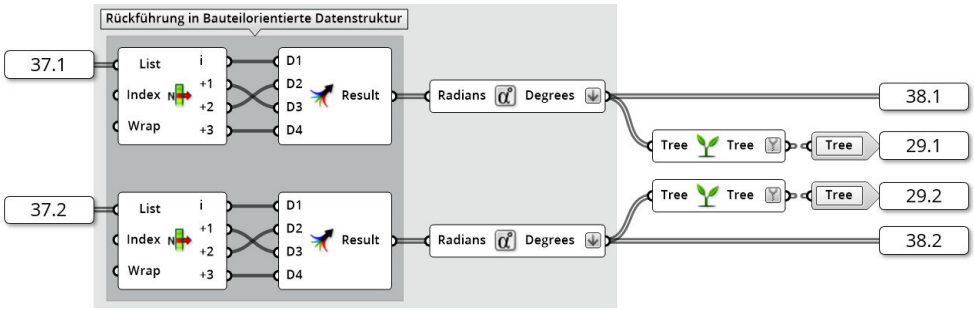


Abbildung 38: Rückführung in die bauteilorientierte Datenstruktur
Quelle: In Anlehnung an Grasshopper

4.3.6 Fehlermanagement

Wie Abbildung 39 zeigt, werden zur Erzeugung der Fehlermeldungen drei selbst geschriebenen Python-Skripte genutzt. Davon sind die linken beiden Skripte das gleiche Skript, sodass nur zwei unterschiedliche Skripte benutzt werden. Die Skripte sind in Anhang Q1 und Q2 beigefügt. Die Funktionsweise des Skripts Q1 „Script: FehlermeldungTrigger“ ist eine Überprüfung der Gültigkeitsbereiche der Winkel mittels if-Bedingungen (vgl. Zeile 15 und 16). Sollten die Gültigkeitsbereiche erfüllt sein, wird der Boolean „error“ auf FALSE, anderenfalls auf TRUE, gesetzt. Damit das Skript die Werte validieren kann, werden 38.1 und 38.2 vorerst mithilfe der „List Item“-Komponente aufgesplittet und anschließend sortiert nach Riegeln (a und b) und Pfosten (c und d) an die beiden „Script FehlermeldungTrigger“ übergeben.

Das zweite Skript Q2 „Script: ErrorMsgBox“ nutzt den Boolean „error“ von beiden vorhergehenden Skripten. Sollte eine der beiden Variablen den Status TRUE enthalten, wird die Nachricht „Bitte überprüfen Sie die Winkel!“ in der „ErrorMsgBox“ aus Abbildung 26 (zu finden auf Seite 53) ausgegeben. Anderenfalls erscheint der Text „Die Winkel sind möglich!“. Die Abfrage, ob einer der beiden boolschen Werte den Status TRUE hat, erfolgt in Zeile 14 des Skripts mit einer if-Bedingung. Mit der Rückgabe der „msg“ 29.3 ist das Cluster abgeschlossen.

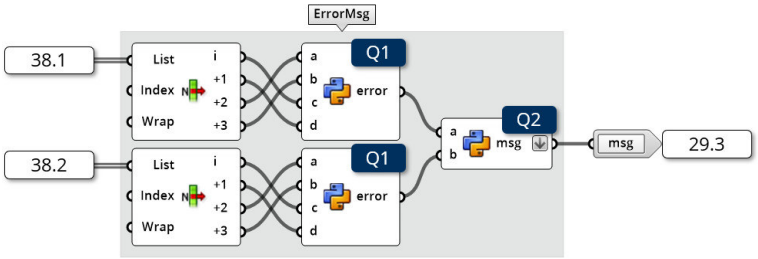


Abbildung 39: Erzeugung der Fehlermeldung „ErrorMsgBox“
Quelle: In Anlehnung an Grasshopper

4.4 Auswahl der Bauteilkomponenten

Nachdem die Winkel zwischen den Fassadenprofilen in Kapitel 4.3 bestimmt und weitere Informationen durch den Nutzer in Kapitel 4.2 ergänzt wurden, können die Komponenten für die einzelnen Profile gewählt werden. Zu diesem Zweck wurde das „Cluster: KomponentenSelector“, das in Abbildung 40 dargestellt ist, implementiert. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/02_ClusterSelectorKomplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Die Auswahl der Komponenten erfolgt dabei mithilfe der Exceltabelle „Schubert06_input.xlsx“, in der die Artikelnummern der ein-

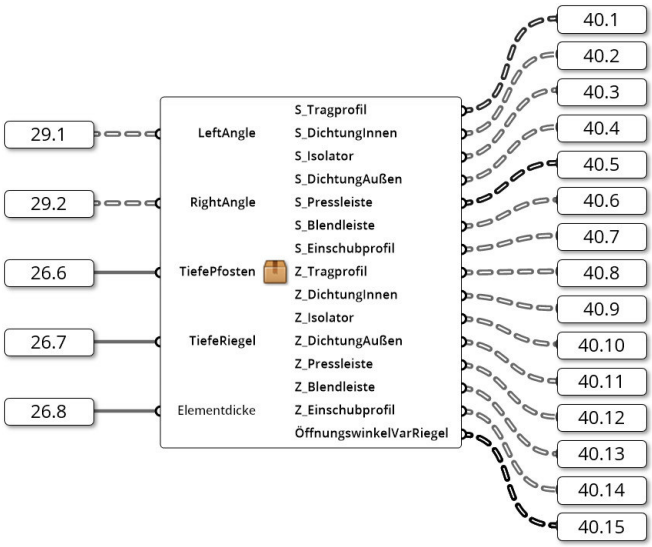
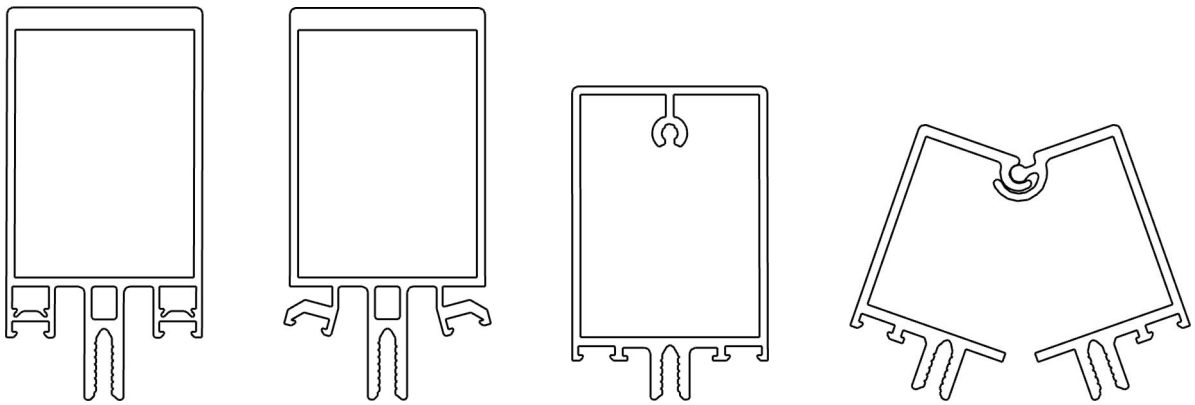


Abbildung 40: Das „Cluster: KomponentenSelector“
Quelle: In Anlehnung an Grasshopper

zelnen Bauteilkomponenten hinterlegt sind. Die Exceltabelle ist in Anhang D6 enthalten. Das „Cluster: KomponentenSelector“ ist für die Auswahl der einzelnen Zeilen- und Spaltennummern aus der Exceltabelle zuständig. Die Artikelnummern und die Bauteilgeometrie werden dann im „Cluster: KomponentenPicker“ ermittelt. Die Exceltabelle ist so aufgebaut, dass sie für jede einzelne Komponente ein eigenes Blatt besitzt. In den Zellen der jeweiligen Blättern sind die Artikelnummern der Komponenten hinterlegt.

4.4.1 Tragprofile

Die Auswahl der Tragprofile erfolgt im Allgemeinen nach zwei Gesichtspunkten. Der erste Gesichtspunkt ist die Tiefe, die die Tragstruktur aufweisen soll. Diese ist im Falle des Pfostens hauptsächlich durch die Statik bedingt. Beim Riegelprofil spielen statische Aspekte eine untergeordnete Rolle, da aus ästhetischen Gründen meist ein Profil mit gleicher Tiefe gewählt wird. Durch die statischen Anforderungen werden gewisse Mindesttiefen vorgegeben. Aufgrund der dafür notwendigen statischen Berechnung, wird im Rahmen dieser Diplomarbeit angenommen, dass der Nutzer eine angemessene Profiltiefe wählt. Dieser Vorgang erfolgt in dem in Kapitel 4.2 beschriebenen „Input“-Bereich. Der zweite Gesichtspunkt ist die Geometrie des Knotenpunktes, beziehungsweise der Gebäudehülle. Da die Fassadenelemente bei einer komplexen Fassadengeometrie nicht in einer Ebene liegen, müssen die Riegel und Pfosten als Übergang zwischen diesen dienen. Da die Profile unterschiedlichste Winkel aufnehmen müssen, kann nicht ein Profil für alle Anwendungsfälle existieren, sodass ein entsprechendes Profil zu wählen ist. Für das Fassadensystem „FW 50+“ stehen, wie in Kapitel 2.4.1.6 beschrieben, variable Pfostenprofile zur Verfügung. Diese können Winkel von bis zu 50 ° (25 ° je Seite) aufnehmen. Dabei sind folgende Abwinke-



(a) Pfosten standard (b) Pfosten variabel (c) Riegel standard (d) Riegel variabel

Abbildung 41: Beispiele von wählbaren Pfosten- und Riegelprofilen
Quelle: Screenshot aus Rhinoceros 6

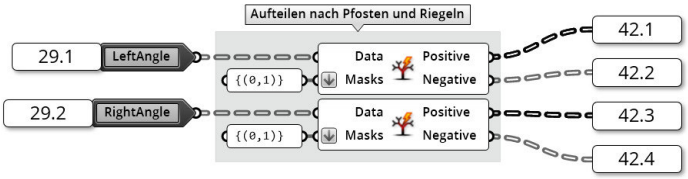


Abbildung 42: Aufteilen der Datenstruktur in Pfosten und Riegel
Quelle: In Anlehnung an Grasshopper

lungen vorhanden: 0 °, 5 °, 10 °, 15 °, 20 °, 25 ° (je Seite). In Abbildung 41 (b) ist ein variables Pfostenprofil und ein standardmäßiges in (a) gegeben. Die Bereiche zwischen den Winkeln werden mithilfe von variablen Abdichtungen ausgeglichen. Die Wahl und Anforderungen der Dichtungen werden später in Kapitel 4.4.2 besprochen.

Wie in Abbildung 40 verdeutlicht, werden dazu der „LeftAngle“ 29.1, „RightAngle“ 29.2, „TiefePfosten“ 26.6, „TiefeRiegel“ 26.7 und „Scheibendicke“ 26.8 als Input benötigt. Nachdem die Komponenten gewählt wurden, werden diese als Zeilen- („Z_Bauteil“ 40.1 - 40.7) und Spaltennummern („S_Bauteil“ 40.8 - 40.14) ausgegeben. Die Spalten- und Zeilennummer beziehen sich auf die Exceltabelle „Schubert06_input.xlsx“. Während der Ausführung des Algorithmus muss die .xslx-Datei geöffnet sein. Ansonsten kann der Algorithmus nicht auf die Information der Datei zugreifen. Die benötigte Exceltabelle ist im Anhang D6 enthalten. Des Weiteren werden in 40.15 die Öffnungswinkel der variablen Riegelprofile zurückgegeben.

Innerhalb des Clusters werden zunächst die „LeftAngle“ 29.1 und „RightAngle“ 29.2 nach Pfosten und Riegel getrennt. Dies wird vorgenommen, da der Algorithmus zur Auswahl einiger Komponenten speziell für diese beiden Anwendungsfälle angepasst werden muss. Wie Abbildung 42 verdeutlicht, erfolgt die Aufteilung mithilfe der „Split Tree“-Komponente. Da

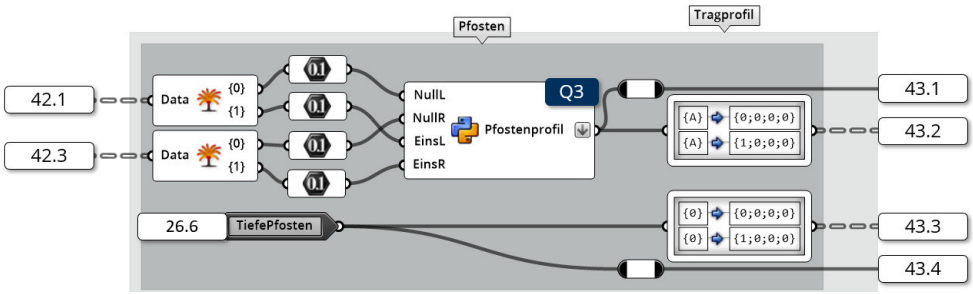


Abbildung 43: Aufteilen der Datenstruktur in Pfosten und Riegel
Quelle: In Anlehnung an Grasshopper

sich die Pfosten in den Pfaden {0} und {1} befinden und die Riegel in {2} und {3}, können mit der Maske {(0,1)} die beiden Pfostenprofile als „Positive“ und die Riegel als „Negative“ ausgegeben werden. Die Winkel der Pfosten werden in 42.1 beziehungsweise 42.3 weitergeleitet. Die Riegel sind dementsprechend in 42.2 und 42.4 enthalten.

Die Wahl der Tragprofile der Pfosten ist maßgeblich von den Winkeln zwischen Referenz- und Elementebene abhängig. Um den Winkelunterschied auszugleichen, können verschieden abgewinkelte Tragprofile gewählt werden. Die Abstufung der Tragprofile erfolgt dabei in 5 °-Schritten. Um die Tragprofile der Pfosten wählen zu können, werden die „LeftAngle“ 29.1 beziehungsweise „RightAngle“ 29.2 der Pfosten in die einzelnen Bauteile zerlegt. Dieser Vorgang ist in Abbildung 43 dargestellt. Dabei erhält man die Pfade {0} für das nach oben zeigende Profil und {1} für das nach unten zeigende. Mithilfe des selbst geschriebenen Skripts Q3 „Script: Select InnerProfile and InnerSealing“ können die benötigten Pfostenprofile bestimmt werden. Hierfür werden die linken und rechten Winkel der Profile {0} und {1} als „NullL“, „NullR“, „EinsL“ und „EinsR“ übergeben. Das Skript ist in der Anlage Q3 enthalten. Dieses überprüft zunächst ab Zeile 15 den Gültigkeitsbereich der Winkel. Sollte der Bereich verlassen werden, wird in Zeile 17 planmäßig ein Error generiert. Dieser Error wird nicht in der „ErrorMsgBox“, vgl. Kapitel 4.2, ausgegeben. Stattdessen wird durch „raise NameError(...)“ der Fehler direkt an der Komponente angezeigt. Ab Zeile 19 folgt die Ermittlung des kleinsten und größten Winkels aus allen Inputs. Dies ist nötig, da diese beiden Extremwerte für die Bestimmung der zulässigen Profile maßgeblich verantwortlich sind. Durch die Verwendung von speziellen Dichtungen gilt, dass der Maximalwert um bis zu 15 ° über- und der Minimalwert um bis zu - 5 ° unterschritten werden kann. Daher können die Maximal- und Minimalwinkel der Tragprofile über die Formeln

$$W_{max} = min + (5 - min \ mod \ 5) \tag{4.1}$$

$$W_{min} = max - (10 + max \ mod \ 5) \tag{4.2}$$

ermittelt werden. Der Maximalwert für beide Winkel ist dabei 25 °. In Zeile 46 bis 49 wird

überprüft, ob für die Randbedingungen ein geeignetes Profil gefunden werden kann. Dies ist der Fall, wenn W_{min} kleiner als W_{max} ist. Ansonsten weichen die Winkel zu sehr voneinander ab und der Unterschied kann nicht durch spezielle Dichtungen ausgeglichen werden. Sollte diese Bedingung nicht gelten, wird erneut ein Error erzeugt. Da nun ein maximal und ein minimal mögliches Profil bekannt sind, muss ein Profil aus dem Bereich [W_{min};W_{max}] gewählt werden. Sollte W_{min} = W_{max} gelten, so ist das Profil „p“ entweder W_{min} oder W_{max}. Ansonsten wird mittels des in Zeile 54 beginnenden Algorithmus ein Profil gewählt, das am geringsten von min und max abweicht. Zum Abschluss des Skripts wird der berechnete Profilwinkel „p“ in eine Spaltennummer umgerechnet. Dies erfolgt in den Zeile 62 und 63. Da die Pfostenprofile durchgängig im Fassadenknoten verbaut werden sollen, wird für beide Pfosten das gleiche Profil gewählt. Daher wird, wie auch in Abbildung 44 ersichtlich wird, die berechnete Spaltennummer sowohl in den Pfaden {0;0;0;0} als auch {1;0;0;0} gespeichert. Die Zeilennummer des zu wählenden Profils kann aus der gewählten Tiefe des Pfostens 26.6 entnommen und in den entsprechenden Speicherplätzen verschoben werden. Zum Aufbau der Pfadstruktur vgl. Kapitel 3.4.2.2.

Die Wahl der Riegelprofile ist nicht so komplex wie die der Pfosten. Dies liegt darin begründet, dass nur ein variables Riegelprofil existiert. Ein variables Riegelprofil ist in Abbildung 41 (d), sowie im Vergleich dazu ein standardmäßiges (c), gegeben. Die variable Variante ist für einen Winkel von 38 ° bis 90 ° ausgelegt. Im Rahmen dieser Arbeit wird ein Wertebereich von 0 ° bis 90 ° angenommen. Im Gegensatz zu den Pfostenprofilen sind keine variablen Dichtungen für Riegel vorhanden. Um Toleranzen eines realen Fassadenknotens darzustellen, wird für das Standardprofil mit Standarddichtungen ein Toleranzbereich von - 2 bis 2 ° angenommen. Die Begründungen für diese Annahmen finden sich in Kapitel 2.4.1.6. Die Winkelinformationen der Riegel werden in Abbildung 45 analog zu den Pfosten – vgl. Abbildung 43 – in die einzelnen Bauteile aufgeteilt. Da die angegebenen Winkelbereiche für beide Referenzelementebenenwinkel gelten, werden die Winkel für die beiden Bau-

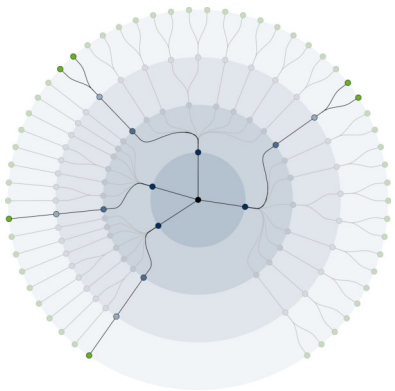


Abbildung 44: Datenstruktur der Tragprofile
Quelle: Eigene Darstellung

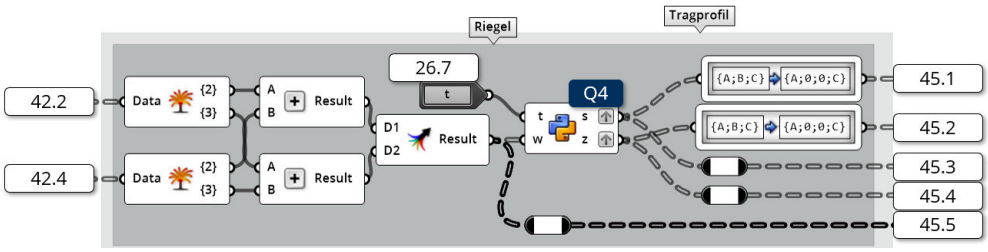


Abbildung 45: Aufteilen der Datenstruktur in Pfosten und Riegel
Quelle: In Anlehnung an Grasshopper

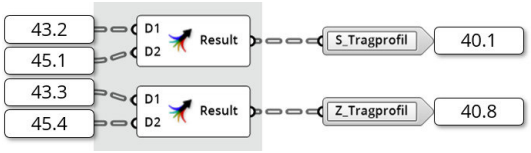


Abbildung 46: Ausgabe der Auswahl der Tragprofile
Quelle: In Anlehnung an Grasshopper

teile addiert und einerseits an das Skript Q4 „Script: Pick Riegelprofile“ gegeben. Das Skript ist in Anlage Q4 enthalten. Andererseits wird das Ergebnis der Addition in 45.5 weitergeleitet und als „ÖffnungswinkelVarRiegel“ 40.15 aus dem Cluster ausgegeben. In dem Skript wird in Zeile 14 unterschieden, ob ein variables Profil benötigt wird oder ob ein Standardprofil anwendbar ist. Sollte der Winkel zwischen - 2 und 2 ° liegen, kann das normale Profil gewählt werden. Hierbei gelten Zeilennummer = 1 und Spaltennummer = 1. Liegt der Winkel außerhalb dieses Bereichs, bestehen die Riegelprofile aus zwei miteinander verbunden Profilhälften. Darum werden in diesem Fall für Spalten- und Zeilennummern je eine Liste mit je zwei Werten ausgegeben. Alle variablen Profilkomponenten befinden sich in Spalte 2. Daher wird s = [2,2] ausgegeben. Da sich beide Profilhälften voneinander unterscheiden, werden zwei verschiedene Zeilennummern zurückgegeben. Dabei entspricht z = [1,2]. Auf das Skript folgend, werden zum einen die beiden Ergebnisse direkt in 45.3 und 45.4 weitergeleitet. Andererseits werden die Werte in 45.1 und 45.2 zuvor in den richtigen Pfaden gespeichert. Entsprechend der Regeln aus Kapitel 3.4.2.2 sind die Pfade {(2,3);0;0;C}, wobei C der eben erläuterten Bauteilhälfte entspricht. Diese Unterteilung der letzten Hierarchieebene ist in Abbildung 44 zu sehen. Durch diese Struktur werden die beiden Profilhälften in einem eigenen Datensatz abgelegt, sodass beide eindeutig unterschieden werden können. Im Falle des Standardriegelprofils entfallen die Pfade {(2,3);0;0;1}, sodass die Struktur mit der der Pfosten vergleichbar ist.

Bevor die Ergebnisse der Auswahl der Tragprofile aus dem Cluster ausgegeben werden, werden die Ergebnisse in der definierten Datenstruktur zusammengefasst. Dieser Vorgang ist in Abbildung 46 dargestellt. Die Spalten- und Zeilennummern werden als „S_Tragprofil “ 40.1 und „Z_Tragprofil“ 40.8 weiterverwendet.

4.4.2 Innere Dichtungen

Die Wahl der Dichtungen erfolgt auf Grundlage der Abweichungen der Elementebenen von den Referenzebenen und den gewählten Tragprofilen. Die Dichtungen für die Pfosten sind in der Lage bis zu - 5 ° in konkaver und + 15 ° in konvexer Richtung auszugleichen. Dabei sind nach dem Katalog von Schüco International KG (2011: S. 60, 61, 64) vier verschiedene Dichtungen vorhanden:

- - 5 ° bis 0 °
- 0 ° (Standarddichtung)
- 0 ° bis 10 °
- 10 ° bis 15 °

Um die Toleranzen der Realität zu berücksichtigen, wird der Gültigkeitsbereich der Standarddichtung für die Arbeit von 0 ° auf -1,25 ° bis 1,25 ° erhöht. Ansonsten ist ein genaues Treffen der Standarddichtung nahezu ausgeschlossen. Da die Riegelprofile in einer anderen Entwässerungsebene als die Pfostenprofile liegen, sind für sie keine variablen Dichtungen vorhanden. Der Ausgleich der Winkel erfolgt ausschließlich über die beiden Profilhälften der variablen Riegelprofile. In Abbildung 47 sind die verschiedenen Dichtungen dargestellt. Neben dem Winkel zwischen Element- und Referenzebene ist die Elementdicke ein weiterer wichtigen Faktor bei der Wahl der inneren Dichtungen. Dies hat den Hintergrund, dass verschiedene Elementdicken innerhalb einer Fassade ausgeglichen werden können und zusätzlich die Anzahl der verschiedenen Isolatoren innerhalb des Fassadensystems reduziert werden kann. Zusammen mit dem Isolator, können so im Bereich zwischen 24 mm bis 62 mm alle geraden Werte erzielt werden. Die meisten Isolatoren weisen dabei einen Abstand von 6 mm zum Nächstgrößeren auf. Im Umkehrschluss sind jeweils drei verschiedene Dichtungsgrößen vorhanden, damit alle geraden Werte erreicht werden können. Sollten die Elementdicken stärker als 6 mm variieren, gibt es spezielle Ausgleichsprofile, die zwischen Dichtung und Tragprofil positioniert werden.

(a) - 1,25 ° – 1,25 ° (b) - 5 ° – 0 ° (c) 0 ° – 10 ° (d) 10 ° – 15 ° (e) Riegel

Abbildung 47: Wählbare Dichtungen für Pfosten (a) bis (d) und Riegel (e)
Quelle: Screenshot aus Rhinoceros 6

Die Umsetzung der Auswahl der Dichtungen in Grasshopper ist in Abbildung 48 und 50 dargestellt. Dabei beschäftigt sich Abbildung 48 mit den Spaltennummern der Dichtungen. Für die Pfosten erfolgt die Auswahl durch das Python-Skript Q5 „Script: SelectInnerSealing“. Das Skript ist in Anlage Q5 in dieser Arbeit enthalten. Dieses benötigt als Eingangsparameter das gewählte Pfostenprofil aus 43.1 und die Winkel der Pfosten aus 42.1 und 42.3. Im Skript wird zunächst die Spaltennummer des Profils in den zugehörigen Winkel zurückgerechnet. Dies erfolgt in Zeile 15 bis 16. Anschließend werden die Abweichungen des gewählten Profils von den real vorhanden Winkeln bestimmt und anschließend ab Zeile 22 eine dazu passende Dichtung gewählt. Diese werden in die bauteilorientierte Datenstruktur gebracht. Da es sich um die inneren Dichtungen der Pfosten handelt, sind, wie in Abbildung 49 ersichtlich, die Pfade {(0,1);1;0;0} und {(0,1);1;1;0} dafür vorgesehen. In der Abbildung ist für die beiden Pfostenprofile eine Abzweigung in der dritten Hierarchieebene erkennbar. Diese Unterscheidung in der Hierarchieebene ist darin begründet, dass je Profil zwei innere Abdichtungen benötigt werden. Es ist vorteilhaft je Dichtung ein Datensatz zu erzeugen, da aufgrund besonderer Geometriestellungen diese beiden Dichtungen unterschiedlich sein können.

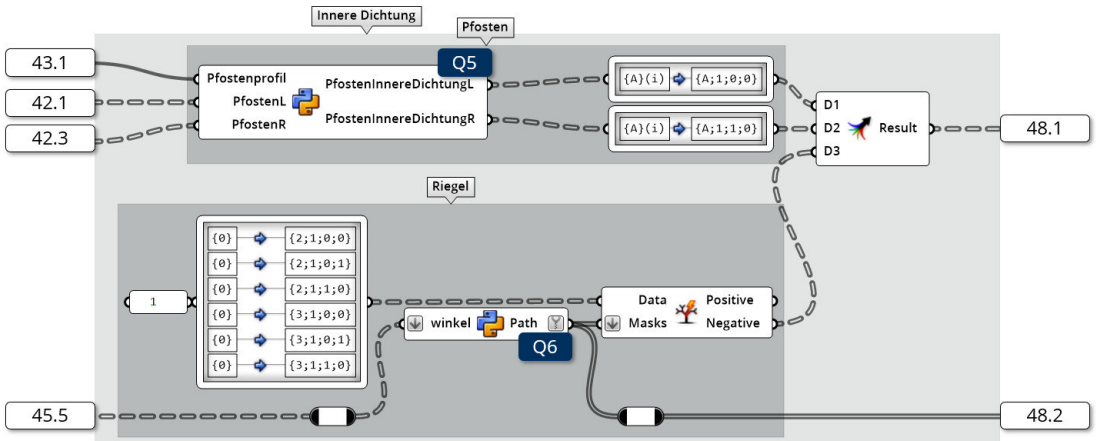


Abbildung 48: Spaltenauswahl der inneren Dichtungen
Quelle: In Anlehnung an Grasshopper

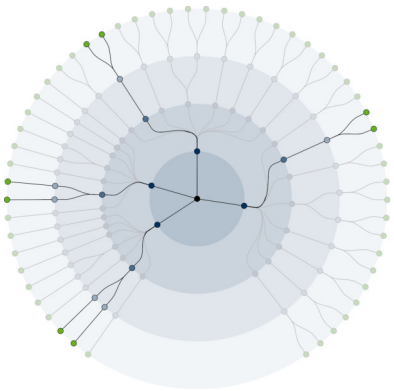


Abbildung 49: Datenstruktur der inneren Dichtungen
Quelle: Eigene Darstellung

Im Vergleich zu den Pfosten verläuft die Wahl der Spalten der Riegel sehr einfach, weil hier nur eine Dichtung vorhanden ist. Dadurch wird immer Spalte 1 gewählt. Jedoch ist hier die Schwierigkeit, eine geeignete Datenstruktur aufzubauen. Das ist darin zu begründen, dass die inneren Dichtungen immer aus zwei Stücken bestehen, diese aber nicht immer denselben Bauteilen zuzuordnen sind. Die Dichtungen sind bei Standardriegelprofilen am selben Profil befestigt. Wenn die Profile aber als zweiteilige variable Varianten ausgeführt werden, wird je eine Dichtung an eine Bauteilhälfte montiert. Dabei ist zu beachten, dass bei der Verblendung zwischen den beiden Riegelprofilhälften keine weiteren inneren Dichtungen verbaut werden. Davon sind die äußeren Dichtungen nicht betroffen. Um dieses Problem in Grasshopper zu lösen, wurden zunächst alle möglichen Datenstrukturpfade für innere Dichtungen erstellt und mit einer 1 belegt. Dazu wurde der in Abbildung 50 zu sehende „Path Mapper“ in der Gruppe „Riegel“, verwendet. Anschließend sollen die Pfade mithilfe der „Split Tree“-Komponente auf den jeweiligen Anwendungsfall angepasst werden. Die Datenstruktur der inneren Dichtungen ist in Abbildung 49 verdeutlicht. Hierbei gilt, dass die Pfade {(2,3);1;0;0} immer vorhanden sind. {(2,3);1;1;0} werden bei den Standardprofilen und {(2,3);1;0;1} bei den variablen Profilen benötigt. Dies bedeutet, dass der in der Abbildung dargestellte Fall variable Riegelprofile beinhaltet, da die vierte Hierarchieebene die beiden Datensätze unterscheidet. Für Standardprofile ist diese Abzweigung bereits in der dritten Ebene vorhanden. Um die „Split Tree“-Komponente mit der entsprechenden Maske zu versorgen, wird das Skript Q6 „Script: PfadmaskenInnereDichtungRiegel“ verwendet. Das Skript ist in Anlage Q6 zu finden. Als Input benötigt das Skript Informationen über die Winkel der Riegel. Diese werden aus 45.5 entnommen. Um eine geeignete Maske generieren zu können, müssen zunächst die vorliegenden Winkel analysiert werden. Dies geschieht mit der if-Anweisung in Zeile 16. Sollte der Fall eintreten, dass das Standardprofil geeignet ist, so wird die Maske {(2,3);1;0;1} ausgegeben. Anderenfalls {(2,3);1;1;0}. Das

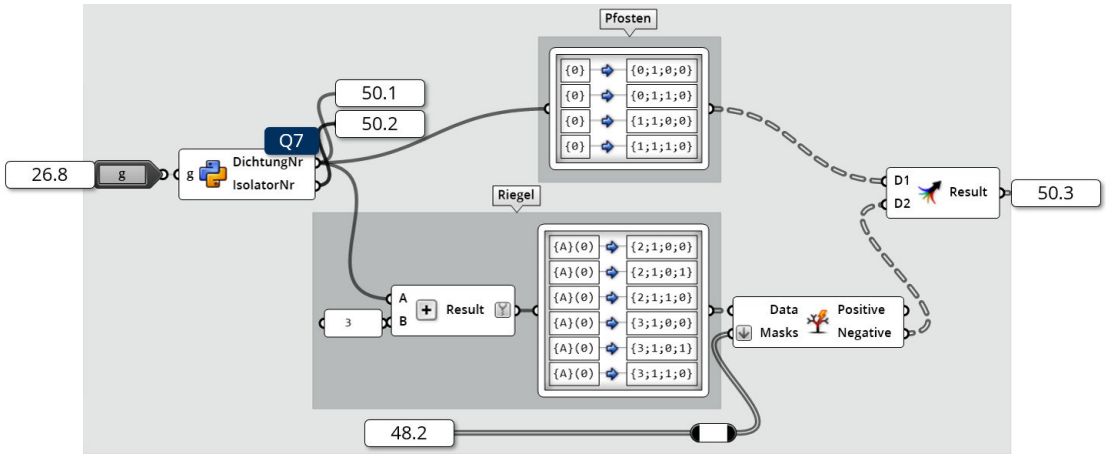


Abbildung 50: Zeilenauswahl der inneren Dichtungen
Quelle: In Anlehnung an Grasshopper

zusammengefasste Ergebnis der Spalten 48.1 wird als „S_DichtungInnen“ in 40.2 ausgegeben. Die Maske 48.2 wird für die Zeilennummern der inneren Dichtungen weitergeleitet.

Um die Zeilennummern der inneren Dichtungen bestimmen zu können, wird die Angabe über die Dicke der Fassadenelemente benötigt. Daher wird die Variable „g“ aus „Elementdicke“ 26.8 in das Cluster geladen. Diese Variable wird dort an das Python-Skript Q7 „Script: Script: InnereDichtungen und Isolator“ übergeben. Das Skript ist in Anhang Q7 gegeben. Dieses übersetzt die „Elementdicke“, die nicht als tatsächliche Dicke, sondern als Zählvariable (1,2,3,...) vorliegt, in je eine Zeilennummer für die inneren Dichtungen und den Isolator. Dabei gilt folgende Logik:

$$DichtungNr = \begin{cases} 1 & \text{für g=0} \ (\equiv 24 \text{ mm}) \vee 2 \ (\equiv 28 \text{ mm}) \vee 5 \ (\equiv 34 \text{ mm}) \\ 2 & \text{für g=1} \ (\equiv 26 \text{ mm}) \vee 3 \ (\equiv 30 \text{ mm}) \vee 6 \ (\equiv 36 \text{ mm}) \\ 3 & \text{für g=4} \ (\equiv 32 \text{ mm}) \vee 7 \ (\equiv 38 \text{ mm}) \end{cases} \tag{4.3}$$

$$IsolatorNr = \begin{cases} 1 & \text{für g=0} \ (\equiv 24 \text{ mm}) \vee 1 \ (\equiv 26 \text{ mm}) \\ \frac{g+1}{3} + 1 & \text{sonst} \end{cases} \tag{4.4}$$

Die Dicken in der Logik sind von den vorhandenen Dichtungen und Isolatoren abgeleitet. Diese können aus den Fertigungsunterlagen von Schüco International KG (2013) entnommen werden. Die so ermittelten Werte, werden einerseits in 50.1 und 50.2 weitergeleitet. Andererseits wird die „DichtungsNr“ für Riegel und Pfosten getrennt betrachtet. Dies hat den Hintergrund, dass die Pfosten und die Riegel unterschiedliche Dichtungen erhalten. Dabei stehen die Dichtungen für Riegelprofile drei Zeilen unter denen der Pfosten. Somit ergibt sich, dass die Zeilennummer der Riegeldichtungen mit 3 addiert werden muss. Anschließend werden sowohl die Riegel-, als auch die Pfostendichtungen in die bauteilorientierte Datenstruktur übertragen. Für die Riegel wird erneut ein Filtervorgang mit der zuvor erzeugten Maske 48.2 vorgenommen, um die Datenstruktur an die variablen oder normalen Profile anzugleichen. Die Zeilennummern werden mit der „Merge“-Komponente zusammengefasst, bevor sie als 50.3 zum Clusterausgang als „Z_DichtungInnen“ 40.9 geleitet werden.

4.4.3 Isolator

Die Wahl des geeigneten Isolators erfolgt auf Grundlage der Elementdicke. In Abbildung 51 sind verschiedene Isolatoren gegeben. Es gilt weiterhin zu beachten, dass bei normalen Profilen der Isolator einmal vorkommt. Sollte jedoch der Einsatz von variablen Riegelprofilen notwendig sein, erhält jede Profilhälfte einen eigenen Isolator. Wie bereits bei den inneren Dichtungen, muss in diesem Fall die Datenstruktur angepasst werden. Diese Anpassung ist in Abbildung 52 gezeigt. Darin ist zu sehen, wie die Pfosten nur einen Datensatz aufweisen und bei den Riegeln in der vierten Hierarchieebene zwei Datensätze angelegt sind. Dies ist darin begründet, dass die gezeigte Datenstruktur variable Riegelprofile enthält.

Zur Umsetzung in Grasshopper werden, wie in Abbildung 53 verdeutlicht, die Winkel der Riegel 45.5 und die bereits in Q7 ermittelten „IsolatorNr“ 50.2 benötigt. Diese beiden

(a) (b)

Abbildung 51: Beispiele von wählbaren Isolatoren
Quelle: Screenshot aus Rhinoceros 6

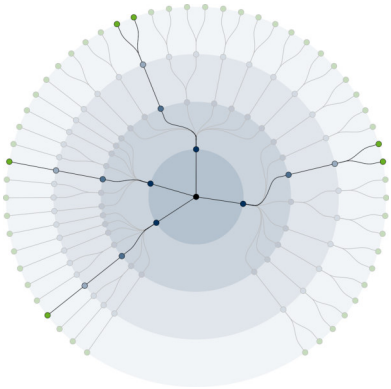


Abbildung 52: Datenstruktur der Isolatoren
Quelle: Eigene Darstellung

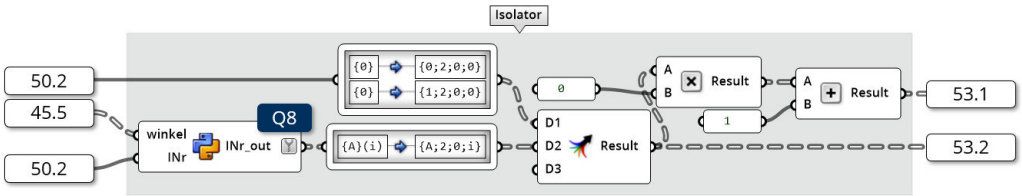


Abbildung 53: Auswahl des Isolators
Quelle: In Anlehnung an Grasshopper

Werte werden als Input für das Python-Skript Q8: „Script: Datenstruktur Isolator“ verwendet. Das Skript ist in Anhang Q8 enthalten. In diesem wird zunächst in Zeile 16 mit einer if-Anweisung überprüft, ob ein variables Profil oder ein Standardprofil verwendet wird. Sollte ein Standardprofil verbaut werden, wird die „INr“, also die Zeilennummer des Isolators an „INr_out“ weitergegeben. Anderenfalls wird eine Liste mit zwei Zeilennummern „[INr,INr]“ ausgegeben, damit die Datenstruktur automatisch mithilfe der auf das Skript folgenden „Path Mapper“-Komponente angepasst werden kann. Hierfür wird der Index als Pfad für die Bauteilhälfte verwendet. Für die Pfostenprofile ist die Wahl der Isolatoren einfacher. Da die Zeilennummer des Isolators bereits gewählt wurde, muss die Nummer noch in die bauteilorienterte Datenstruktur übertragen werden. Nach Abbildung 52 gelten für die Isolatoren die Pfade {(0,1);2;0;0} und {(2,3);2;0;(0,1)}. Die Zeilennummern der Pfosten und Riegel werden mit der „Merge“-Komponente zusammengefasst und zum einen in 53.2 als „Z_Isolator“ 40.10 aus dem Cluster ausgegeben. Zum anderen kann die für die Spaltennummern der Isolatoren aufgebaute Datenstruktur erneut verwendet werden. Da bei den Isolatoren keine weiteren Entscheidungskriterien vorhanden sind, stehen alle Isolatoren in Spalte 1. Daher wird die Datenstruktur zunächst mit 0 multipliziert, damit alle Einträge den Wert 0 aufweisen. Anschließend wird der Wert 1 addiert und als „S_Isolator“ 40.3 aus dem Cluster ausgegeben.

4.4.4 Äußere Dichtungen

Die Auswahl der äußeren Dichtung ist weniger komplex als die Wahl der Inneren. Dies ist darin begründet, dass die äußeren Dichtungen keiner Anpassung an die Elementdicke unterliegen. Dadurch spielt bei der Wahl ausschließlich die gewählte Pressleiste und der Winkel zwischen Referenz- und Elementebene eine maßgebende Rolle. In Abbildung 54 sind einige wählbare Dichtungen dargestellt. Hierbei gilt zu beachten, dass für die variablen Riegelprofile nur das Profil (a) verwendet wird, da die Profile auf den exakten Winkel eingestellt werden können.

In Abbildung 55 ist die Umsetzung in Grasshopper dargestellt. Die Auswahl der Dichtungen erfolgt getrennt nach Pfosten und Riegel, da für die Pfosten variable Dichtungen vorhanden sind, um die Winkel zwischen den in Kapitel 4.3.1 eingeführten Referenz- und den in Kapi-

(a) - 1,25 ° – 1,25 ° (b) - 5 ° – 0 ° (c) 0 ° – 10 °

Abbildung 54: Wählbare äußere Dichtungen
Quelle: Screenshot aus Rhinoceros 6

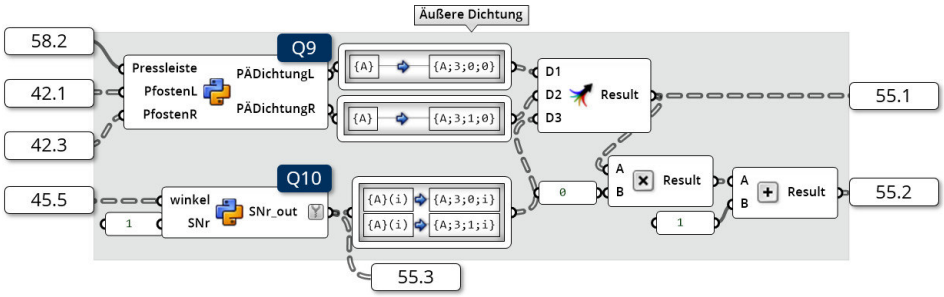


Abbildung 55: Auswahl der äußeren Dichtungen
Quelle: In Anlehnung an Grasshopper

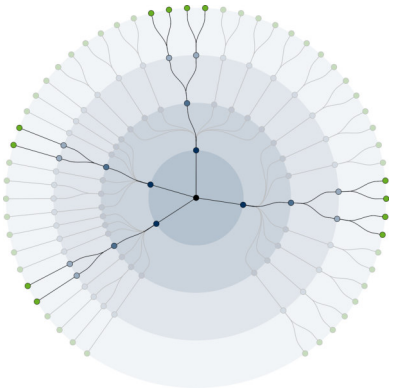


Abbildung 56: Datenstruktur der äußeren Dichtungen
Quelle: Eigene Darstellung

tel 4.3.4 eingeführten Elementebene auszugleichen. Andererseits existieren für die Riegel variable Profile, die eine genaue Einstellung an die Winkel ermöglichen und somit keine ausgleichenden Dichtungen verwendet werden. Da für die Pfosten die Wahl der Dichtungen von der Pressleiste abhängt, wird die Zeilennummer der Pressleiste aus 58.2 verwendet. Zusätzlich zum gewählten Pressleistenprofil werden die Winkel zwischen der Referenz- und den beiden Elementebenen benötigt. Diese können aus 42.1 und 42.3 entnommen werden. Die genannten Parameter werden im Python-Skript Q9 „Script: Select OuterSealing“ zu geeigneten Dichtungen verarbeitet. Das Skript ist in Anlage Q9 gegeben. Im Skript wird zunächst die Nummer der Pressleiste in den dazugehörigen Winkel umgewandelt. Dies wird mit folgender Logik (vgl. ab Zeile 17) erzielt:

$$p = \begin{cases} 0 & \text{für Pressleiste} = 1 \\ 3.75 & \text{für Pressleiste} = 2 \\ 7.5 & \text{für Pressleiste} = 3 \\ 15 & \text{für Pressleiste} = 4 \\ 20 & \text{für Pressleiste} = 5 \\ 30 & \text{für Pressleiste} = 6 \\ 35 & \text{für Pressleiste} = 7 \end{cases} \tag{4.5}$$

Anschließend kann die Differenz des Winkels der Pressleiste zum realen Winkel gebildet werden. Dieser Vorgang findet ab Zeile 33 statt. Danach kann eine Dichtung zum Ausgleich dieser Differenz gewählt werden. Dies erfolgt mit der ab Zeile 36 implementierten Logik. Eine äquivalente Logik ist folgend gegeben:

$$PDichtungL = \begin{cases} 2 & \text{für -1.25} \geq \text{PfostenL} \\ 1 & \text{für -1.25} < \text{PfostenL} < 1.25 \\ 3 & \text{für } 1.25 \leq \text{PfostenL} < 10 \\ 4 & \text{für } 10 \leq \text{PfostenL} \leq 15 \end{cases} \tag{4.6}$$

$$PDichtungR = \begin{cases} 2 & \text{für -1.25} \geq \text{PfostenR} \\ 1 & \text{für -1.25} < \text{PfostenR} < 1.25 \\ 3 & \text{für } 1.25 \leq \text{PfostenR} < 10 \\ 4 & \text{für } 10 \leq \text{PfostenR} \leq 15 \end{cases} \tag{4.7}$$

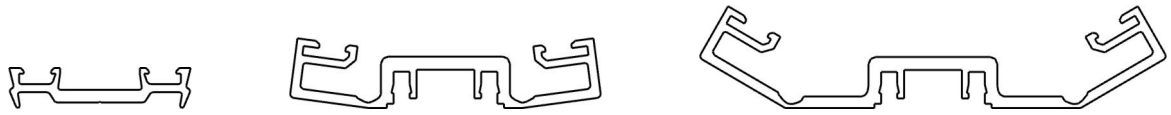
Die Ergebnisse werden aus dem Skript zurückgegeben und mit der „Path Mapper“-Komponente an die richtige Position innerhalb der bauteilorientierten Datenstruktur verschoben. Wie aus Abbildung 56 ersichtlich ist, sind für die äußeren Dichtungen der Pfosten die Pfade {(0,1);3;(0,1);0} vorgesehen. Die Wahl der äußeren Dichtungen der Riegelprofile erfolgt analog zu den Pfosten. Dabei werden die Winkel der Riegelprofile aus 46.5 benötigt. Diese werden in das Python-Skript Q10 „Script: Riegel Datenstruktur äußere Dichtung“ gegeben. Die Funktionsweise dieses Skripts ist analog zu Q8. Daher wird zusätzlich der Wert 1 als „SNr“ an das Skript übergeben. Da dieser Wert fest steht, kann er direkt im Skript implementiert werden. Durch das Skript wird die Spaltennummer (=1) als Liste ausgegeben. Für Standardprofile wird der Wert einmal und für die variable Variante zweimal zurückgegeben. Mithilfe des „Path Mappers“ wird die Datenstruktur angepasst. Die Pfade werden dadurch zu {(2,3);3;(0,1);(0,1)}. Wie in Abbildung 56 zu sehen ist, werden pro variablem Riegelprofil vier Datensätze erzeugt. Anschließend werden die Daten mit den Ergebnissen der Pfosten zusammengeführt und einerseits in 55.1 als „S_DichtungAußen“ 40.4 ausgegeben. Andererseits wird, wie bereits beim Isolator, die Datenstruktur zur Bildung der Zeilennummern genutzt. Hierbei erfolgt erneut die Multiplikation mit 0 und die darauffolgende Addition von 1. Nachdem die Zeilennummern gebildet wurden, werden sie in 55.2 als „Z_DichtungAußen“ 40.11 aus dem Cluster ausgegeben.

4.4.5 Pressleiste

Für die Wahl der Pressleisten sind die Winkel zwischen Referenz- und Elementebenen maßgeblich verantwortlich. Dem Pfosten stehen mehrere verschieden abgewinkelte Pressleisten zur Verfügung. Nach Schüco International KG (2011: S. 107) sind folgende Winkel vorhanden: 0 °, 7.5 °, 15 °, 20 °, 30 °, 35 °. Hier gilt zu beachten, dass zwei verschiedene Profile mit dem Winkel 0 ° vorhanden sind. Beide Profile unterscheiden sich in der Breite. Daher soll das schmalere Profil für kleinere und das Breitere für größere Winkel angewendet werden. In Abbildung 57 ist eine Auswahl verschiedener Pressleisten gegeben.

Wie bereits bei den variablen Tragprofilen der Pfosten werden die verschiedenen Profile mit den variablen Dichtungen ergänzt, sodass sich ein sehr flexibles System im Bereich von -5 ° bis 50 ° ergibt. Dabei gilt zu beachten, dass die Tragprofile einen maximalen Winkel von 40 ° zulassen. Bei den äußeren Dichtungen ist das gleiche System wie bei den inneren Dichtungen vorhanden. Diese umfassen den Bereich von -5 ° bis 15 °. Innerhalb dieses Bereichs kann die Elementebene von der Referenzebene abweichen.

In Grasshopper wird das Ganze, wie in Abbildung 58 ersichtlich, mit dem Python-Skript Q11 „Script: Select OuterProfile“ umgesetzt. Diesem Skript werden die einzelnen Winkel der



(a) 0° (b) 7,5 ° (c) 30 °

Abbildung 57: Wählbare Pressleisten (Auswahl)
Quelle: Screenshot aus Rhinoceros 6

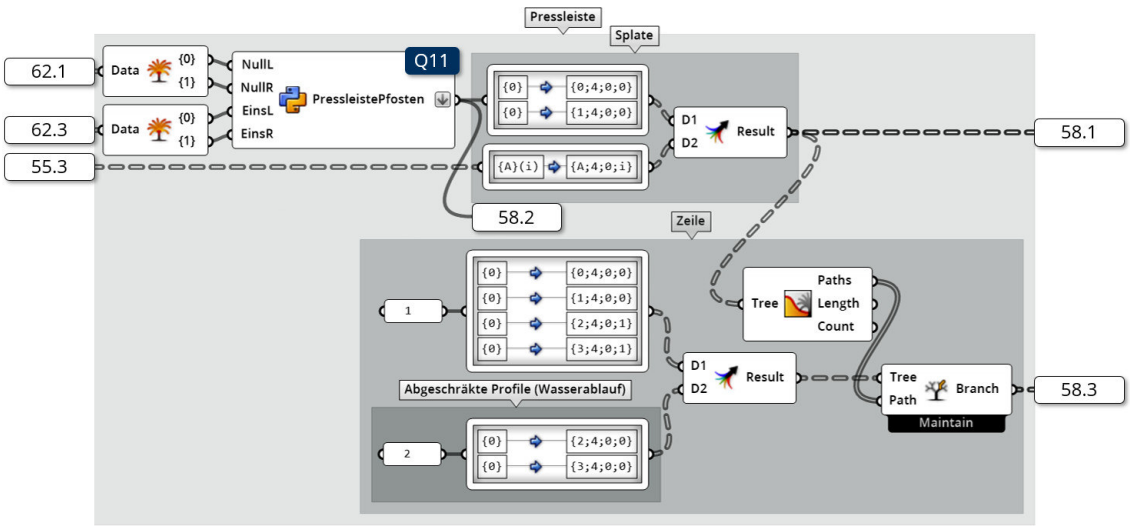


Abbildung 58: Auswahl der Pressleiste
Quelle: In Anlehnung an Grasshopper

Pfosten 42.1 und 42.3 übergeben. Diese werden zuvor durch die „Split Tree“ aufgeteilt. Das Skript wählt anschließend, auf Grundlage der gegebenen Bedingungen, die am besten geeignete Pressleiste aus. In dem in Anlage Q11 beigelegte Skript wird zunächst in Zeile 15 ff. der Gültigkeitsbereich überprüft und wenn nötig eine Fehlermeldung ausgegeben. Da aufgrund der float-Division in Python Rundungsfehler auftreten können, werden zunächst die Winkel um einen kleinen Betrag erhöht. Dies ist darin begründet, dass anderenfalls bei der Übergabe eines Integers mit dem Wert 5 gilt: 5 mod 5 = 1. Ab Zeile 25 werden die Extremwerte aus den Eingangswinkeln (min und max) herausgefiltert, um anschließend ab Zeile 40 die Profile herauszufinden, die in den Grenzfällen möglich sind. Dabei wird folgende Logik befolgt:

$$W_{max} = \begin{cases} 0 & \text{für } \min < 2.5 \\ 7.5 & \text{für } 2.5 \leq \min = 10 \\ 15 & \text{für } 10 \leq \min = 15 \\ 20 & \text{für } 15 \leq \min = 25 \\ 30 & \text{für } 25 \leq \min = 30 \\ 35 & \text{für } 30 \leq \min = 45 \end{cases} \tag{4.8}$$

Daraus ergibt sich ein Profil mit einem maximalen Winkel W_{max}. Es kann kein Profil gewählt werden, das stärker abgewinkelt ist, da sonst keine geeignete Abdichtung vorhanden ist. Analog zu Formel 4.8 kann auch W_{min}, also der kleinste zulässige Profilwinkel, ermittelt werden. Hierbei ist zu beachten, dass andere Randbedingungen zu wählen sind und dass der logische Aufbau der Abfrage (vgl. ab Zeile 63) entgegengesetzt strukturiert ist. Anschließend werden W_{max} und W_{min} analysiert. Sollte $W_{max} < W_{min}$ gelten, dann liegen die beiden Extremwerte der Winkel zu sehr voneinander entfernt, sodass keine Dichtung gefunden werden kann, die diese Differenz ausgleicht. Ansonsten wird ab Zeile 81 ein optimaler Profilwinkel ausgewählt. Der Algorithmus dahinter vergleicht dabei den Mittelwert der beiden Extremwerte mit den vorhandenen und geeigneten Profilen. Zu diesem Zeitpunkt liegt das ausgewählte Profil in Grad vor. Da das Profil aus einer Exceltabelle ausgewählt wird, wird abschließend der Profilwinkel in eine Spaltennummer des Profils umgewandelt. Dieser Vorgang findet ab Zeile 104 statt. Nachdem das Python-Skript die Spaltennummer der Pressleiste zurückgegeben hat, werden die Daten einerseits als 58.2 zur Ermittlung der äußeren Dichtungen genutzt. Andererseits werden sie in die bauteilorientierte Datenstruktur umgewandelt. Wie in Abbildung 59 dargestellt, sind für die Pressleisten der Pfostenprofile die Pfade {(0,1);4;0;0} vorgesehen. Die Wahl der Pressleisten der Riegel kann über die Spaltennummern der Außendichtungen der Riegel 55.3 erfolgen. Dies ist darin begründet, dass die Pressleisten der Riegel immer in Spalte 1 liegen und die Datenstruktur analog zu der Datenstruktur der äußeren Dichtungen angepasst werden muss. Das Skript

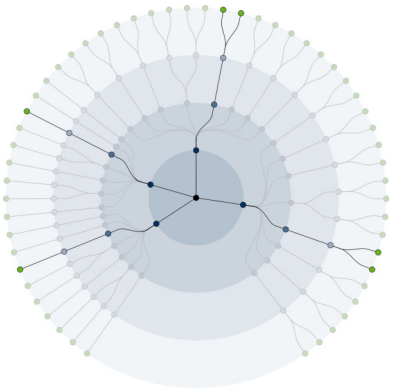


Abbildung 59: Datenstruktur der Pressleisten
Quelle: Eigene Darstellung

Q10 übermittelt einen einzelnen Wert, sollte es sich um ein Standardprofil handeln. Im Fall der variablen Variante, wird eine Liste mit zwei Werten zurückgegeben. Mit der „Path Mapper“-Komponente wird die Datenstruktur für die Pressleisten der Riegel auf die Pfade {(2,3);4;0;(0,1)} angepasst. Abbildung 59 verdeutlicht die Datenstruktur der Pressleisten. Es ist zu erkennen, dass jedem Riegel zwei Datensätze zugeordnet werden, da es sich um variable Riegel handelt. Im Fall von Standardprofilen entfallen die Pfade {(2,3);4;0;1}. Zusammen mit den Spaltennummern der Pfosten wird das Ergebnis in 58.1 zum einen für die Ermittlung der Blendleisten weitergenutzt und zum anderen als {S_Pressleiste} 40.5 ausgegeben. Die Zeilennummern der Pressleisten ergeben ein statisches Verhalten. Dabei sind alle Pfostenpressleisten in Spalte 1 geschrieben. Die Riegel sind für Bauteilhälfte 1 immer 1 und für Bauteilhälfte 0 immer 2. Die Pressleiste in Spalte 2 ist abgewinkelt, vgl. Abbildung 103 (zu finden auf Seite 105), sodass Wasser besser ablaufen kann. Diese Werte werden mit der „Path Mapper“-Komponente in allen theoretisch möglichen Datenstrukturplätzen gespeichert und anschließend mit der „Tree Branch“-Komponente gefiltert. Diese Komponente erhält als Filter die Pfade aus der fertigen Datenstruktur der Spaltennummern. Eine Anpassung der Datenstruktur erfolgt so automatisch. Die gefilterten Zeilennummern werden in 58.3 einerseits als „Z_Pressleiste“ 29.12 aus dem Cluster ausgegeben und andererseits für die Ermittlung der Blendleisten weitergenutzt.

4.4.6 Blendleisten

Die Wahl der Blendleisten beschränkt sich auf die drei in Abbildung 60 gegebenen Varianten. Zur Verfügung steht eine Standardblendleiste (a), eine abgewinkelte Blendleiste (b) und eine Blendleiste für abgewinkelte Pressleisten (c). Für die Ermittlung der Blendleisten können die Informationen der Pressleiste genutzt werden. Diese Informationen müssen dabei nur geringfügig manipuliert werden. Wie in Abbildung 61 dargestellt ist, müssen für die Zeileninformationen nur die Zeilennummern der Pressleisten 58.3 mithilfe des „Path Mappers“ innerhalb der bauteilorientierten Datenstruktur verschoben werden. Dazu werden die Pfade {(0,1);5;0;0} sowie {(2,3);5;0;(0,1)} verwendet. Somit ist die in Abbildung 62 dargestellte Datenstruktur mit der Datenstruktur der Pressleisten vergleichbar. Für Spaltennummern werden die Spaltennummern der Pressleisten 58.1 an ein Python-Skript übergeben. Das Skript Q12 „Script: Blendleiste“ ist in Anhang Q12 beigefügt. In dem Skript werden die Spaltennummern der Pressleiste in die entsprechenden Spaltennummern der Blendleisten umgewandelt. Dabei erhalten alle abgewinkelten Pressleisten (ab Spalte 2) dieselbe Blendleiste (Spalte 2). Die Konvertierung findet hauptsächlich ab Zeile 12 statt. Die Ergebnisse der Blendleiste werden in 61.1 als „Z_Blendleiste“ 40.13 und 61.2 als „S_Blendleiste“ 40.6 aus dem Cluster ausgegeben.

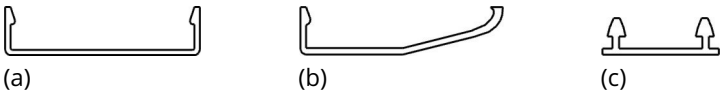


Abbildung 60: Wählbare Blendleisten
Quelle: Screenshot aus Rhinoceros 6

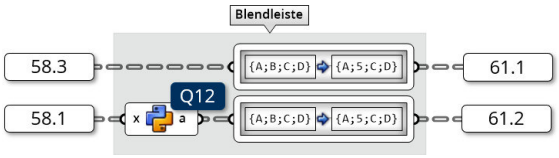


Abbildung 61: Auswahl der Blendleiste
Quelle: In Anlehnung an Grasshopper

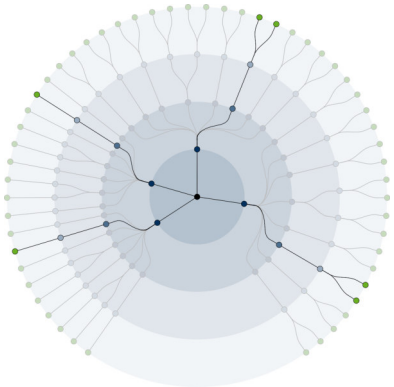


Abbildung 62: Datenstruktur der Blendleisten
Quelle: Eigene Darstellung

4.4.7 Einschubprofile

Zuletzt erfolgt die Ermittlung der Einschubprofile. In Abbildung 63 sind einige Einschubprofile gegeben. Für Pfosten (a) sind die Einschubprofile ausschließlich von der gewählten Pfostentiefe abhängig. Bei Riegelprofilen erfolgt eine weitere Unterscheidung in standardisierte (b) und variable (c) Riegelprofile. Dabei ist zu beachten, dass für Riegel keine Einschubprofile zur Verfügung stehen und es sich bei den gezeigten Profilen um Annahmen des Autors handeln, um ein vollständiges System zu generieren. Die Umsetzung ist in Abbildung 64 gegeben. Da die Einschubprofile nur von den gewählten Tragprofilen abhängig sind, können die Spalten- und Zeilennummern übernommen werden. Somit können aus 45.3 und 45.4 die Zeilen- und Spaltennummern der gewählten Tragprofile der Riegel wiederverwendet werden. Für die Pfosten ist der Vorgang etwas anders, da im Gegensatz zu den Tragprofilen die Einschubprofile nicht vom Winkel zwischen Referenz- und Elementebene abhängig sind. Daher ist die Spaltennummer der Pfosteneinschubprofile immer 1 und die Zeilennummer wird von der Tiefe der Pfosten 26.6 abgeleitet. Die Daten werden abschließend in ihrer Datenstruktur angepasst. Wie in Abbildung 65 dargestellt ist, sind die Pfade {(0,1);6;0;0} sowie {(2,3);6;0;(0,1)} für die Einschubprofile vorgesehen. Hier ist zu erwähnen, dass bei Standardriegelprofilen die Pfade {(2,3);6;0;1} entfallen. Abschließend wird 64.1 als „S_Einschubprofil“ 40.7 und 64.2 als „Z_Einschubprofil“ 40.15 ausgegeben.

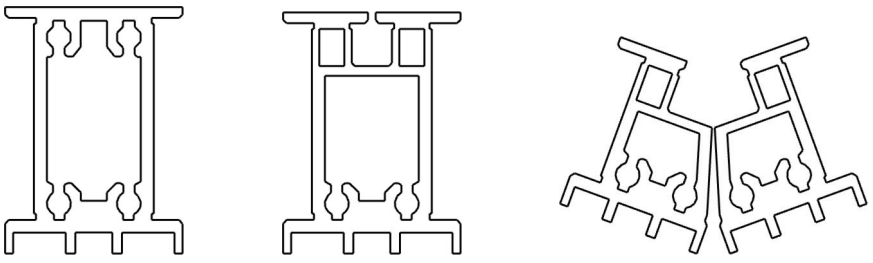


Abbildung 63: Wählbare Einschubprofile
Quelle: Screenshot aus Rhinoceros 6

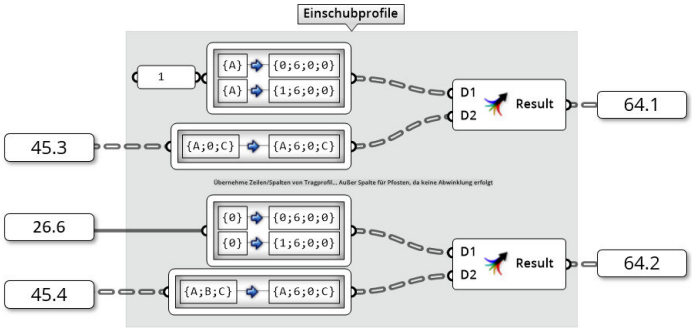


Abbildung 64: Auswahl der Einschubprofile
Quelle: In Anlehnung an Grasshopper

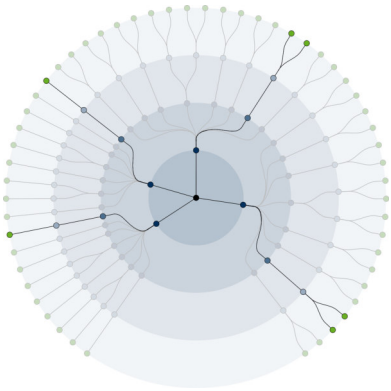


Abbildung 65: Datenstruktur der Einschubprofile
Quelle: Eigene Darstellung

4.5 Laden der Komponentengeometrie

Nachdem in Kapitel 4.4 die Auswahl der einzelnen Geometrien erläutert wurde, kann in Kapitel 4.5 damit begonnen werden, die Geometrien für die einzelnen Bauteilkomponenten zu laden. Zu diesem Zweck wurden CAD-Daten der Komponenten in das proprietäre Dateiformat .3dm umgewandelt. Die Dateien wurden unter der Benennung „Artikelnummer.3dm“ abgespeichert. Diese sind in Anhang D1 im Ordner „Schubert01_CAD“ beigelegt. So können über die in den Exceltabellen enthaltenen Artikelnummern die Geometrien geladen werden. Die Artikelnummern sind dabei an die Kataloge von Schüco International KG (2011) und (2013) angelehnt. Die Exceltabelle wurde vom Autor dieser Arbeit erstellt und ist in Anhang D6 gegeben.

Das „Cluster: Komponenten Picker“, das in Abbildung 66 gegeben ist, übernimmt diese Aufgabe. Dazu benötigt das Cluster zum einen alle Zeilen- 40.8 - 40.15 und Spaltennummern 40.1 - 40.7, die im „Cluster: KomponentenSelector“ generiert wurden und zum anderen

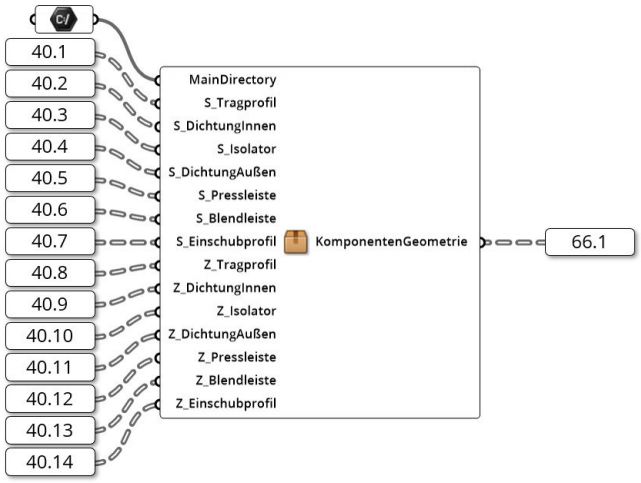


Abbildung 66: Das „Cluster: Komponenten Picker“
Quelle: In Anlehnung an Grasshopper

einen Dateipfad zum Verzeichnis der CAD-Daten auf dem Computer. Dabei ist der im Anhang D1 befindliche Ordner zu nutzen. Der Ordner „Schubert01_CAD“ muss hierfür in der „File Path“-Komponente ausgewählt werden. Dazu muss die Option „Select a Directory“ per Rechtsklick auf die „File Path“-Komponente gewählt werden.

4.5.1 Excel Reader

Die Abbildung 67 zeigt, dass das „Cluster: Komponenten Picker“ aus zwei weiteren Clustern besteht. Dem ersten der beiden, in Abbildung 68 dargestelltem „Cluster: ExcelReader“ werden sämtliche Spalten- 40.1 - 40.7 und Zeilennummern 40.8 - 40.15 übergeben. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/03_ClusterExcelReaderKomplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Das Cluster soll daraufhin die Artikelnummern aus der Exceltabelle heraussuchen. Die so erzeugten Artikelnummern werden anschließend als 68.1 ausgegeben.

Im Cluster wird zunächst, wie in Abbildung 69 dargestellt, allen möglichen Datenstrukturpfaden eine Nummer zugeordnet. Dabei entspricht die Nummer einer Exceltabelle. Zum Beispiel werden in Tabelle 1 die Informationen der Tragprofile der Pfosten gespeichert. Hier ist zu beachten, dass das Tabellenblatt 7 zunächst übersprungen wird. Dies ist darin begründet, dass in diesem Blatt die Informationen der Tragprofile der Riegel hinterlegt sind, da sich die Riegelprofile von den Pfostenprofilen unterscheiden. Mithilfe des „Path Mappers“ kann die Inputliste in die bauteilorientierte Datenstruktur überführt werden.

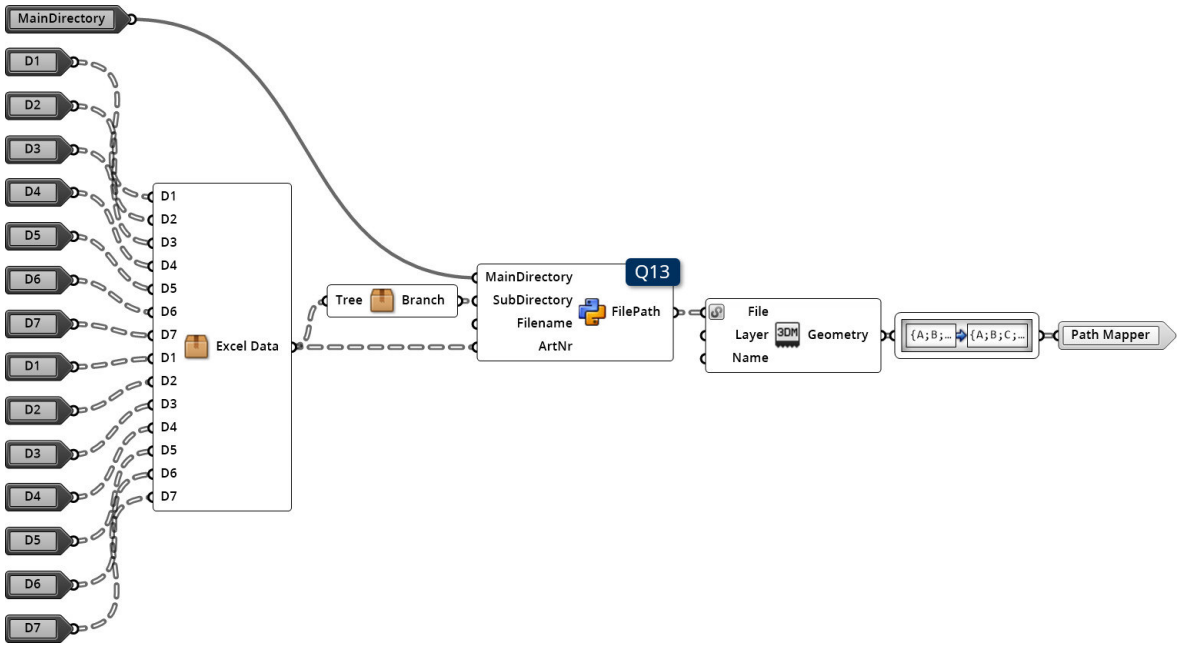


Abbildung 67: Das „Cluster: Komponenten Picker“
Quelle: In Anlehnung an Grasshopper

Da nicht immer alle Artikelnummern einer Bauteilkomponente innerhalb eines Tabellenblattes aufgeführt werden, sondern teilweise, wie bei den Tragprofilen, die Artikelnummern für Pfosten und Riegel getrennt abgespeichert sind, muss die angelegte Datenstruktur der Tabellenblätter 69.1 bearbeitet werden. Aus diesem Grund werden, wie in Abbildung 70 dargestellt, mithilfe der „Split Tree“-Komponente einzelne Teilmengen herausgefiltert und angepasst. Die dabei zu modifizierenden Komponenten sind zum einen die Tragprofile. Die Tragprofile der Riegel können mit der Maske {(2,3);0;(0,1);(0,1)} herausgefiltert werden. Anschließend wird die Tabellennummer um sechs auf sieben erhöht. Dasselbe Verfahren wird bei den Einschubprofilen angewendet. Hier kann mit der Maske {(2,3);6;(0,1);(0,1)} die zu modifizierende Teilmenge herausgefiltert und angepasst werden. Für die Einschubprofile der Riegel wird die Tabellennummer von acht auf neun

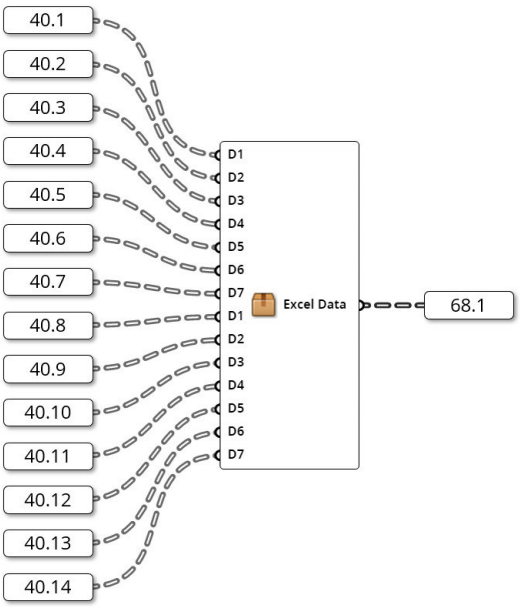


Abbildung 68: Das „Cluster: ExcelReader“
Quelle: In Anlehnung an Grasshopper

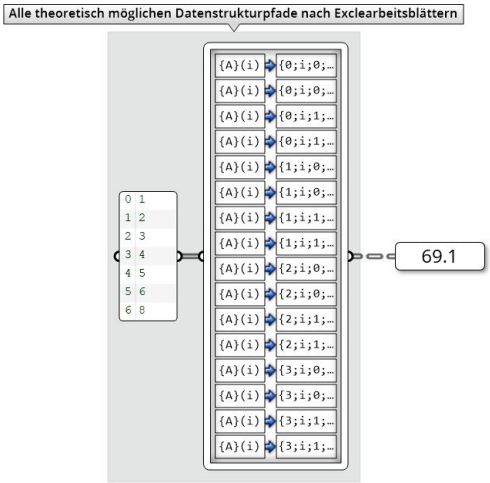


Abbildung 69: Zuordnung der Tabellenblätter
Quelle: In Anlehnung an Grasshopper

erhöht. Die Ergebnisse der Bearbeitung der Tabellennummern werden anschließend wieder zu einem Baum zusammengefasst. Dieser Filtervorgang ist in Abbildung 71 dargestellt. Die linksseitige Datenstruktur wird in die drei Teilmengen, die in der Mitte dargestellt sind, aufgeteilt. Die Teilmengen werden modifiziert und wieder zusammengefügt, sodass die rechtsseitige Datenstruktur entsteht. Abschließend wird das Ergebnis in 70.1 weitergeleitet.

Parallel dazu werden in dem Cluster die Zeilen 40.8 - 40.15 und Spaltennummern 40.1 - 40.7 zu einem Datenbaum zusammengefasst. Dieser Vorgang ist in Abbildung 72 dargestellt. Da die Daten bereits, wie in Kapitel 4.4 beschrieben, in die richtigen Pfade verschoben

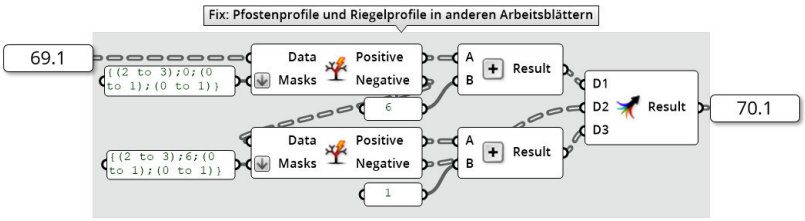


Abbildung 70: Korrektur der Tabellenblätter
Quelle: In Anlehnung an Grasshopper

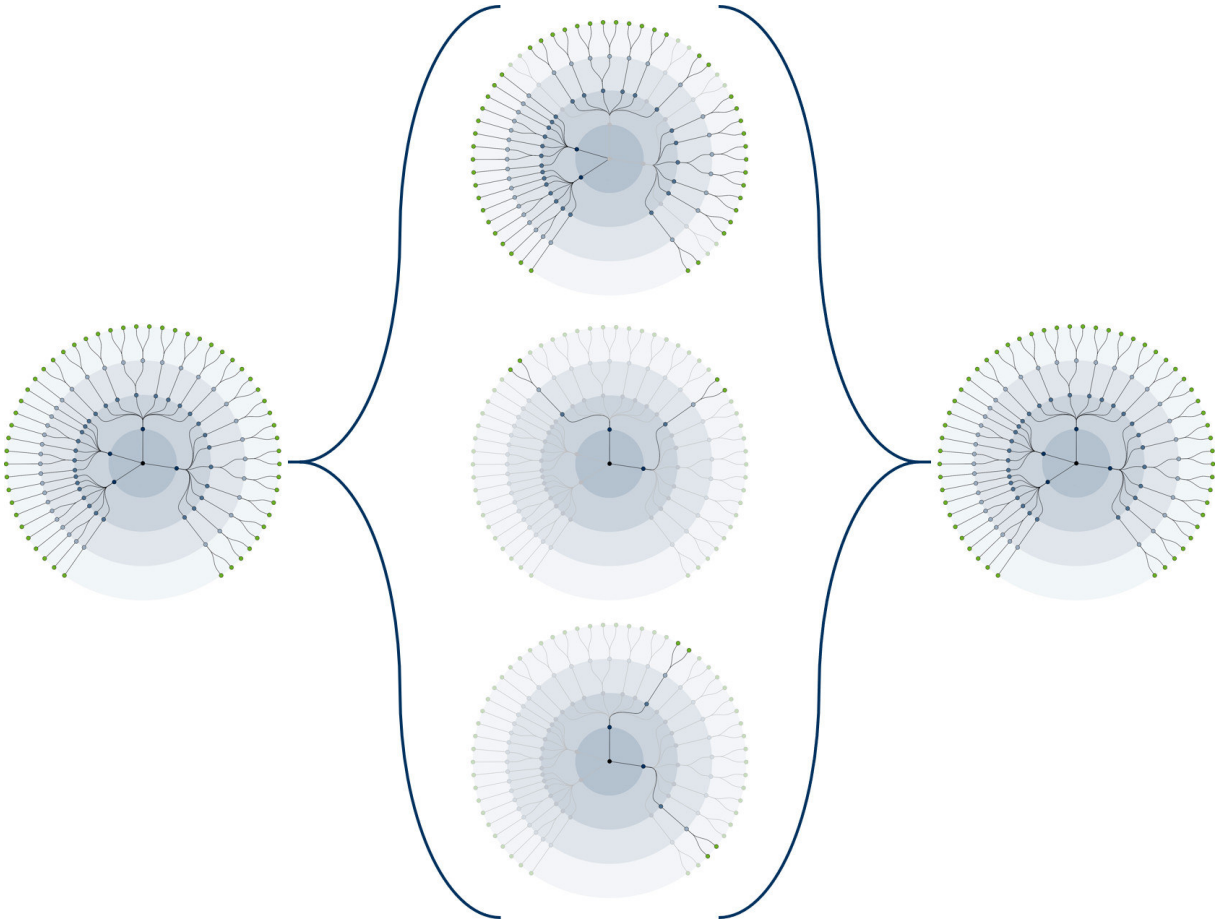


Abbildung 71: Modifizieren der Tabellennummern
Quelle: Eigene Darstellung

wurden, können alle Ergebnisse mithilfe zweier „Merge“-Komponenten zu je einem Baum mit Spalten- und Zeilennummern kombiniert werden. Da die „Excel Reader LEGACY“[4]-Komponente später Integer als Zeilen- und Spaltennummern benötigt, werden die beiden Bäume vorsorglich in Integer umgewandelt. Dazu wird die „Integer“-Komponente verwendet. Die beiden Datenbäume werden in 72.1 und 72.2 weiter geleitet.

Darauffolgend kann die Datenstruktur der Tabellennummern gefiltert werden, sodass sie der Datenstruktur der tatsächlich vorhanden Bauteilkomponenten entspricht. Dieser Filtervorgang erfolgt mit der in Abbildung 73 dargestellten „Tree Branch“ und „Tree Statistics“-Komponenten. Als zu filternde Datenstruktur werden die Tabellennummern aus 70.1 an die „Tree Branch“-Komponente übergeben. Der Filter kann auf Grundlage der Datenstruktur der Spaltennummern 72.1 erzeugt werden. Dazu können mithilfe der „Tree Statistics“-Komponente alle Pfade aus der Datenstruktur ausgelesen werden. Diese Pfade werden zur

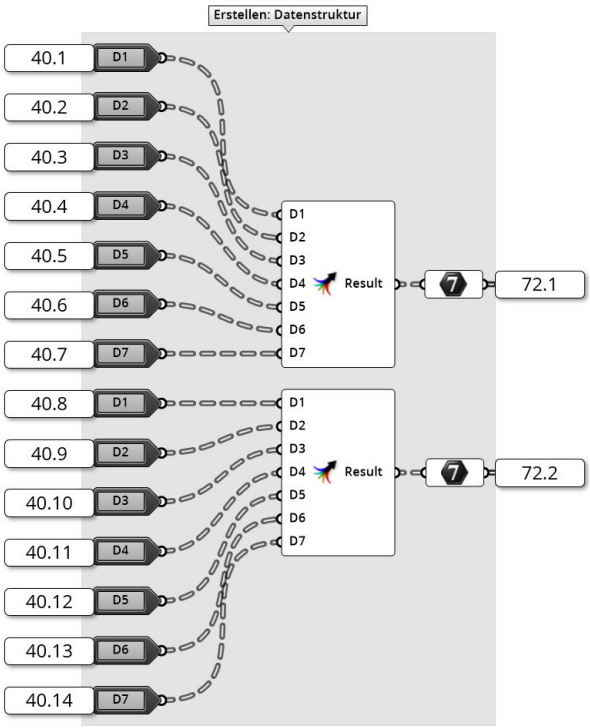


Abbildung 72: Datenbaum der Zeilen- und Spaltennummern
Quelle: In Anlehnung an Grasshopper

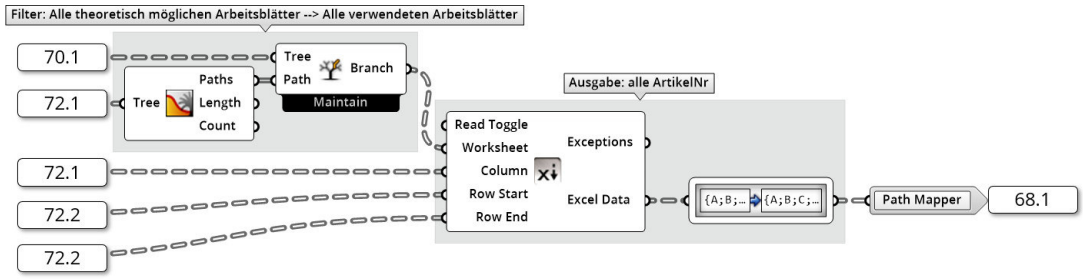


Abbildung 73: Ermittlung der Artikelnummern
Quelle: In Anlehnung an Grasshopper

Angleichung der beiden Datenstrukturen genutzt. Zu diesem Zeitpunkt besitzen alle Datenstrukturen, also die der Tabellen-, Spalten-, und Zeilennummern, die gleiche Datenstruktur, sodass sie problemlos an die „Excel Reader LEGACY“-Komponente aus dem Plug-In „Lunchbox“[4] übergeben werden können. Dazu wird die Zeilennummer sowohl als „Row Start“ als auch als „Row End“ verwendet, da immer nur genau eine Artikelnummer bestimmt werden soll. Die ausgelesenen Artikelnummern werden anschließend in ihrer Datenstruktur angepasst, sodass diese bauteilorientiert ist und als „Excel Data“ 68.1 aus dem Cluster ausgegeben wird.

4.5.2 Unterordner

In Abbildung 74 ist das zweite Cluster aus dem „Cluster: Komponenten Selector“ aus Abbildung 67 dargestellt. Dabei handelt es sich um das „Cluster: SubDirectory Pfade“. Die Geometriedaten befinden sich in Unterordnern innerhalb des zu Beginn dieses Kapitels gewählten Verzeichnisses. Aus dem Cluster wird passend zu den Artikelnummern 68.1 die Namen der Unterordner 74.1 ausgeben.

Innerhalb des Clusters „Cluster: SubDirectory Pfad“ werden vordefinierte Unterordner zunächst in allen theoretisch möglich vorkommende Datenpfaden gespeichert und mithilfe der „Tree Statistics“- und „Tree Branch“-Komponenten werden nicht vorkommenden Pfade herausgefiltert. Bei diesem in Abbildung 75 dargestellten Vorgang, wird als „Tree Statistics“-Input die Datenstruktur der Artikelnummern 68.1 genutzt. Nach dem Filtervorgang wird die bauteilorientierte Datenstruktur der Unterordner als „Branch“ 74.1 aus dem Cluster ausgegeben.

Abbildung 74: Das „Cluster: SubDirectory Pfade“
Quelle: In Anlehnung an Grasshopper

[4] Das Plug-In „Lunchbox“ stammt vom Autoren Nathan Miller und ist unter https://www.food4rhino.com/app/lunchbox (aufgerufen am 28.09.2019) verfügbar.

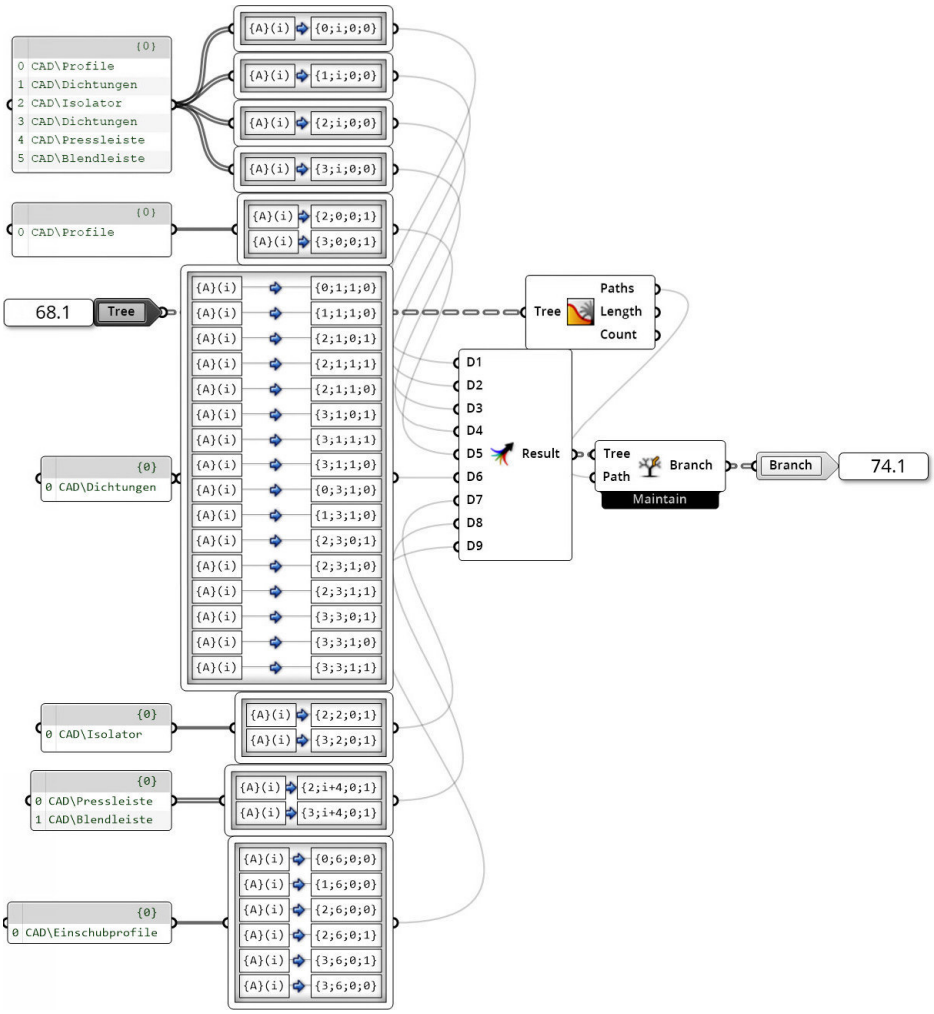


Abbildung 75: Das Innere des „Cluster: SubDirectory Pfade“
Quelle: In Anlehnung an Grasshopper

4.5.3 Geometrien der Bauteilkomponenten

Bevor die Bauteilgeometrien geladen werden können, müssen im in Abbildung 67 dargestelltem „Cluster: KomponentenPicker“ die Artikelnummern, Unterordner und das gewählte Verzeichnis zu Dateipfaden umgewandelt werden. Dieser Vorgang wird vom Python-Skript Q13 „Script: Filename“ übernommen. Dieses Skript ist in Anhang Q13 gegeben. Dem Skript kann wahlweise eine Artikelnummer oder ein vollständiger Dateiname übergeben werden. Das Skript setzt alle ihm gegebenen Informationen zu einem vollständigen Dateipfad zusammen. Mithilfe des Dateipfades kann die in Abbildung 67 dargestellte „Import 3DM“-Komponente die Bauteilgeometrien laden. Zum Abschluss des Clusters wird die Datenstruktur bereinigt, sodass sie weiterhin bauteilorientiert vorliegt und zurückgegeben.

In Abbildung 76 ist die so erzeugte Querschnittsgeometrie abgebildet. Auf der linken Seite ist ein Pfosten- und auf der rechten ein Riegelprofil dargestellt. Es ist zu erkennen, dass die

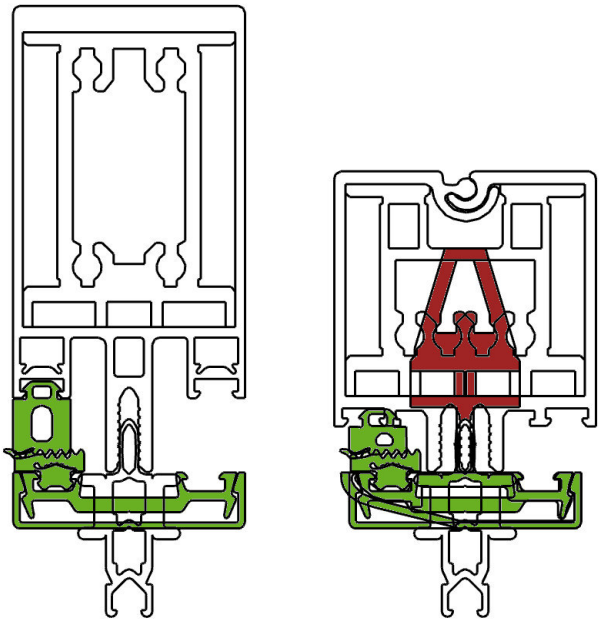


Abbildung 76: Profilquerschnitt direkt nach dem Laden der Geometrie
Quelle: In Anlehnung an Rhinoceros 6

grün markierten Bauteile nicht richtig positioniert sind. Bei den Dichtungen ist dabei zu beachten, dass diese noch nicht doppelt verbaut werden. Dies ist darin begründet, dass die Geometrie zwar erzeugt wurde, jedoch übereinander liegt. Im späteren Verlauf wird eine dieser Geometrien um die Symmetrieachse des Profils gespiegelt, sodass die Dichtungen richtig positioniert sind. Die äußeren Profilkomponenten werden später entsprechend der Elementdicke verschoben. Die rot markierten Bereiche stellen Überlappungen des variablen Riegelprofils dar. Dies hat den Hintergrund, dass das Riegelprofil noch nicht geöffnet wurde, und erst ab einem Öffnungswinkel von rund 38 ° kollisionsfrei ist.

5 Vorbereitung der Profilquerschnitte

Nachdem die einzelnen Komponenten des Fassadenknotens geladen sind, muss die Geometrie für die Erzeugung eines digitalen Fassadenknotens weiter vorbereitet werden. Hierbei stehen zwei Aufgaben im Fokus:

1. Erzeugung von „Cuttern“
2. Korrektur der Positionen der Komponenten

Die Einordnung dieses Abschnitts in den gesamten Algorithmus wurde in Kapitel 4.1 gegeben. In Abbildung 25 (zu finden auf Seite 51) entspricht dieser Teil den hellblauen Komponenten.

5.1 Erzeugung der Cuttergeometrie

Nach der Erzeugung Profilquerschnitte, sollen in diesem Kapitel die sogenannten „Cutter“ generiert werden. Eine grobe Definition von Cutter erfolgte bereits in Kapitel 3.4.2.2. Cutter sind sämtliche Geometrien, die zum Verschneiden der Profile genutzt werden und daher keine Addition weiterer Geometrie zum Profilquerschnitt hervorrufen. Die Cuttergeometrien orientieren sich dabei an einigen Komponenten der Profilquerschnitte, sind dabei jedoch stark vereinfachte Geometrien, sodass die boolschen Operationen, mit denen die Profile verschnitten werden, schneller und fehlerfrei berechnet werden können. Zudem sind die Geometrien der Cutter in einer oder mehreren Richtungen im Vergleich zur dargestellten Bauteilkomponente erweitert, sodass Bauteilelemente, die über den Profilquerschnitt heraus kragen, restlos entfernt werden können. Insgesamt werden vier Cutter benötigt. Davon bilden drei Cutter die Geometrie der Pfostenprofile nach. Genauer repräsentieren zwei der Cutter die Geometrie der Tragprofile und einer die Geometrie der Press- und Blendleiste. Der Vierte vereinfacht die Tragprofile der Riegel. Die Erzeugung der Geometrien erfolgt an dieser Stelle zunächst als Surface-Flächen. Im späteren Programmablauf werden die Flächen mit den Profilquerschnitten extrudiert, sodass sie Volumenkörpern entsprechen.

Die Erzeugung dieser Geometrien erfolgt in dem in Abbildung 77 dargestellten „Cluster: Cutter“. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/04_ClusterCutter-Komplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Damit die Cuttergeometrie erzeugt werden kann, werden drei Inputparameter benötigt. Diese sind zum

einen die gewählte Spaltennummer der Pressleiste 40.5, der Öffnungswinkel der Riegelprofile 40.15 und die gewählte Elementdicke 26.8 als „Scheibendicke“. Als Ergebnis des Clusters werden die Cuttergeometrien in 77.1 als „Result“ ausgegeben.

Die erste Cuttergeometrie repräsentiert die Tragprofile der Pfosten. Dabei wird die Geometrie in Richtung der Tiefe der Pfosten erweitert, sodass keine Abhängigkeit zur Pfostentiefe besteht. Dadurch ergibt sich keine Beziehung zu den gewählten Parametern, sodass die benötigte Geometrie für alle Fälle gilt und diese nicht veränderlich ist. Mit den in Abbildung 78 dargestellten Koordinaten, können alle Eckpunkte der Geometrie definiert und zu einer Liste zusammengefasst werden. Anschließend können die Punkte, wie in Abbildung 79 verdeutlicht, zu einer Polyline zusammengefasst werden. Dazu wird die „PolyLine“-Komponente verwendet. Mit der „Boundary Surfaces“-Komponente kann aus der Polyline eine Surface-Fläche erzeugt werden, deren Geometrie durch die Polyline begrenzt wird. Diese Surface-Flächen können im späteren Verlauf extrudiert werden. Da der erste Cutter

Abbildung 77: Das „Cluster: Cutter“
Quelle: In Anlehnung an Grasshopper

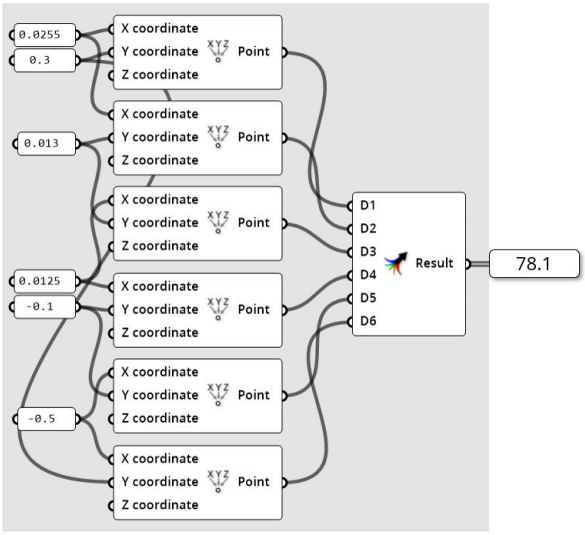


Abbildung 78: Erzeugen der Cuttergeometire (1)
Quelle: In Anlehnung an Grasshopper

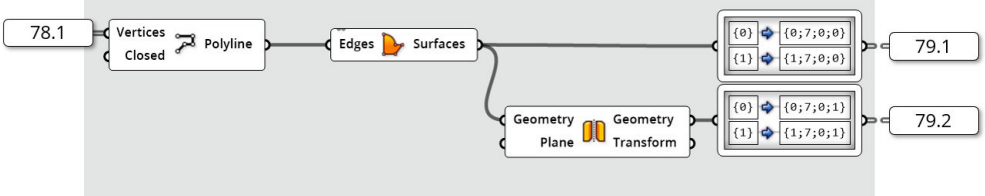


Abbildung 79: Erzeugen der Cuttergeometrie (2)
Quelle: In Anlehnung an Grasshopper

nur eine Hälfte eines Pfostenprofils repräsentiert, muss eine weitere gespiegelte Geometrie hinzugefügt werden. Dies kann mit der „Mirror“-Komponente erreicht werden. Da außerdem zwei Pfostenprofile vorhanden sind, werden die Cutter jeweils für beide Profile sowohl normal, als auch gespiegelt gespeichert. Die Verdopplung erfolgt mit dem „Path Mapper“, wodurch der Cutter insgesamt viermal vorhanden ist. Als Pfade für den ersten Cutter sind {(0,1);7;0;(0,1)} vorgesehen. Es ist dabei zu beachten, dass bei den Cuttern eine Abweichung von der bauteilorientierten Datenstruktur erfolgt, da anderenfalls nicht genügend Hierarchieebenen vorhanden sind. Dies ist darin zu begründen, dass mehr als nur ein Cutter für die Pfostenprofile vorhanden ist. In der Cutter-Datenstruktur ist der Aufbau:

{X;Y;Z;A}[i]

Mit: X - Bauteil (0,1 – Pfosten; 2,3 – Riegel)
Y - Bauteilkomponente (= 7 (Cutter))
Z - Cutter-Nr. (da mehrere Cutter vorhanden sein können)
A - Cutter-Variante (z. B. Spiegelung oder Bauteilkomponentenhälfte)
i - Einzelne Geometrien (in diesem Fall je eine Fläche)

Dabei wird A genutzt, um sowohl Z als auch A der bauteilorientierten Datenstruktur zu übernehmen. Da nur eine geringe Anzahl an Cuttern vorhanden ist, stellt dies im weiteren Verlauf kein Problem dar.

Der zweite Cutter repräsentiert die Press- und Blendleiste der Pfostenprofile. Damit der Cutter an die gewählte Press- und Blendleiste angepasst werden kann, wird die Spaltennummer der Pressleiste „S_Pressleiste“ 40.5 benötigt. Dies ist in Abbildung 80 ersichtlich. Da nur die Pfostenprofile betroffen sind, können diese aus der Menge aller „S_Pressleiste“ herausgefiltert werden. Hierzu wird die Maske {(0,1);*:*:*} genutzt, sodass alle Informationen zu den Pfosten in „Positive“ herausgegeben werden. Diese können mit dem Python-Skript Q14 „Script: CutterPressleiste“ weiterverarbeitet werden. Das Skript ist in Anhang Q14 beigelegt. In diesem wird über if-Anweisungen eine zur Pressleiste passende Liste mit den Eckpunkten des Cutters ausgegeben. Das weitere Verfahren ist identisch zum Verfahren des ersten Cutters. Die Eckpunkteliste des zweiten Cutters wird, wie in Abbildung 79 dargestellt, zu einer Polyline und anschließend zu einer Surface-Fläche zusammengefügt.

Abbildung 80: Erzeugen der Cuttergeometrie (3)
Quelle: In Anlehnung an Grasshopper

Anschließend erfolgt eine Spiegelung, sodass beide Seiten der Pfostenprofile repräsentiert werden. Lediglich die zugewiesenen Pfade werden auf {(0,1);7;1;(0,1)} gesetzt. Da diese Outputs nicht extra dargestellt sind, werden sie in Anlehung an Abbildung 79 als 79.1* und 79.2* bezeichnet.

Der dritte Cutter repräsentiert die Tragprofile und inneren Dichtungen der Riegel. Da die Geometrie der Tragprofile der Riegelprofile maßgeblich vom Öffnungswinkel der Riegelprofile 40.15 und die innere Dichtung von der Elementdicke 26.8 abhängt, werden diese Parameter zur Erzeugung des Cutters benötigt. Abbildung 81 verdeutlicht, wie die Inputparameter für das Skript Q15 „Script: CutterRiegel“ genutzt werden. Da der dritte Cutter nur ein Rechteck sein muss, werden im in Anlage Q15 beigelegten Skript die Höhe und Breite des Rechtecks in den Zeilen 13 bis 23 festgelegt und anschließend in Zeile 25 in die dazugehörigen Punkte umgewandelt. Somit gibt das Skript, wie beim vorherigen Cutter, eine Liste mit den Eckpunkten der Cuttergeometrie aus. Zur Erzeugung der Surface-Fläche wird diesmal jedoch ein anderer Ansatz verfolgt. Die Punkteliste wird mit der „List Item“-Komponente in die einzelnen Eckpunkte zerlegt. Diese werden mit der „4Point Surface“-Komponente wieder zusammengesetzt. Da die anderen Cutter aus je fünf Eckpunkten bestehen, kann die Komponente bei diesen Cuttern nicht angewendet werden. Analog zu den ersten beiden Cuttern wird die Geometrie gespiegelt und mit den „Path Mappern“ in die vorgesehene Position der Datenstruktur verschoben.

Der vierte Cutter ist eine Abwandlung des ersten Cutters. Die Erzeugung der Cuttergeometrie ist analog zu Abbildung 78 und 79. Ausschließlich ein Inputwert und die Ausgabepfade werden verändert. Da die Outputs nicht dargestellt sind, werden sie folgend in Anlehnung Abbildung reffig:ClusterCutter02 als 79.1** und 79.2** bezeichnet.

In Abbildung 82 wird verdeutlicht, dass alle Cuttergeometrien zu einem Datenbaum zusammengefasst werden. Da die Datenstruktur aufgrund der variablen Riegel variieren kann, muss der Baum gefiltert werden, um Pfade entsprechend umzubenennen. Dazu wird die „Split Tree“-Komponente mit einer Filtermaske aus dem Python-Skript Q16 „Script: FilterCutter“ verwendet. Das Skript ist in Anlage Q16 enthalten. In dem Skript wird auf Grundlage

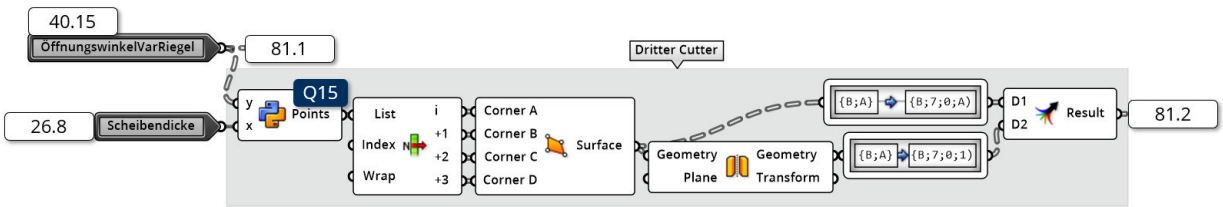


Abbildung 81: Erzeugen der Cuttergeometrie (4)
Quelle: In Anlehnung an Grasshopper

des Öffnungswinkels der Riegel 40.15 eine geeignete Maske ausgegeben. Wie in Abbildung 82 zu sehen, kann die Maske auch leer sein, sodass keine Anpassung vorgenommen werden muss. Sollte eine Maske vorhanden sein, werden die Pfade entsprechend der Eingaben umbenannt und anschließend mit dem Rest der Datenstruktur vereint. Das Ergebnis wird in 77.1 als „Result“ aus dem Cluster ausgegeben.

Nachdem die Geometrien der Cutter und die der Bauteilkomponenten erzeugt und die bauteilorientierte Datenstruktur aufgebaut wurde, können die beiden Datenbäume mit der „Merge“-Komponente zusammengeführt werden. Dieser Vorgang ist in Abbildung 84 dargestellt. Zu diesem Zeitpunkt liegt zum ersten Mal die vollständige Datenstruktur vor, wie sie in Kapitel 3.4.2.2 beschrieben wurde.

5.2 Positionierung der Bauteilkomponenten

Abbildung 83 verdeutlicht die Geometrien der einzelnen Bauteilkomponenten, nachdem diese durch das „Cluster: KomponentenPicker“ geladen wurden. In der Grafik ist die Querschnittgeometrie des Pfostens {0} dargestellt. Dabei ist zu erkennen, dass die grün markierten Bauteilkomponenten nicht richtig positioniert sind. Hier sind zum Beispiel das Einschubprofil und das Tragprofil zu nennen. Der Grund dafür ist, dass diese Komponenten

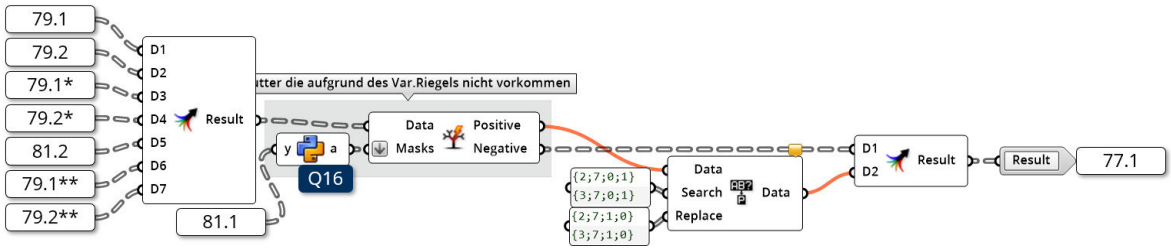


Abbildung 82: Erzeugen der Cuttergeometrie (5)
Quelle: In Anlehnung an Grasshopper

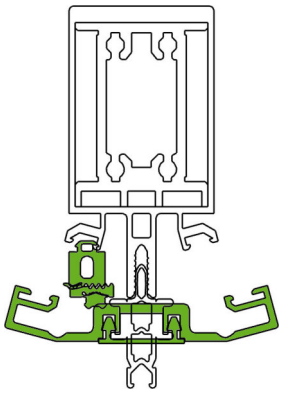


Abbildung 83: Geometrien der Bauteilkomponenten des Pfostens {0} nach dem Laden der .3dm-Datein
Quelle: In Anlehnung an Rhinoceros 6

noch durch verschiedene Einflussparameter verschoben werden müssen. So können die relativen Positionen der Komponenten zueinander bei der Erstellung der .3dm-Dateien nicht berücksichtigt werden. Andere Bauteilkomponenten unterliegen den Einflussparametern des Algorithmus. Dabei sind die gewählte Elementdicke und die Geometriestellung der Pfosten und Riegel im Knoten maßgebend. Zudem besteht eine Abhängigkeit zu vorangegangen Programmentscheidungen. Zum Beispiel können die äußeren Dichtungen nur auf Grundlage der gewählten Pressleiste erfolgreich positioniert werden.

Um diese Fehler zu korrigieren, wird das in Abbildung 84 dargestellte „Cluster: KorrigiereQuerschnitt“ in den Algorithmus implementiert. Es korrigiert die Positionen der fehlerhaft positionierten Komponenten und verschiebt den Gesamtquerschnitt so, dass sich der Ursprung in der Mitte der späteren Elemente befindet. Das Cluster benötigt dazu die Geometriedaten der Bauteilkomponenten 66.1 und Cutter 77.1, die vom Nutzer eingestellten Parameter „Elementdicke“ 26.8, „MPfosten“ 26.1 und „MRiegel“ 26.2, die daraus resultierenden Winkel „LeftAngle“ 29.1 und „RightAngle“ 29.2 und die vom Algorithmus gewählten Bauteilkomponenten, beziehungsweise deren Spaltennummern, „S_Tragprofil“ 40.1 und „S_Pressleiste“ 40.5. Nach dem das Cluster ausgeführt wurde, wird die korrigierte Querschnittsgeometrie in „Geometry“ 84.2 ausgegeben. Das Cluster ist in weitere Cluster aufgeteilt. Der Aufbau der Cluster ist dabei:

1. Cluster: ÄußereProfileVerschieben
2. Cluster: DichtungenDrehenUndPositionieren
3. Cluster: DichtungenSpiegeln
4. Cluster: VarRiegelAufdrehen
5. Cluster: GeometrieZurScheibenmitte
6. Cluster: KorrigierePressBlendleisteRiegel

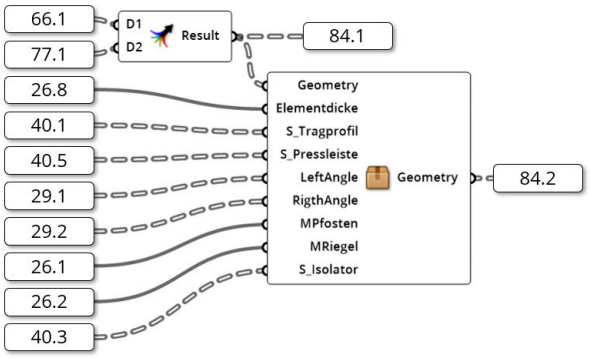


Abbildung 84: Das „Cluster: KorrigiereQuerschnitt“
Quelle: In Anlehnung an Grasshopper

5.2.1 Äußere Profile verschieben

Das erste der in Abbildung 83 ersichtlichen geometrischen Probleme ist, dass die Blend- und Pressleiste nicht richtig positioniert sind und sich derzeit innerhalb des Isolators befinden. Um die Position dieser Bauteilkomponenten zu korrigieren, wird das „Cluster: ÄußereProfileVerschieben“ implementiert. Wie in Abbildung 85 dargestellt ist, wird zum Verschieben der äußeren Profile zum einen die zu korrigierende Geometrie 84.1 und zum anderen die Elementdicke 26.8 benötigt. Wie Abbildung 86 zeigt, wird im „Cluster: ÄußereProfileVerschieben“ zunächst mithilfe des Python-Skripts Q17 „Script: ÄußereVerschiebung“ eine Strecke „a“ berechnet, die der Verschiebung in y-Richtung entspricht. Das Skript ist in Anlage Q17 beigelegt. Die Berechnung erfolgt dabei nach folgender Formel:

$$a = \begin{cases} c & \text{für g=0} \vee \text{g=1} \\ c - 0.004 & \text{für } 1 < \text{g} \leq 4 \\ c - 0.004 - 0.006 * int(\frac{g-2}{3}) & \text{für } \text{g} > 4 \end{cases} \tag{5.1}$$

$$mit : c = -0.0261$$

Dabei ist „c“ die Mindestverschiebung. Somit ist c die Verschiebung, die stattfinden muss, wenn die kleinste Elementdicke gewählt wurde. Die so ermittelte Strecke wird anschließend an „Cluster: GibMaskeWert“ übergeben. Da das Cluster ausschließlich aus Datenstruktur-Management-Operationen besteht, wird dieses nicht näher erläutert. Aus dem Cluster wird die bauteilorientierte Datenstruktur ausgegeben, die in allen Pfaden der Press- und Blendleiste die Verschiebung „a“ aufweist. Alle anderen Pfade besitzen den Wert 0. Mithilfe der „Vector XYZ“-Komponente können die Strecken anschließend in Vektoren umgewandelt werden, wobei die Strecke „a“ der Verschiebung in y-Richtung entspricht. Die so gebildeten Vektoren können genutzt werden, um die Geometrie der Bauteilkomponenten zu ver-

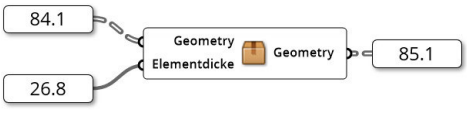


Abbildung 85: „ÄußereProfileVerschieben“
Quelle: In Anlehnung an Grasshopper

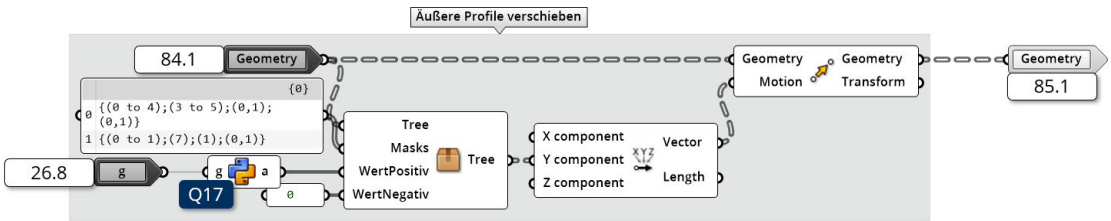


Abbildung 86: Korrektur der äußeren Profile
Quelle: In Anlehnung an Grasshopper

schieben. Dazu wird die „Move“-Komponente verwendet. Die korrigierte Geometrie wird in 85.1 ausgegeben. Diese ist in Abbildung 87 dargestellt. Es ist zu erkennen, dass die Position der Press- und Blendleiste deutlich verbessert wurde.

5.2.2 Dichtungen positionieren

In Abbildung 87 ist erkennbar, dass die grün markierten Dichtungen weiterhin fehlerhaft sind. Dies betrifft sowohl die äußere als auch die innere Dichtung. Daher werden die Dichtungen mithilfe des „Cluster: Dichtungen Drehen und Positionieren“ per Rotation und Translation in die richtige Position gebracht. Das in Abbildung 88 gegebene Cluster benötigt die zu rotierende Geometrie 85.1, die Spaltennummer des Tragprofils 40.1 und die der Pressleiste 40.5. Als Output wird die korrigierte Geometrie ausgegeben 88.1. Wie Abbildung 89 zeigt, werden zunächst die Inputparameter gefiltert, sodass nur die Daten zu den Pfosten übrig bleiben. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/05_ClusterDichtungPosKomplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Dies hat den Hintergrund, dass die Positionierung der Riegeldichtung separat erfolgt, da diese vor allem vom Öffnungswinkel des variablen Riegelprofils abhängig ist. Bei den Pfosten hingegen, ist die Abhängigkeit auf die Wahl der Trag- und Pressleistenprofile beschränkt. Der Filtervorgang erfolgt mit den entsprechenden Masken und „Split Tree“-Komponenten. Da „S_Tragprofil“ und „S_Pressleiste“ in der Datenstruktur bei den jeweiligen Bauteilkomponenten gespeichert sind, die Informationen in diesem Fall jedoch

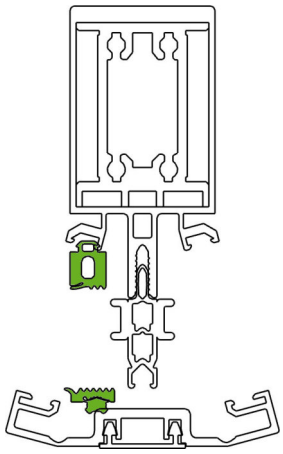


Abbildung 87: Korrigierte äußere Profile
Quelle: In Anlehnung an Rhinoceros 6

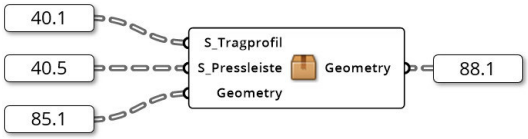


Abbildung 88: Das „Cluster: Dichtungen Drehen und Positionieren“
Quelle: In Anlehnung an Grasshopper

auf die inneren und äußeren Dichtungen, also anderen Bauteilkomponenten, angewandt werden sollen, werden die Daten in die Datenstrukturplätze der zu modifizierenden Dichtungen verschoben. Dies erfolgt mit dem „Path Mapper“. Die Daten werden entsprechend der Abbildung weitergeleitet.

Die gefilterten und umstrukturierten Daten der „S_Tragprofil“ 89.1 und „S_Pressleiste“ 89.2 können, wie in Abbildung 90 ersichtlich, weiterverwendet werden, um die Positionierung der Dichtungen zu definieren. Dazu werden die beiden Skripte Q18 „Script: InnereDichtung“ und Q19 „Script: ÄußereDichtung“ verwendet. Beide Skripte sind in den Anlagen Q18 und Q19 beigefügt. In Skript Q18 werden die Rotationswinkel der inneren Dichtungen bestimmt. Dies geschieht auf Grundlage der Spaltennummer der gewählten Tragprofile 40.1 „t“. Der Rotationswinkel „w“ wird dabei über die Formel

$$w = (t - 1) * 5 \tag{5.2}$$

bestimmt. Somit kann der Winkel des gewählten Profils ermittelt und die Dichtung entsprechend gewählt werden. Das Skript Q19 ist etwas komplexer, da hier nicht nur der Rotationswinkel, sondern auch Daten zur Translation ermittelt werden müssen. Zudem kann der Rotationswinkel nicht so trivial wie in Formel 5.2 bestimmt werden, da die Pressleistenwin-

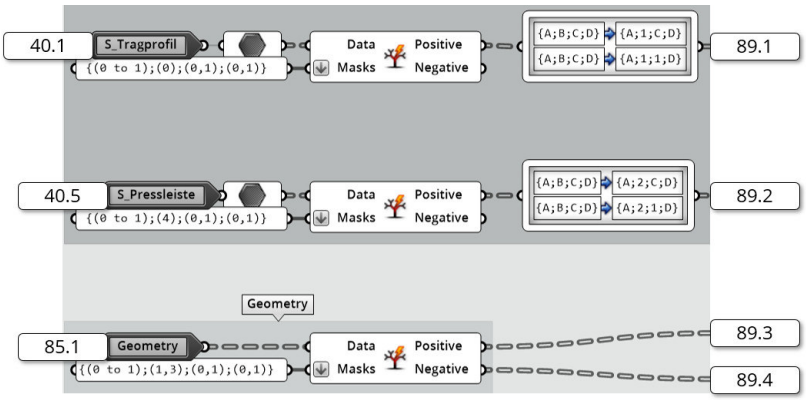


Abbildung 89: Filtern der Inputparameter
Quelle: In Anlehnung an Grasshopper

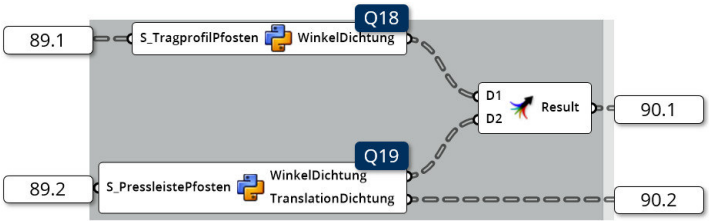


Abbildung 90: Erzeugung der Transformationsdaten
Quelle: In Anlehnung an Grasshopper

kel nicht gleichmäßig abgestuft sind. Die Logik zur Wahl des Rotationswinkels entspricht folgender Formel:

$$w = \begin{cases} 0 & \text{für p=0} \vee \text{p=1} \\ 7.5 & \text{für p=2} \\ 15 & \text{für p=3} \\ 20 & \text{für p=4} \\ 30 & \text{für p=5} \\ 35 & \text{für p=6} \end{cases} \tag{5.3}$$

„p“ entspricht in der Formel der Spaltennummer der gewählten Pressleisten. Mithilfe einer zu Formel 5.3 vergleichbaren Formel kann im Skript aus verschiedenen Vektoren, die die auszuführende Translation abbilden, gewählt werden. Dabei gilt:

$$\vec{t} = \begin{cases} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \text{für p=0} \\ \begin{pmatrix} -0.0107 \\ 0 \\ 0 \end{pmatrix} & \text{für p=1} \\ \begin{pmatrix} -0.01275 \\ 0.00008 \\ 0 \end{pmatrix} & \text{für p=2} \\ \begin{pmatrix} -0.01625 \\ 0.0001 \\ 0 \end{pmatrix} & \text{für p=3} \\ \begin{pmatrix} -0.02425 \\ 0.0025 \\ 0 \end{pmatrix} & \text{für p=4} \\ \begin{pmatrix} -0.0293 \\ 0.0029 \\ 0 \end{pmatrix} & \text{für p=5} \\ \begin{pmatrix} -0.03175 \\ 0.0079 \\ 0 \end{pmatrix} & \text{für p=6} \end{cases} \tag{5.4}$$

Über die Formeln 5.3 und 5.4 kann eine genaue Positionierung der äußeren Dichtungen auf Grundlage der gewählten Pressleiste vorgenommen werden. Die Rotation wird wie in

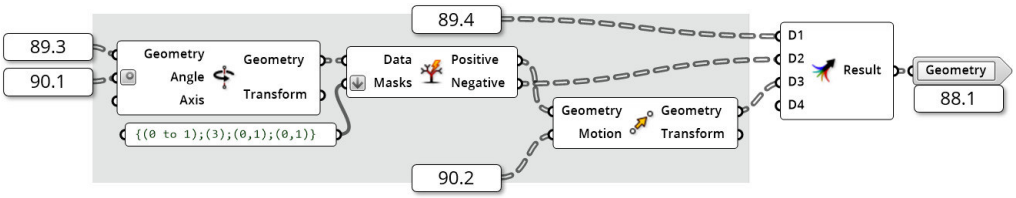


Abbildung 91: Rotation und Translation der Dichtungen
Quelle: In Anlehnung an Grasshopper

Abbildung 91 ersichtlich mit der „Rotate Axis“-Komponente vorgenommen. Die hierfür notwendigen Achsen sind direkt in der Komponente gespeichert, da diese für jeden Anwendungsfall gleich sind. Als zu rotierende Geometrie werden die herausgefilterten inneren und äußeren Dichtungen 89.3 als „Geometry“ an die Komponente übergeben. Als Rotationswinkel kommen die beiden „WinkelDichtung“en 90.1 zum Einsatz. Dabei ist zu beachten, dass die Winkel in den Skripten Q18 und Q19 in Grad definiert wurden und Grasshopper standardmäßig mit Rad rechnet. Um dieses Problem zu lösen, kann der „Angle“-Input der „Rotate Axis“-Komponente auf Grad umdefiniert werden. Nach der Rotation kann für die äußeren Dichtungen die Translation vorgenommen werden. Hierzu werden zunächst die äußeren Dichtungen herausgefiltert und an die „Move“-Komponente übergeben. Als „Motion“ werden die in Formel 5.4 bestimmten Vektoren verwendet. Abschließend werden alle Datenbaumteile wieder mit der „Merge“-Komponente zusammengesetzt und aus dem Cluster als „Geometry“ 88.1 ausgegeben.

Durch das „Cluster: Dichtungen Drehen und Positionieren“ können sowohl die inneren als auch die äußeren Dichtungen der Pfostenprofile in die richtige Position gebracht werden. Dieser Fortschritt ist in Abbildung 92 dargestellt.

5.2.3 Dichtungen spiegeln

Wie in Abbildung 92 auffällt, sind die grün markierten Dichtungen derzeit nur auf einer Seite vorhanden. Das „Cluster: Dichtungen Spiegeln“ spiegelt die Dichtungen. Dabei ist zu beachten, dass keine neue Geometrie erzeugt, sondern bestehende Geometrie modifiziert wird. Das hat den Hintergrund, dass die gespiegelten Dichtungen bereits seit Kapitel 4.4.2 und 4.4.4 berücksichtigt werden. Dies ist notwendig, da die Winkel zwischen der Referenzebene und den beiden Elementebenen unterschiedlich sein können und somit zwei unterschiedliche Dichtungen benötigt werden.

Aus diesem Grund wird in Abbildung 93 die Geometrie korrigiert. So wird zunächst die Datenstruktur der Geometrie 88.1 gefiltert. Dabei kommen zwei verschiedene Masken zum Einsatz, da sich die zu spiegelnde Geometrie an zwei verschiedenen Positionen der Datenstruktur befinden kann. Mithilfe der „Mirror“-Komponente werden die inneren und äußeren Dichtungen gespiegelt. Anschließend werden sie mit der „Merge“-Komponente wieder in die Datenstruktur eingesetzt und als „Geometry“ 93.1 aus dem Cluster ausgegeben. Die Korrektur der Dichtungen ist in Abbildung 94 (a) dargestellt.

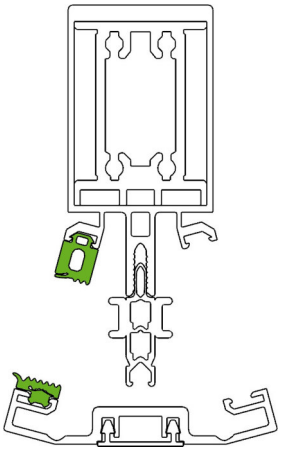


Abbildung 92: Korrigierte innere und äußere Dichtungen
Quelle: In Anlehnung an Rhinoceros 6

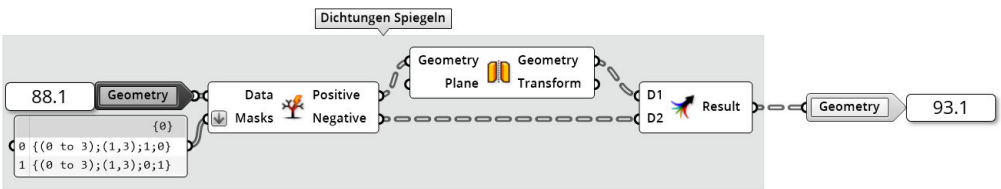
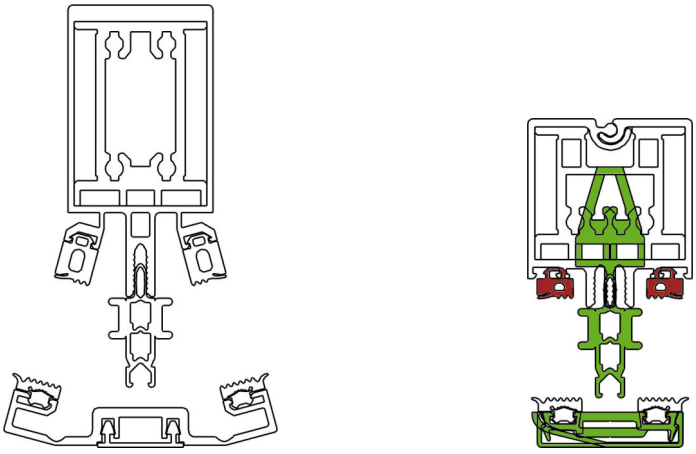


Abbildung 93: Spiegelung der Dichtungen
Quelle: In Anlehnung an Grasshopper

5.2.4 Aufdrehen der variablen Riegelprofie

Nachdem die Positionen der Pfostenkomponenten korrigiert wurden, werden folgend die der Riegel korrigiert. Die Riegel liegen zu diesem Zeitpunkt in dem in Abbildung 94 (b) dargestellten Zustand vor. Einerseits ist zu sehen, dass das variable Profil noch nicht geöffnet wurde und dadurch die Geometrien der Bauteilkomponenten in den grün markierten Bereichen ineinander liegen. Dies betrifft auch die beiden Press- und Blendleisten sowie die Isolatoren die derzeit übereinanderliegen. Andererseits kann festgestellt werden, dass die rot markierten inneren Dichtungen nicht richtig positioniert sind. Dies liegt daran, dass die Dichtungen auf das Standardprofil eingestellt sind. Da die variable Variante etwas breiter ist, müssen die Dichtungen nach außen hin verschoben werden.

Um diese Probleme zu lösen, wurde das in Abbildung 95 dargestellte Cluster implementiert. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/06_ClusterVarRiegelKomplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Zum einen soll das „Cluster: VarRiegelAufdrehen“ die variablen Profile öffnen und zum anderen sollen die Positionen einzelner Komponenten weiter optimiert werden. Zum Aufdrehen der variablen Riegelprofile werden die beiden Winkel „LeftAngle“ 29.1 und „RightAngle“ 29.2, die Spalten-



(a) (b)

Abbildung 94: Zustand der Pfosten (a) und Riegel (b) nach dem „Cluster: Dichtungen Spiegeln“
Quelle: In Anlehnung an aus Rhinoceros 6

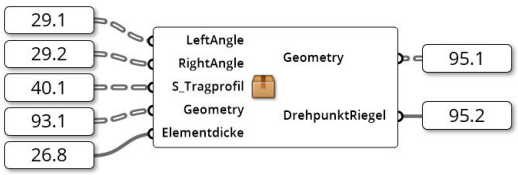


Abbildung 95: Das „Cluster: VarRiegelAufdrehen“
Quelle: In Anlehnung an Grasshopper

nummer des gewählten Tragprofils „S_Tragprofil“ 40.1, die gewählte Elementdicke 26.8 sowie die zu modifizierende Geometrie 93.1 benötigt. Die meisten dieser Inputparameter müssen zunächst in ihrer Datenstruktur angepasst werden, sodass sie für den Algorithmus verarbeitbar sind. Dieser Vorgang ist in Abbildung 96 und 97 dargestellt. Alle Daten, außer der Elementdicke, werden zunächst gefiltert. Dabei werden die Informationen der Riegel von den Informationen der Pfosten separiert. Die Spaltennummern der Tragprofile 40.1 müssen außerdem in ihrer Information angepasst werden. Dies hat den Hintergrund, dass die Positionierung der inneren Dichtung und das Aufdrehen der beiden Profilhälften nur stattfinden, wenn ein variables Profil gewählt wurde. Da die Spaltennummer des Riegeltragprofils den Wert 1 aufweist, handelt es sich um das Standardprofil. Anderenfalls weist die Spaltennummer den Wert 2 auf. Daher kann die Spaltennummer um 1 reduziert werden. So werden die Rotationswinkel und die Translationsvektoren mit 0 multipliziert, falls es sich um ein Standardprofil handelt. Im Fall des variablen Profils findet eine Multiplikation mit 1 statt, sodass die Rotation und Verschiebung erfolgen. Die beiden Winkel „LeftAngle“ und „RightAngle“ wurden vor dem Filtervorgang zu einer Datenstruktur zusammengefasst. Nach dem Filtern werden sie als 96.1 in Abbildung 97 weiter gefiltert. Hierbei kommt die Maske aus dem Python-Skript Q20 „Script: MaskeInputFix“ zur Anwendung. Das Skript bekommt als Input die Anzahl der Pfade von „S_Tragprofil“. Dies hat den Hintergrund, dass diese Anzahl der Pfade in der Datenstruktur für variable und standardisierte Profile unterschiedlich ist. Die Anzahl der berechneten Winkel bleibt hingegen immer gleich. Sollten nun beide Datenstrukturen miteinander verrechnet werden, müssen beide Datenbäume auf die gleiche Länge gebracht werden. Anschließend können die Winkel mit einer Multiplikation der Winkel und der Korrekturfaktoren aus „S_Tragprofil“ verrechnet werden, sodass

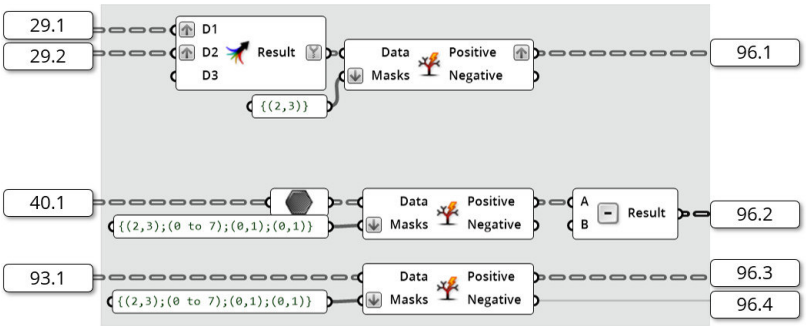


Abbildung 96: Vorbereitung der Inputdaten (1)
Quelle: In Anlehnung an Grasshopper

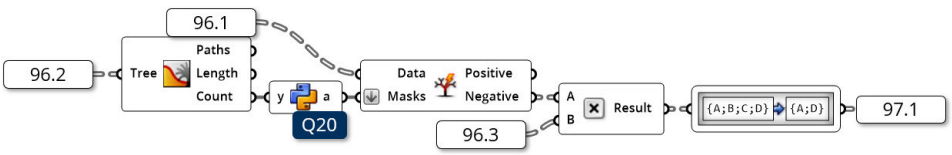


Abbildung 97: Vorbereitung der Inputdaten (2)
Quelle: In Anlehnung an Grasshopper

bei Standardprofilen die Winkel den Wert 0 erhalten. Darauffolgend wird die Datenstruktur mithilfe des „Path Mappers“ angepasst.

In Abbildung 98 wird gezeigt, wie die Datenstruktur der Geometrie 96.3 gefiltert wird. Hierbei werden zunächst die inneren Dichtungen und die Isolatoren der beiden Riegelprofile gefiltert. Um diese zu erhalten, kann die Maske {(2,3);(1,2);0;(0,1)} angewendet werden. Somit sind im Ausgang „Positive“ 98.1 die Bauteilkomponenten „innere Dichtung“ und „Isolator“ enthalten. Der Ausgang „Negative“ 98.2 beinhaltet die restlichen Bauteilkomponenten der Riegelprofile. Die Korrekturfaktoren 96.2, die eine Multiplikation mit dem Wert 0 ermöglichen, werden ebenfalls gefiltert, sodass genau zwei dieser Faktoren in „Positive“ ausgegeben werden. In dem Panel vor dem „Path Mapper“ werden Verschiebungsvektoren definiert, die angewendet werden müssen, sollte es sich um variable Profile handeln. Dabei sind vier Vektoren für die vier Bauteilkomponenten, also eine innere Dichtung und ein Isolator je Bauteilhälfte, vorgesehen. Da zwei Riegel vorhanden sind, werden die vier Vektoren im „Path Mapper“ verdoppelt und jeweils in einem eigenen Ast gespeichert. So erhält der Pfad {2;0;0;0} die vier Verschiebungsvektoren für das Riegelprofil {2}. Sollte keine Positionierung von Nöten sein, erfolgt in der „Multiplication“-Komponente eine Multiplikation mit dem Wert 0. Dies ist der Fall, wenn ein Standardprofil verwendet wird. So erzeugen die Vektoren 98.3 keine Verschiebung.

Die Verschiebungsvektoren 98.3 werden mit der „Graft“-Komponente gegraftet, sodass jeder Vektor einen eigenen Ast erhält. Dies ist in Abbildung 99 dargestellt. Anschließend werden die Vektoren gefiltert. Dazu wird die Maske durch das Python-Skript Q21 „Script: Mas-

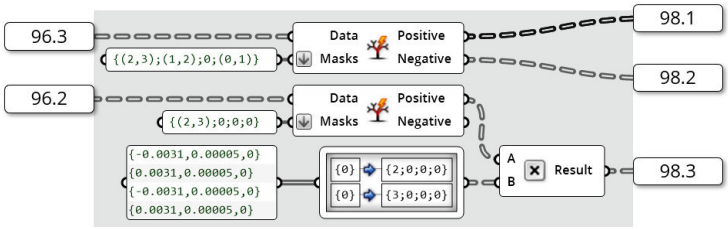


Abbildung 98: Korrektur der Position der inneren Dichtungen und der Isolatoren (1)
Quelle: In Anlehnung an Grasshopper

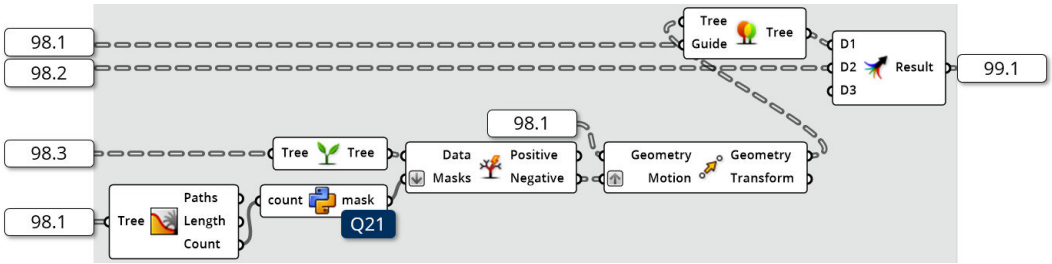


Abbildung 99: Korrektur der Position der inneren Dichtungen und der Isolatoren (2)
Quelle: In Anlehnung an Grasshopper

keVerschiebungsVektoren“ generiert. Das Skript erhält durch eine „Tree Statistics“-Komponente die Anzahl der Pfade der zu korrigierenden Bauteilkomponenten 98.1 und passt mit der entsprechenden Maske die Datenstruktur der Vektoren an die Daten der zu verschiebenden Geometrie an. Dabei werden nicht benötigte Pfade als „Positive“ ausgegeben, sodass „Negative“ für die „Move“-Komponente verwendet wird. Des Weiteren bekommt die Komponente die zu verschiebenden Geometrien 98.1. Die korrigierten Dichtungen und Isolatoren werden abschließend mithilfe der „Match Tree“-Komponente in die Datenstruktur gebracht, in der sie vor der Verschiebung 98.1 vorlagen und mit den nicht zu modifizierenden Geometrien 98.2 vereint. Das Ergebnis 99.1 ist in Abbildung 100 dargestellt. Darin ist erkennbar, dass die Positionen der Isolatoren und der inneren Dichtungen korrigiert wurden.

Nachdem die Positionen aller Bauteilkomponenten korrigiert sind, muss vor dem Aufdrehvorgang die Datenstruktur der Geometrien 99.1 angepasst werden. Dieser Prozess ist in Abbildung 101 gegeben. Die drei gezeigten Panels verdeutlichen die Datenstruktur vor der Umstrukturierung (links), nach der Umstrukturierung (mittig) und nach dem Gruppieren (rechts). Das Ziel der Umstrukturierung ist eine bauteilhälftenorientierte Datenstruktur. Bei dieser ist der Fokus auf die Unterscheidung der verschiedenen Bauteilhälften gesetzt.

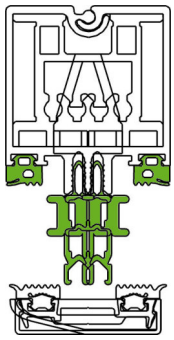


Abbildung 100: Korrigierte innere Dichtungen und Isolatoren der variablen Profile
Quelle: In Anlehnung an Rhinoceros 6

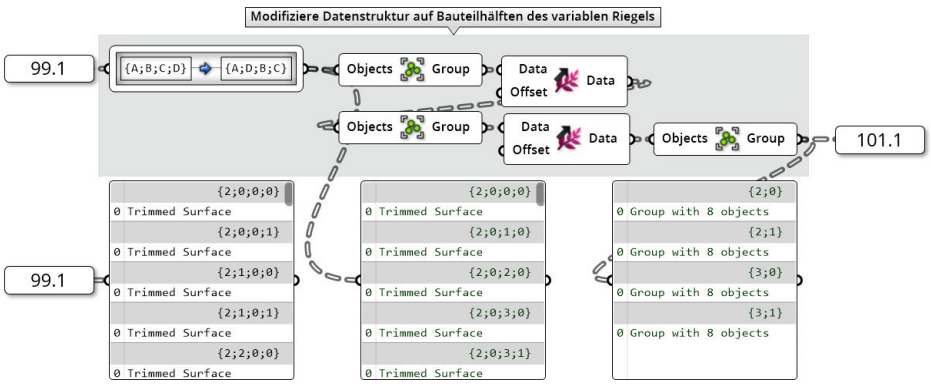


Abbildung 101: Von der bauteilorientierten zur bauteilhälftenorientierten Datenstruktur
Quelle: In Anlehnung an Grasshopper

Der Prozess kann dabei vollständig mit dem „Path Mapper“ erfolgen. Hierbei muss lediglich ein Mapping von {X;Y;Z;A}[i] nach {X;A;Y;Z}[i] erfolgen. Die Bedeutung der einzelnen Variablen ist dem Kapitel 3.4.2.2 zu entnehmen. Um die Geometriedaten für die Rotation weiter zu vereinfachen, wird neben der Umstrukturierung zusätzlich eine Gruppierung vorgenommen. Dies erfolgt mithilfe der „Group“-Komponenten. Hierbei werden alle Elemente in der Liste zu einer Gruppe zusammengefasst. Mit der darauffolgenden „Shift Paths“-Komponente kann eine Hierarchieebene entfernt werden, sodass mehrere Gruppen in einer Liste stehen. Dieser Gruppiervorgang wird entsprechend der Abbildung 101 wiederholt, bis die im rechten Panel gegebene Datenstruktur erreicht ist. Diese Gruppierung hat den Vorteil, dass nun nur noch vier Elemente vorliegen. Die Datenstruktur entspricht dabei {X;A}, sodass über X die beiden Bauteile und über A die beiden Bauteilhälften angesprochen werden können. Die gruppierte Datenstruktur wird in 101.1 weitergegeben.

Zu diesem Zeitpunkt kann, wie in Abbildung 102 ersichtlich, mit dem Öffnen der beiden Profilhälften begonnen werden. Dazu werden die Öffnungswinkel 97.1 für beide Profilhälften getrennt. Der Winkel für die Bauteilhälfte {X;0} wird mit dem Wert -1 multipliziert, sodass sich beide Profilhälften entgegengesetzt drehen. Das Ergebnis wird mit den Winkeln für die zweite Bauteilhälfte zusammengefasst und als „Angle“ für die „Rotate Axis“-Komponente benutzt. Dabei ist zu beachten, dass die Winkelangaben innerhalb der Komponente von Rad in Grad umgestellt werden. Die zu rotierenden Bauteilhälften entsprechen der zuvor gruppierten Datenstruktur 101.1. Das Ergebnis der geöffneten variablen Riegelprofile 102.1 ist in Abbildung 103 abgebildet. Wie zu erkennen ist, sind die Kompo-

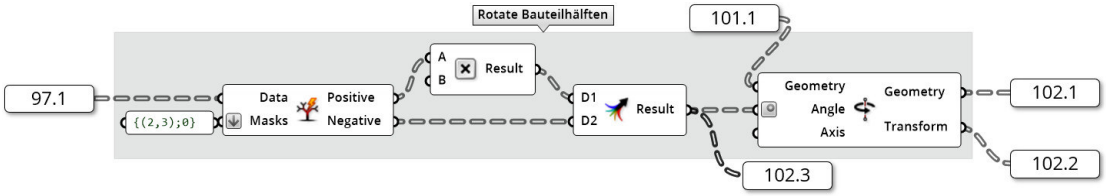


Abbildung 102: Rotation der variablen Riegelprofilhälften
Quelle: In Anlehnung an Grasshopper

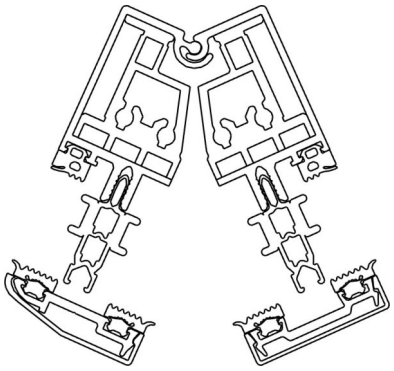


Abbildung 103: Korrigierte innere Dichtungen und Isolatoren der variablen Profile
Quelle: Screenshot aus Rhinoceros 6

nenten des variablen Riegelprofils richtig positioniert und die beiden Profilhälften konnten erfolgreich geöffnet werden.

Nachdem die Profile geöffnet wurden, können die Profile von der bauteilhälftenorientierten zurück in die bauteilorientierte Datenstruktur überführt werden. Dieser Prozess ist in Abbildung 104 dargestellt. Hierbei erfolgt der umgekehrte Prozess zu Abbildung 101. Mithilfe der „Ungroup"-Komponenten werden die Gruppen geöffnet und mithilfe der „Graft"-Komponente werden die zuvor entfernten Hierarchieebenen wiederhergestellt. Nachdem die Gruppen entfernt wurden, kann mit dem „Path Mapper" das vorherige Mapping wieder rückgängig gemacht werden, sodass die Daten in der bauteilorientierten Datenstruktur vorliegen.

Folgend soll in diesem Cluster ein weiteres Problem gelöst werden. Wie in Abbildung 103 ersichtlich ist, befindet sich derzeit eine Lücke zwischen den beiden Profilhälften. Im Folgenden soll ein Sandwichpaneel modelliert und in den Zwischenraum positioniert werden. Die Geometrie des Paneels ist maßgeblich vom gewählten Isolator abhängig. Wie in Abbildung 105 ersichtlich ist, wird dazu die Spaltennummer des Isolators „S_Isolator" als „g" 40.3 in das Cluster geladen. Da nur eine Elementdicke für den gesamten parametrischen Fassadenknoten gewählt werden kann, wird nur ein „S_Isolator" benötigt. Dazu wird mithilfe der „Tree Explode"-Komponente ein Pfad eines Riegels ausgewählt und weitergegeben. Mit diesem Parameter können im Python-Skript Q22 „Skript: DickeVarMitten" die Dicken der Paneele bestimmt werden. Hierfür folgt das Skript der Formel:

$$PaneelDicke = \begin{cases} -0.0375 & \text{für } S_{Isolator} = 0 \\ -0.0375 - 0.004 & \text{für } S_{Isolator} = 1 \\ -0.0375 - 0.004 - 0.006 * (S_{Isolator} - 1) & \text{für } S_{Isolator} \geq 1 \end{cases} \tag{5.5}$$

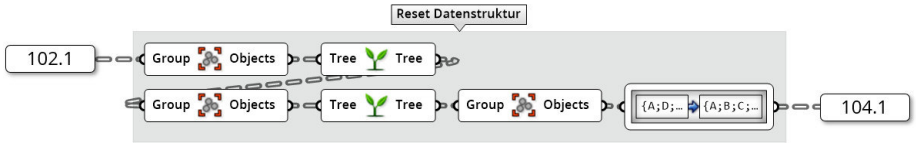


Abbildung 104: Zurücksetzen der Datenstruktur
Quelle: In Anlehnung an Grasshopper

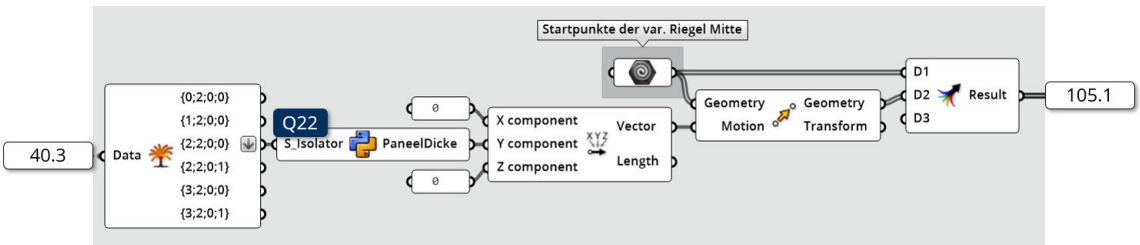


Abbildung 105: Dicke der Mittenblende der variablen Riegelprofile
Quelle: In Anlehnung an Grasshopper

Das Ergebnis der Formel wird genutzt, um zwei Punkte, die in der „Geometry“-Komponente gespeichert sind, mit der „Move“-Komponente zu verschieben. Da der gesamte Zwischenraum ausgefüllt werden soll, werden beide Geometrien, die unverschobene und die verschobene, als 105.1 zusammengeführt.

Wie in Abbildung 106 dargestellt, können anschließend die vier entstandenen Punkte 105.1 mit der „Mirror“-Komponente gespiegelt werden. Da die Spiegelung die Mittenverblendung der zweiten Bauteilhälfte ausdrückt, werden die Pfade der Geometrien angepasst und zusammen geführt. Hierfür werden zwei „Path Mapper“ und eine „Merge“-Komponente verwendet. In der bauteilorientierten Datenstruktur sind, wie in Abbildung 107 verdeutlicht, die Pfade {(2,3);7;0;(0,1)} vorgesehen. Es werden acht Eckpunkte, je vier pro Profilhälfte, für die Mittenblenden erzeugt 106.1. Um die Mittenblenden entsprechend der Öffnung der variablen Riegelprofile zu verdrehen, wird die „Transform“-Komponente mit den „Transform“-Informationen 102.2 aus der „Rotate Axis“-Komponente aus Abbildung 102 verwendet. Somit wird auf die acht Eckpunkte der Mittenblenden 106.2 die gleiche Rotation wie auf die beiden Riegelprofilhälften ausgeübt.

Da nun die Eckpunkte des Paneels innerhalb der Profile erzeugt wurden, müssen die Paneele beider Profilhälften miteinander verschnitten werden. Wie in Abbildung 108 gezeigt, werden die Punkte eines Paneels 106.1 mit der „Tree Split“-Komponente herausgefiltert. Die Punkte werden mit der „Deconstruct“-Komponente in die einzelnen Koordinaten zerlegt. Um die Punkte der Schnittebene zu erhalten, können die z-Koordinaten übernommen

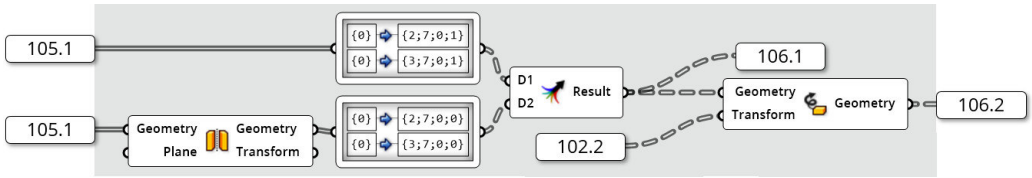


Abbildung 106: Erzeugung der Mittenblenden der variablen Riegelprofile
Quelle: In Anlehnung an Grasshopper

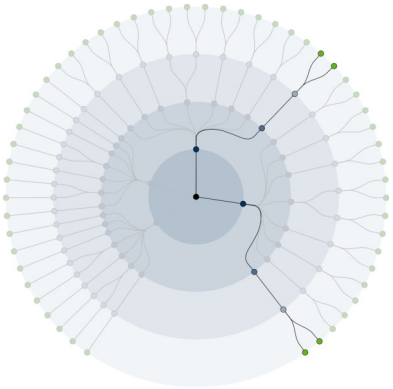


Abbildung 107: Datenstruktur der Mittenblenden
Quelle: Eigene Darstellung

werden. Die x-Koordinaten werden auf den Wert 0 gesetzt. Die y-Koordinaten der Punkte „$y_{Punkt'}$“ müssen neu zu „y_{Punkt}“ berechnet werden. Die Berechnung erfolgt in Abbildung 109 und 110 nach folgender Formel:

$$y_{Punkt} = y_{Rotation} - \left[(y_{Rotation} - y_{Punkt'}) * \left(\frac{1}{cos\left(\frac{W_0 - W_1}{2}\right)} \right) \right] \tag{5.6}$$

Dazu wird der halbe Öffnungswinkel der variablen Riegelprofile und die y-Koordinate des Rotationspunktes „$y_{Rotation}$“ 108.2 benötigt. Da die Öffnungswinkel 108.1 für jede Bauteilhälfte entgegengesetzt definiert sind, müssen die beiden Winkel W_0 und W_1 per Subtraktion zusammengerechnet werden. Anschließend kann der Winkel halbiert werden. Die weiteren Berechnungsschritte können der Formel 5.6 und den Abbildung 109 und 110 entnommen werden. Neben der Berechnung werden dabei Datenstrukturmanagementoperationen ausgeführt. So können die korrigierten y_{Punkt}-Koordinaten 110.1 für die Schnittpunkte 108.4 berechnet werden.

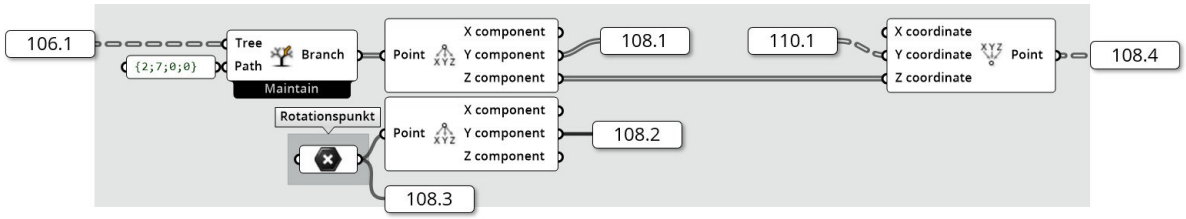


Abbildung 108: Erzeugung der Schnittpunkte der Mittenblenden der variablen Riegel (1)
Quelle: In Anlehnung an Grasshopper

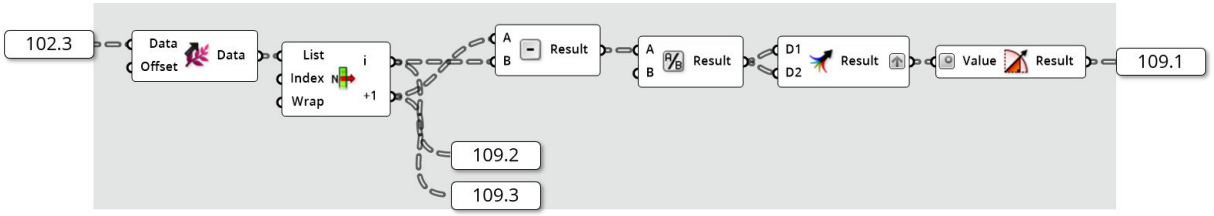


Abbildung 109: Erzeugung der Schnittpunkte der Mittenblenden der variablen Riegel (2)
Quelle: In Anlehnung an Grasshopper

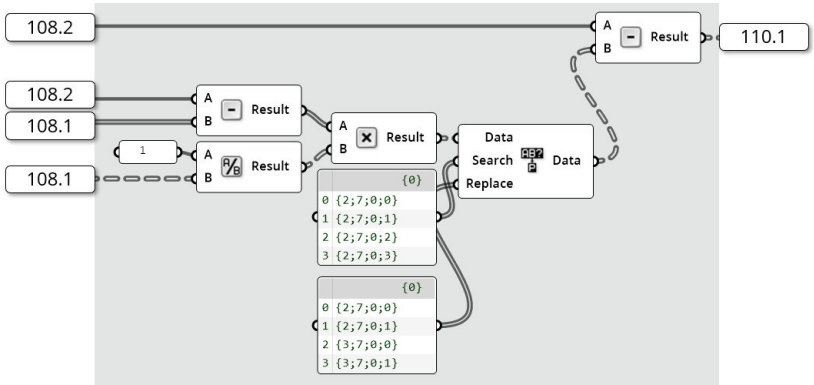


Abbildung 110: Erzeugung der Schnittpunkte der Mittenblenden der variablen Riegel (3)
Quelle: In Anlehnung an Grasshopper

Wie in Abbildung 111 dargestellt, muss für ungleich geöffnete variable Riegelprofile die Schnittebenengeometrie um den halben Differenzwinkel der beiden Öffnungswinkel gedreht werden. Die Differenz der beiden Winkel 109.2 und 109.3 wird aufgrund ihrer Gegenläufigkeit durch Addition erzeugt. Die vier Schnittpunkte 108.4 werden mit der „Rotate Axis“-Komponente entsprechend rotiert. Abschließend können die vier Punkte der Schnittebene und die acht Eckpunkt der Mittenblenden 106.2 zur Querschnittsgeometrie verknüpft werden. Dazu werden die Eckpunkte und die Schnittpunkte mit den „4Point Surface“-Komponenten zu Surfaces 111.1 verbunden.

Nach dem Zusammenführen der Geometrien, wird in den beiden Python-Skripten Q23 „Script: FilterMittenVarRiegel“ überprüft, ob die Mitten der variablen Riegel benötigt werden. Dieser Vorgang ist in Abbildung 112 dargestellt. Werden die Mittenblenden nicht benötigt, werden entsprechende Masken 112.1 und 112.2 durch die Skripte erzeugt. Im Anhang Q23 sind die Skripte gegeben. Die Skripte benötigen den gesamten Öffnungswinkel. Dieser wird, wie oben beschrieben, mit einer Subtraktion gebildet.
Zum Ende des „Cluster: VarRiegelAufdrehen“ werden alle erzeugten Informationen zusammengeführt. Dieser Vorgang ist in Abbildung 113 dargestellt. Die Bauteilgeometrie der geöffneten variablen Riegel 104.1, alle nicht modifizierten Bauteilgeometrien 96.4 und die neu erzeugten Mittenblenden 111.1 werden zusammengeführt. Danach wird mit der „Tree Split“-Komponente der Filtervorgang mit den zuvor erzeugten Masken 112.1 und 112.2 angewendet, um die Mittenblenden bei zu geringen Öffnungswinkeln zu deaktivie-

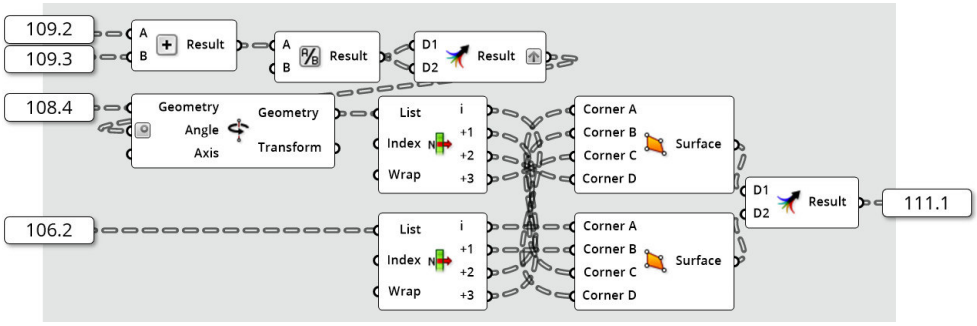


Abbildung 111: Erzeugung der Schnittpunkte der Mittenblenden der variablen Riegel (4)
Quelle: In Anlehnung an Grasshopper

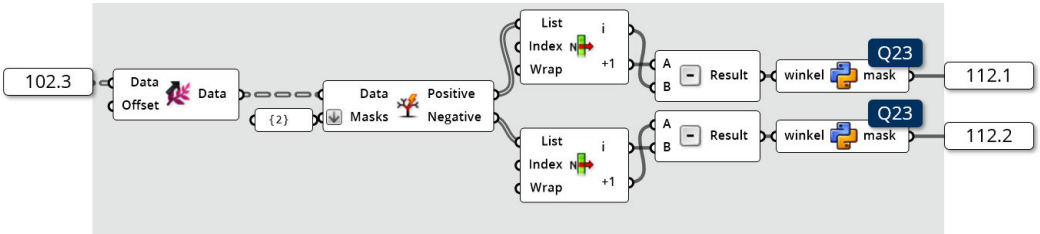


Abbildung 112: Erzeugung der Schnittpunkte der Mittenblenden der variablen Riegel (5)
Quelle: In Anlehnung an Grasshopper

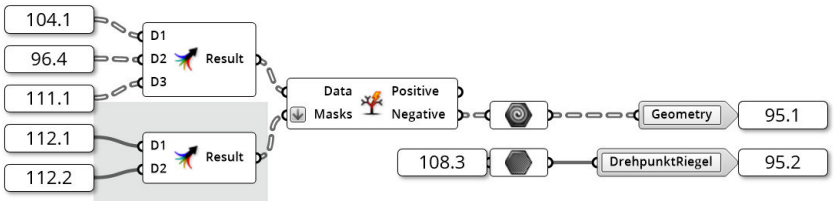


Abbildung 113: Output des „Cluster: VarRiegelAufdrehen“
Quelle: In Anlehnung an Grasshopper

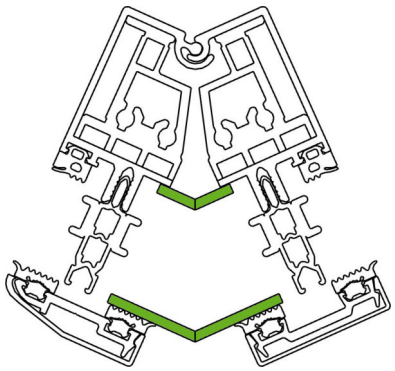


Abbildung 114: Um Mittenblende ergänztes variables Riegelprofil
Quelle: In Anlehnung an Rhinoceros 6

ren. Das Ergebnis wird als „Geometry“ 95.1 und der Rotationspunkt 108.3 als „DrehpunktRiegel“ 95.2 aus dem Cluster zurückgegeben. In Abbildung 114 ist die durch das Cluster verbesserte Bauteilgeometrie dargestellt.

5.2.5 Mittelpunkt der Querschnittsgeometrie korrigieren

Mithilfe des „Cluster: GeometrieZurScheibenmitte“ werden die Profilquerschnitte so verschoben, dass der Mittelpunkt der Geometrien in der Mitte des SZR liegt. Dies hat den Vorteil, dass so Verschiebungen aufgrund der späteren Rotation der Profile minimiert werden können.

Die Mitte des SZR ist von der gewählten Elementdicke „x“ 26.8 abhängig. Die Elementdicke „x“ liegt dabei nicht als Länge, sondern als Index vor. Wie in Abbildung 115 ersichtlich, kann die Elementdicke „x“ mithilfe des Skripts Q24 „Script: Verschiebung Geometriemitte“ zu einer Verschiebung entlang der y-Achse umgerechnet werden. Das Skript ist in Anhang Q24 beigefügt. Die Strecke der Verschiebung „a“ wird im Skript nach folgender Formel bestimmt:

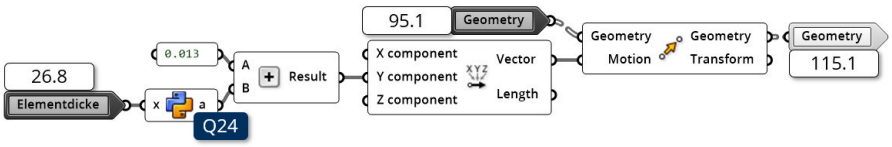


Abbildung 115: Korrektur des Profilmittelpunktes
Quelle: In Anlehnung an Grasshopper

$$a = \begin{cases} \frac{12-x}{1000} & \text{für x} < 2 \\ \left(28 + int\left(\frac{x-2}{3}\right) * 6\right) - \left(28 + int\left(\frac{x-2}{3}\right) * 6 + 2 * ((x-2)\, mod 3)\right) : 2000 & \text{für x} \geq 2 \end{cases} \quad (5.7)$$

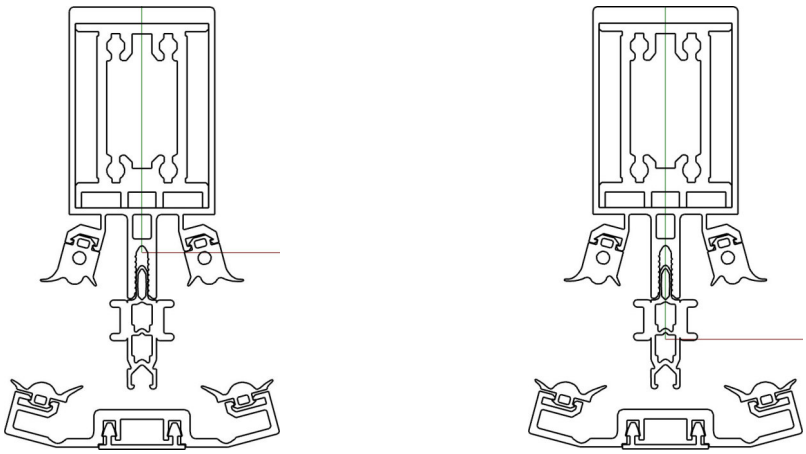
Das Ergebnis der Formel 5.7 wird mit 0.013 addiert und als y-Koordinate eines Verschiebungsvektors in der „Move“-Komponente genutzt, sodass in 115.1 die korrigierte Geometrie aus dem Cluster ausgegeben werden kann. In Abbildung 116 ist der Unterschied durch das Cluster dargestellt.

5.2.6 Ausrichtung und Position der Press- und Blendleisten der Riegelprofile

Bei der Erzeugung eines parametrischen Fassadenknotens mit den in Abbildung 116 (a) dargestellten Querschnitten treten zwei Problemfelder auf. Diese sind:

1. die Riegelprofile befinden sich bei zunehmender Riegelöffnung zu weit vorne
2. die abgeflachten Press- und Blendleisten zeigen nach unten

Um diese beiden Probleme zu beheben, wurde das „Cluster: Optimiere Riegelprofile“ in die Grasshopper-Definition implementiert. Das in Abbildung 117 gegebene Cluster benutzt



(a) (b)

Abbildung 116: Vor (a) und nach (b) dem „Cluster: GeometrieZurScheibenmitte“
Quelle: Screenshot aus Rhinoceros 6

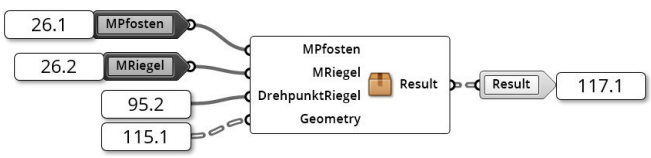


Abbildung 117: Das „Cluster: Optimiere Riegelprofile“
Quelle: In Anlehnung an Grasshopper

dazu zum einen die Geometrie der Bauteilkomponenten 115.1. Des Weiteren werden die beiden Winkel „MPfosten“ 26.1 und „MRiegel“ 26.2 und zusätzlich der Drehpunkt der variablen Riegelprofile 95.2 benötigt.

Das erste der beiden Probleme entsteht dadurch, dass die Rotation der beiden Pfostenprofile um den in Abbildung 116 (a) gezeigten Punkt stattfindet. Die beiden Profilhälften der variablen Riegelprofile drehen sich zwar mit den beiden Pfosten mit, jedoch um einen deutlich weiter hinten liegenden Punkt. Daraus resultiert eine relative Verschiebung der Profile zueinander. Aus diesem Grund werden die Riegelprofile auf Grundlage der Verdrehung der Pfostenprofile entlang der y-Achse verschoben. Die Berechnung der Verschiebung „v“ erfolgt dabei über die folgende Formel:

$$v = \left[1.625 * \left(\frac{1}{cos(M_{Pfosten})} - 1\right) * y_{Rotation}\right] + \left[-0.1 * \left(\frac{1}{cos(M_{Riegel})} - 1\right) * y_{Rotation}\right] \quad (5.8)$$

Die Umsetzung der Formel 5.8 ist in Abbildung 118 dargestellt. Die beiden Korrekturfaktoren 1.625 und -0.1 wurden dabei empirisch ermittelt. Mithilfe der Formel werden die Riegelprofile in Abbildung 119 mit dem resultierenden Vektor 118.1 entsprechend ihres Öffnungswinkels verschoben, sodass die Position der Riegel innerhalb des Knotens optimiert wurde.

Das zweite Problem kann schneller als das erste behoben werden. Hier werden die beiden Riegelprofile gespiegelt. Dieser Vorgang kann neben der Verschiebung der Riegelprofile in Abbildung 119 betrachtet werden. So werden beide Probleme in diesem Cluster behoben. Die Geometrie wird in 117.1 aus dem „Cluster: Riegel Optimieren“ und anschließend in 84.1 aus dem „Cluster: KorrigiereQuerschnitt“ gegeben. Zu diesem Zeitpunkt sind die

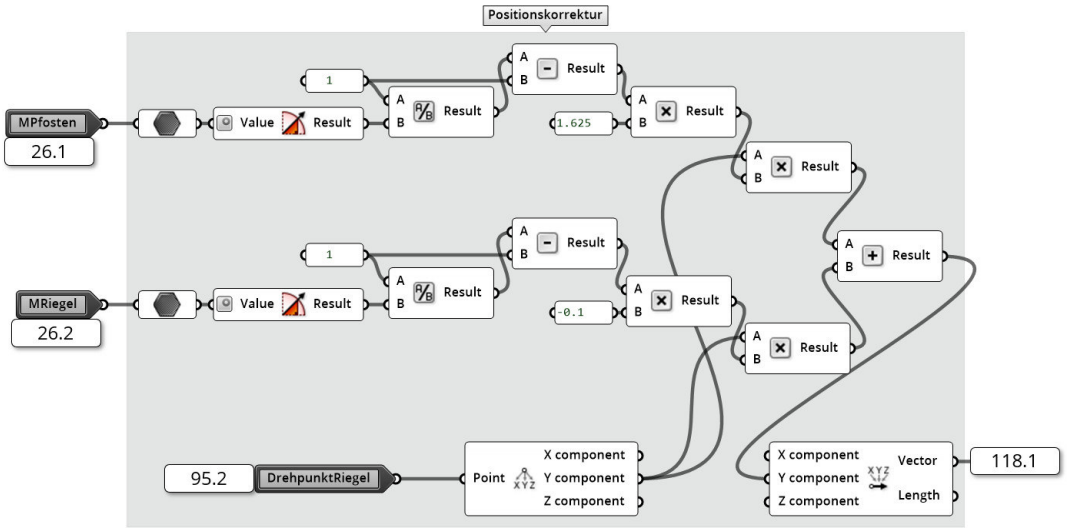


Abbildung 118: Berechnung der Verschiebung der Riegelprofile
Quelle: In Anlehnung an Grasshopper

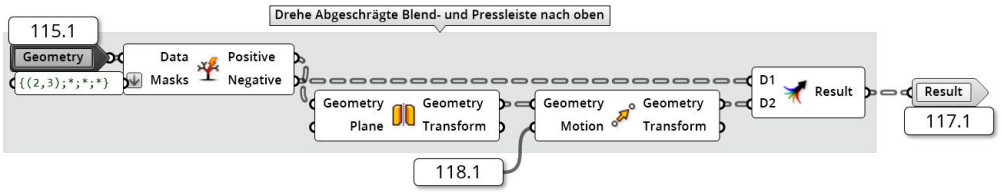


Abbildung 119: Berechnung der Verschiebung der Riegelprofile
Quelle: In Anlehnung an Grasshopper

Profilquerschnitte soweit vorbereitet, dass sie zur Erzeugung eines parametrischen Fassadenknotens genutzt werden können.

6 Erzeugung der Knotengeometrie

6.1 Positionierung der Profile

Nachdem gezeigt wurde, wie die Querschnittsprofile zusammengesetzt sind, wird in diesem Kapitel damit begonnen, die Querschnitte zur Generierung eines dreidimensionalen Fassadenknotens zu nutzen. Damit ein dreidimensionaler Fassadenknoten erzeugt werden kann, werden die Vektoren der Profilrichtungen 29.4, die Geometrie der Profilquerschnitte 84.2, die Länge der Profile 26.9 und die fünf vom Nutzer einstellbaren Winkel MPfosten 26.1, MRiegel 26.2, SPfosten 26.3, SRiegelR 26.4 und SRiegelL 26.5 benötigt. Um den Fassadenknoten zu erzeugen, wurde das „Cluster: Erzeuge Knotengeometrie“, das in Abbildung 120 dargestellt ist, implementiert. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/08_ClusterKnotengeometrieKomplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Das Innere des Clusters weist fünf weitere untergeordnete Cluster auf. Die dabei enthaltenen Cluster sind:

1. Cluster: Referenzprofile erzeugen
2. Cluster: Erzeuge Boxen und Referenzgeometrien
3. Cluster: Positioniere Referenzboxen
4. Cluster: PositioniereEinschub
5. Cluster: ErzeugeKnotengeometrie

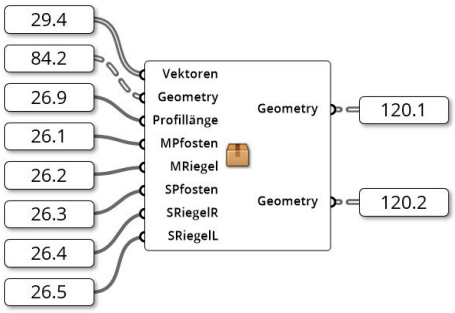


Abbildung 120: Das „Cluster: Erzeuge Knotengeometrie“
Quelle: In Anlehnung an Grasshopper

6.1.1 Erzeugung von Referenzprofilen

Um die Rechenleistung beim Verschieben der hochdetaillierten Fassadenprofile zu minimieren, wird der Ansatz verfolgt, das LoD zu reduzieren. Aus diesem Grund werden zunächst Referenzprofile erzeugt, die eine Boundary Box erhalten. Die Boundary Box wird anschließend modifiziert. Nach Abschluss der Transformation der Box wird die Referenzgeometrie von der Referenz Boundary Box auf die geänderte Box abgebildet, sodass die Transformationen der Box auf die Fassadenprofile ausgeübt werden. In diesem Kapitel werden zunächst die Referenzgeometrien in Form von Fassadenprofilen erzeugt. Die Erzeugung erfolgt, indem die Querschnitte der Profile in eine bestimmte Richtung extrudiert werden. Aus diesem Anlass wird das „Cluster: Referenzprofile erzeugen“ implementiert. Wie in Abbildung 121 zu sehen ist, benötigt das Cluster die Querschnittsgeometrien der einzelnen Profile 84.2 und deren Länge 26.9. In der digitalen Anlage D2 „Schubert02_Abbildungen/01_Cluster/07_ClusterReferrenzprofileKomplett.jpg“ ist die komplette Darstellung des Clusterinneren gegeben. Der Ausschnitt des Inneren von „Cluster: Referenzprofile erzeugen“ in Abbildung 122 zeigt, dass zunächst die Querschnittsgeometrien 84.2 in die einzelnen vier Profile 122.1 - 122.4 zerlegt werden. Die erste Zerteilung mit der Maske {(0,3);*;*;*} separiert alle Profile, die in eine andere Richtung extrudiert werden müssen. Die unterschiedlichen Extrudierungsrichtungen resultieren daraus, dass gegenüberliegende Profile gespiegelt sind. Anstelle einer Spiegelung der Profilquerschnitte wird dies durch entgegengesetzte Extrudierungsrichtungen realisiert. Im zweiten Teilungsschritt werden die Pfosten und Riegel getrennt, sodass alle Profile einzeln vorliegen.

In Abbildung 123 ist dargestellt, wie sich die einzelnen Extrudierungsvektoren zusammensetzen. Da sich die Querschnitte in der x-y-Ebene befinden, wird ausschließlich in z-Richtung extrudiert. Die hierfür benötigte Länge ist durch 26.9 vom Nutzer definiert. Des Weiteren erhalten die gegenüberliegenden Profile jeweils das entgegengesetzte Vorzeichen. Anschließend können die gebildeten Vektoren genutzt werden, um mithilfe der „Extrude“-

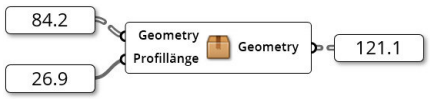


Abbildung 121: Das „Cluster: Referenzprofile erzeugen“
Quelle: In Anlehnung an Grasshopper

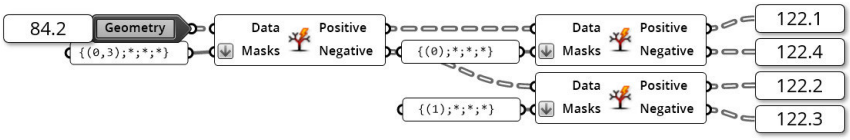


Abbildung 122: Zerteilen der Datenstruktur der Querschnittsgeometrien
Quelle: In Anlehnung an Grasshopper

Komponente dreidimensionale Fassadenprofile zu erzeugen. Die einzelnen Profile werden zu einer Datenstruktur zusammengefasst und in „Geometry“ 121.1 aus dem Cluster ausgegeben.

In Abbildung 124 (a) ist ein durch das Cluster extrudiertes Fassadenprofil dargestellt. Dieses ist von einem Pfosten. Die so erzeugten Profile dienen folgend als Referenzgeometrien und werden mithilfe von repräsentierenden Boxen transformiert.

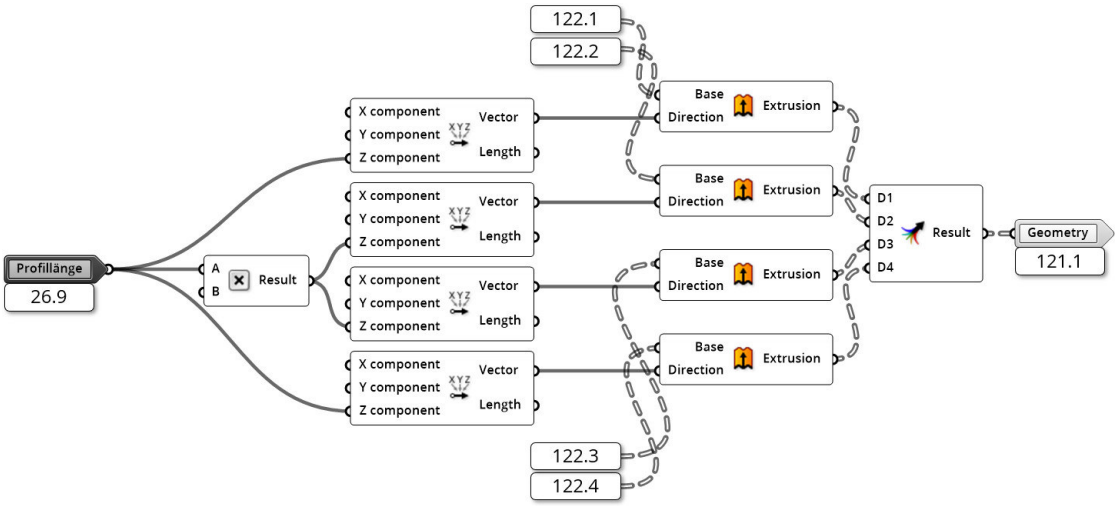
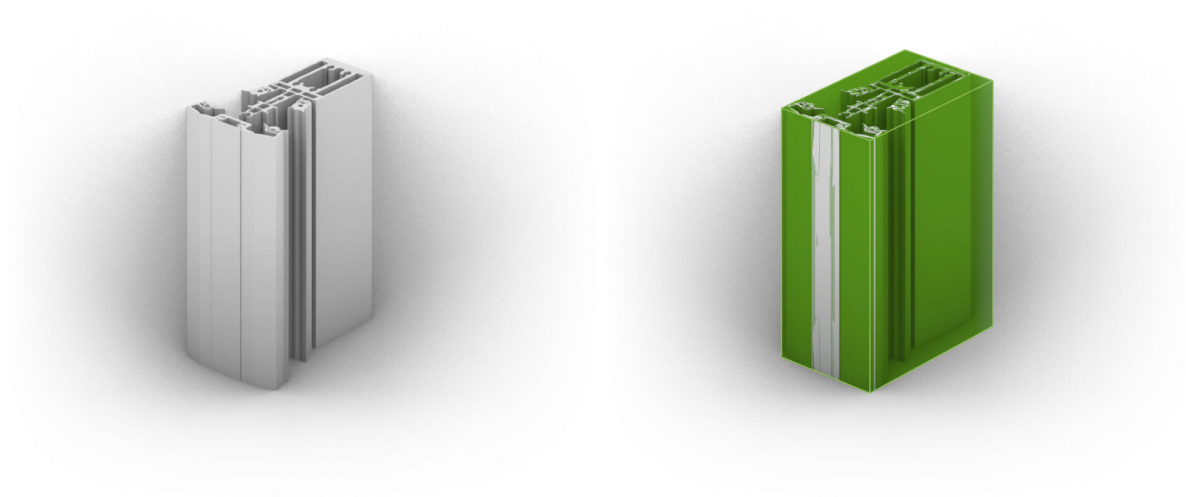


Abbildung 123: Extrudieren der Profilquerschnitte
Quelle: In Anlehnung an Grasshopper



(a) (b)

Abbildung 124: Referenzprofil des Pfostens (a) und zugehörige Referenzbox (b)
Quelle: Screenshot aus Rhinoceros 6

6.1.2 Erzeuge Referenzboxen

Um die LoD verringern zu können, müssen die Profile in Boxen umgewandelt werden. Eine Box kann als einfacher Quader verstanden werden. Der Sinn einer Referenzbox ist, eine Geometrie zu repräsentieren. Abbildung 124 (b) zeigt neben dem extrudierten Profil diese Box. Für weitere Transformationen der Profile finden anschließend die vereinfachten Geometrien Verwendung. Um diese Quader zu erzeugen, wurde das „Cluster: Erzeuge Boxen und Referenzgeometrien“ implementiert.

Das Cluster ist in Abbildung 125 dargestellt. Zur Erzeugung der Referenzboxen werden einerseits die Referenzprofile 121.1 und andererseits die Profillängen 26.9 benötigt. Als Output kann mit diesem Cluster eine Geometrie für die Einschubprofile „GeometryBox“ 125.1, die dazugehörige Referenzbox „BoxEinschub“ 125.2, die Geometrie der restlichen Bauteilkomponenten „Geometry“ 125.3 inklusive Referenzboxen „Box“ 125.4 und ein Korrekturfaktor 125.5 erzeugt werden.

Innerhalb des Clusters werden, wie in Abbildung 126 zu sehen, die Einschubprofile vom Rest der Geometrie getrennt. Da die Einschubprofile an siebter Stelle der Bauteilkomponenten stehen, wird die „Split Tree“-Komponente mit der Maske {(0 to 3);6;*;*} verwendet. Anschließend können beide Datenbäume soweit gruppiert werden, dass nur noch die einzelnen Bauteile vorhanden sind. Für den Gruppiervorgang wird ein Cluster verwendet. Die Funktionsweise des Clusters entspricht dem Gruppiervorgang aus Abbildung 101 (zu finden auf Seite 104). Die Bauteilkomponenten ohne Einschubprofile können anschließend direkt in Referenzboxen umgewandelt werden. Hierfür steht die „Bounding Box“-Komponente zur Verfügung. Die Komponente erzeugt eine kleinstmögliche Box um eine

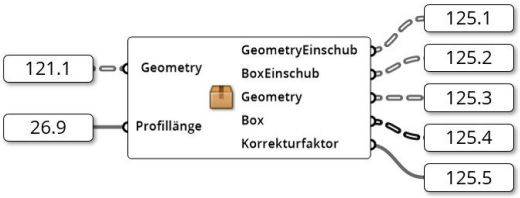


Abbildung 125: Das „Cluster: Erzeuge Boxen und Referenzgeometrien“
Quelle: In Anlehnung an Grasshopper

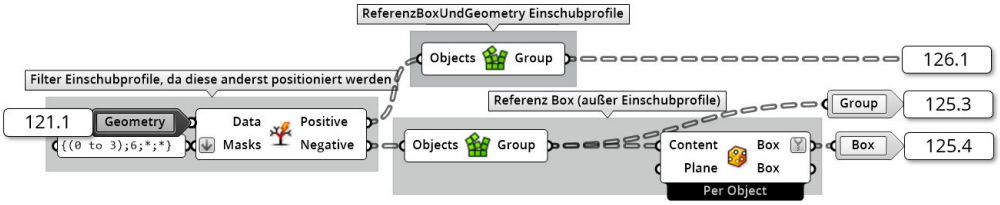


Abbildung 126: Erzeugung der Referenzgeometrien und -boxen außer Einschubprofile
Quelle: In Anlehnung an Grasshopper

gegebene Geometrie, die in diesem Fall orthogonal zum Weltkoordinatensystem ist. Sollten Abweichungen von diesem Koordinatensystem notwendig sein, ist eine andere Ebene als „Plane“ an die Komponente zu übergeben und der untere „Box“-Output ist zu verwenden. Die Referenzgeometrie der Bauteilkomponenten ohne Einschubprofile wird als „Geometry“ 125.3 und die dazugehörigen Referenzboxen als „Box“ 125.4 aus dem Cluster ausgegeben.

Für die Einschubprofile ist ein Zwischenschritt von Nöten. Wie Abbildung 127 verdeutlicht, wird die Geometrie des Einschubprofils zunächst auf die entsprechende Länge skaliert. Aufgrund des Extrahierungsvorgangs mit der Profillänge l_{Profil} (vgl. Kapitel 6.1.1) gilt für die Länge des Einschubprofils $l_{Einschubprofil}$ derzeit:

$$l_{Einschubprofil} = l_{Profil} \tag{6.1}$$

Aus diesem Grund kann eine neue Länge l_{neu} mit dieser Formel definiert werden:

$$l_{Einschubprofil} = \frac{1}{l_{Profil}} * l_{neu} \tag{6.2}$$

Nachdem mit der Formel 6.2 und der „Scale NU“-Komponente die Länge des Einschubprofils eingestellt wurde, werden die Einschubprofile in die Grundstellung gebracht. Dazu werden sie um die y-Achse mit der „Rotate Axis“-Komponente rotiert. Die dafür benötigten Winkel und Achsen sind direkt in der Komponente gespeichert. So entsteht die rotierte Referenzgeometrie der Einschubprofile. Zum einen wird sie als „GeometryEinschub“ 125.1 aus dem Cluster ausgegeben. Andererseits werden auch für die Einschubprofile die Referenzbox mit der „Bounding Box“-Komponente erzeugt und als „BoxEinschub“ 125.2 ausgegeben.

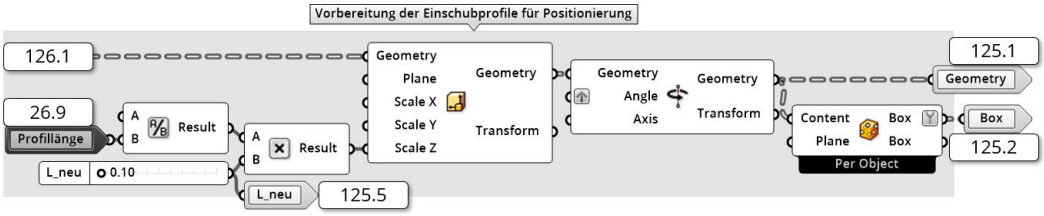


Abbildung 127: Erzeugung der Referenzgeometrien und -boxen der Einschubprofile
Quelle: In Anlehnung an Grasshopper

6.1.3 Positioniere Referenzboxen

Nachdem die Referenzgeometrien und -boxen erzeugt wurden, kann mit der Erstellung der Geometrie des parametrischen Fassadenknotens begonnen werden. Dazu werden zunächst die Referenzboxen in die Position der späteren Fassadenprofile gebracht. Um diesen Vorgang innerhalb der Grasshopper-Definition umzusetzen, wurde das „Cluster: PositioniereReferenzboxen“ implementiert. Dieses in Abbildung 128 dargestellte Cluster soll mithilfe der fünf vom Nutzer definierten Winkel „MPfosten“ 26.1, „MRiegel“ 26.2, „SPfosten“ 26.3, „SRiegelR“ 26.4 und „SRiegelL“ 26.5 sowie der Geometrie der Referenzboxen 125.4 als „Geometry“ die gegebenen Boxen entsprechend der gegebenen Winkel positionieren. Aus dem Cluster werden die positionierten Boxen als „Geometry“ 128.1 und die Transformationsinformationen als „Transform“ 128.2 und 128.3 ausgegeben.

Wie in Abbildung 129 dargestellt, werden die Boxen 125.4 in die Grundstellung gebracht. Diese Grundstellung kann dabei mit den vier Startvektoren aus Kapitel 4.3 verglichen werden. Die hierfür benötigten Achsen und Winkel sind direkt in der ersten „Rotate Axis“-Komponente gespeichert. Anschließend werden alle Winkel 26.1 - 26.5 so zusammengefasst, dass sie zur Rotation der vier Boxen geeignet sind. Damit jede Box genau zweimal rotiert wird, werden für jede Rotation genau vier Winkel benötigt. Die Rotationen erfolgen

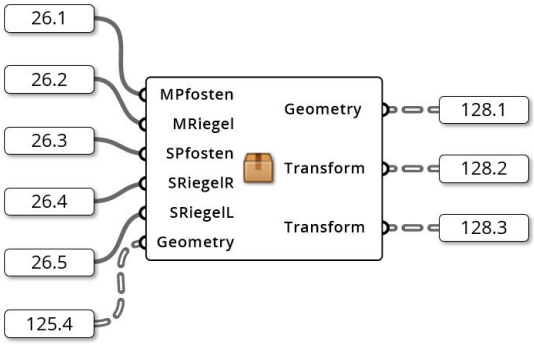


Abbildung 128: Das „Cluster: PositioniereReferenzboxen“
Quelle: In Anlehnung an Grasshopper

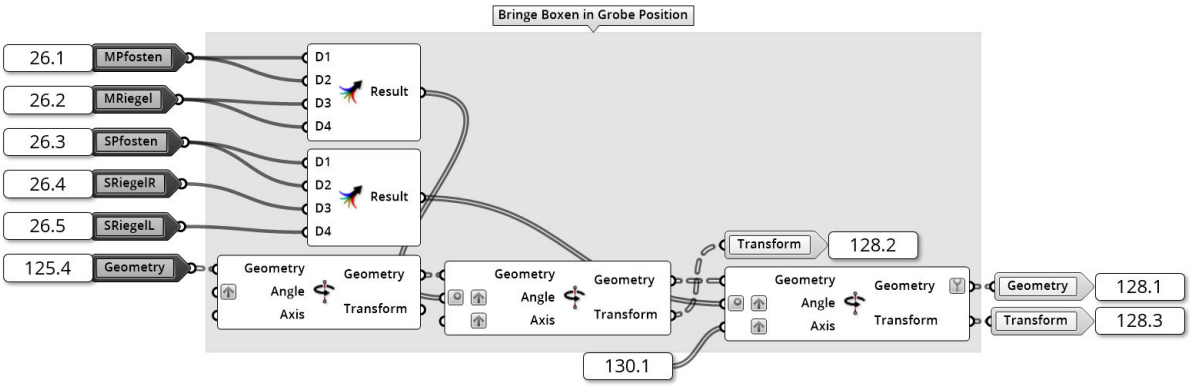


Abbildung 129: Rotiere Referenzboxen
Quelle: In Anlehnung an Grasshopper

nach demselben Schema wie die Rotation der Vektoren in Kapitel 4.3. Zusammengefasst erfolgt die erste Rotation um die x- beziehungsweise z- Achse um die Winkel „MPfosten“ beziehungsweise „MRiegel“ und die zweite Rotation um die y-Achsen. Die hierfür benötigten Achsen sind direkt in den jeweiligen „Rotate Axis“-Komponenten gespeichert. Die Transformationsinformationen aus den „Rotate Axis“-Komponenten der Rotation der Vektoren in Kapitel 4.3 kann hierfür nicht genutzt werden, da die Achsen der zweiten Rotation in 130.1 für die Rotation der dreidimensionalen Bauteile optimiert sind. Jedoch können die Transformationsinformationen aus diesen Rotationen weiter für die Einschubprofile genutzt werden. Dazu werden die „Transform“-Ausgänge der beiden Komponenten als „Transform“ 129.2 und 129.3 ausgegeben.

Wie bereits angesprochen, werden die Achsen der Rotation im Vergleich zur Vektorrotation aus Kapitel 4.3 an die dreidimensionale Ausprägung der Bauteile angepasst. In Abbildung 130 ist dieser Vorgang dargestellt. Hierfür bekommen alle Bauteile eine eigene Rotationsachse. Die Achsen der beiden Profile beginnen beide bei (0;0;0). Die beiden Achsen der Riegel beginnen jeweils an der äußeren Kante der Profile. Da die Profile in diesem Fassadensystem 5,00 cm breit sind, sind die Startpunkte (0.025;0;0) beziehungsweise (-0.025;0;0). Im Laufe der Arbeit hat es sich erprobt, die x-Koordinate der Startpunkte der Riegelachsen weiter zu optimieren. Dadurch wird die gesamte Achse um einen kleinen Betrag in x-Richtung verschoben, sodass das rotierte Profil in der z-Achse verschoben werden kann. Folgende Formel konnte dabei zur Optimierung genutzt werden:

$$x_{optimiert} = x + x * (-0.2 * cos(S_{Riegel})) \tag{6.3}$$

Nachfolgend werden die Startpunkte mit den in die jeweilige y-Richtung verschobenen Endpunkten zu Achsen umgewandelt und als 130.1 an die zweite Rotation als Achse gegeben.

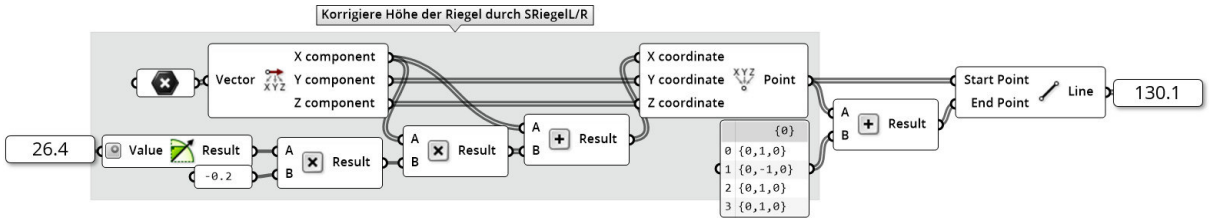


Abbildung 130: Optimiere Rotationsachsen
Quelle: In Anlehnung an Grasshopper

6.1.4 Positioniere Einschubprofile

Folgend soll erläutert werden, wie die Einschubprofile positioniert werden. Dafür wird das „Cluster: PositioniereEinschub“ implementiert. Dieses ist in Abbildung 131 gegeben. Innerhalb des Clusters werden zunächst die Referenzboxen der Einschubprofile 125.2 entsprechend der Rotation der restlichen Referenzboxen transformiert. Hierfür werden „Transform“-Komponenten mit den Transformationsinformationen 128.2 und 128.3 verwendet. Nachdem die Boxen rotiert sind, werden die Boxen entlang der Vektoren verschoben. Die Einschubprofile sollen soweit transformiert werden, dass sie zu 50 % aus den Profilen ragen. Die benötigte Verschiebung $\vec{m}$ kann mithilfe der Vektoren $\vec{v}$, der Profillänge l_{Profil} und der Länge der Einschubprofile $l_{Einschub}$ über folgende Formel bestimmt werden:

$$\vec{m} = \left(l_{Profil} - \frac{l_{Einschub}}{2} \right) * \vec{v} \tag{6.4}$$

Die daraus resultierende Zielbox kann anschließend verwendet werden, um die Geometrie der Referenzprofile abzubilden. Dazu wird die „Box Morph“-Komponente verwendet. Die Geometrie der Referenzprofile 125.1 wird als „Geometry“ benutzt. „Target“ ist die eben erzeugte Zielbox und „Reference“ die Referenzbox der Einschubprofile 125.2. Die daraus resultierende Geometrie ist in Abbildung 132 dargestellt.

6.1.5 Erzeugung der Knotengeometrie

Nachdem die Referenzboxen der Geometrien positioniert sind, wird die Geometrie dieser Boxen durch die Geometrie der Fassadenprofile ersetzt. Aus diesem Anlass wird das „Cluster: ErzeugeKnotengeometrie“ in der Grasshopper-Definition implementiert. Der Aufbau des Clusters ist in Abbildung 133 gegeben. Das Cluster benötigt dafür die Geometrie der Boxen als „Geometry“ 125.3, die Referenzboxen der Profile 125.4, die Vektoren der Profilrichtungen 29.4 und die transformierten Zielboxen 129.1.

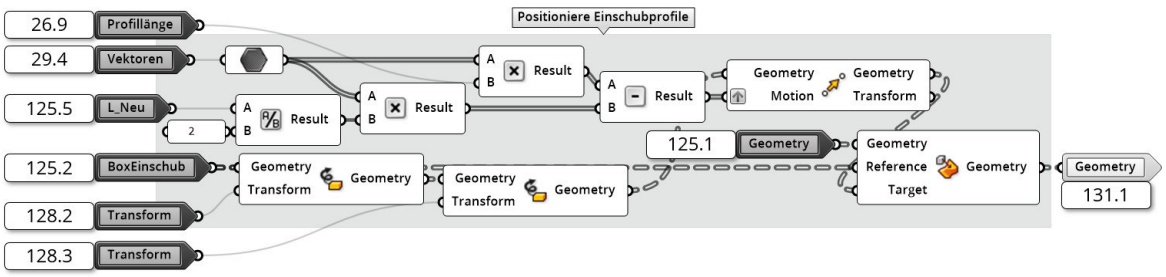


Abbildung 131: Das „Cluster: PositioniereEinschub“
Quelle: In Anlehnung an Grasshopper

Das Cluster übernimmt, neben der reinen Profilsubstitution, auch die Aufgabe, die beiden Pfostenprofile miteinander zu verschneiden. Dafür werden zunächst diese beiden Boxen aus der Datenstruktur der Zielboxen gefiltert und an das „Cluster: Verschneide Pfosten“ übergeben. Dieses Cluster übernimmt den Verschneidevorgang. Als weitere Inputs werden für das Cluster die Richtungsvektoren 29.4 und eine Filtermaske „Dispatch pattern“ benötigt. Diese Maske wird aus dem vorangestellten „Cluster: FilterProfilmitten“ entnommen. Das Ziel dieser Maske ist es, alle Punkte der Boxen zu erhalten, die sich im Inneren des Fassadenknotens befinden. Abbildung 134 verdeutlicht den Unterschied zwischen den Punkten im Inneren des Knotens (grau) und im Äußeren (grün). In Abbildung 135 ist das Innere des Clusters dargestellt. Um die Maske zu erstellen, wird sich der Fakt zunutze ge-

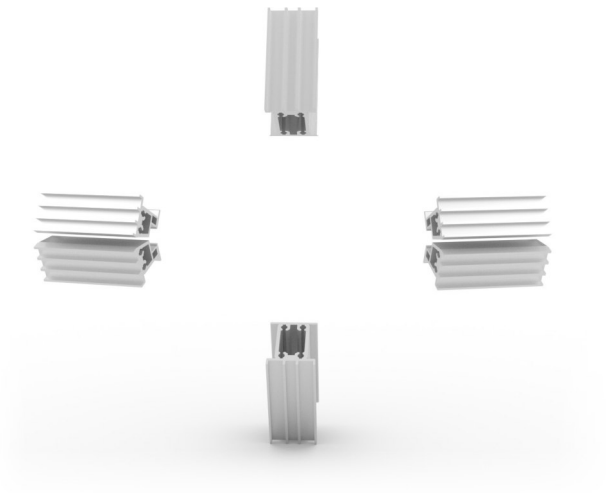


Abbildung 132: Rendering der Einschubprofile
Quelle: Screenshot aus Rhinoceros 6

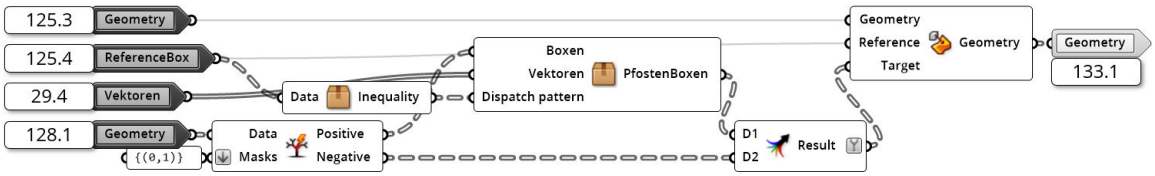


Abbildung 133: Das „Cluster: PositioniereEinschub“
Quelle: In Anlehnung an Grasshopper

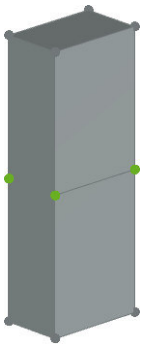


Abbildung 134: Rendering der Einschubprofile
Quelle: Screenshot aus Rhinoceros 6

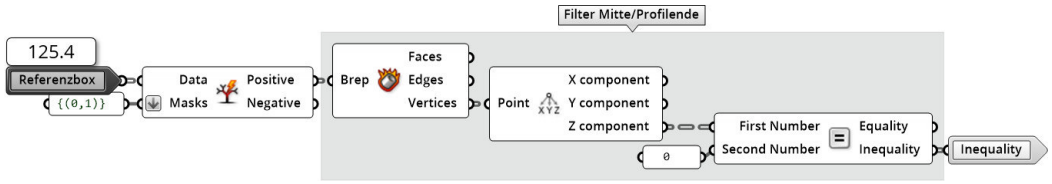


Abbildung 135: Das „Cluster: FilterProfilmitten“
Quelle: In Anlehnung an Grasshopper

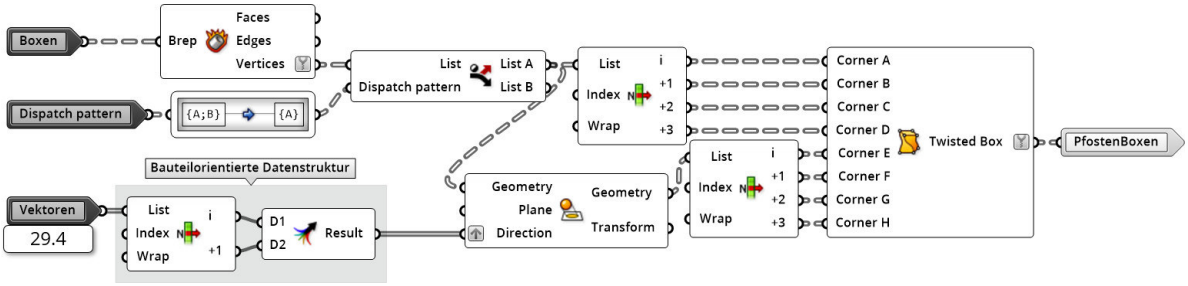


Abbildung 136: Das „Cluster: Verschneide Pfosten“
Quelle: In Anlehnung an Grasshopper

macht, dass die Punkte des Knoteninneren vor der Rotation alle in der x-y-Ebene liegen. Dadurch kann durch eine Überprüfung der z-Koordinaten aller Punkte das Knoteninnere vom Äußeren getrennt werden. Die Maske in „Inequality“ enthält dann eine Maske, die für alle Punkte des Knotenäußeren den Wert „True“ aufweist. Diese Maske kann, wie in Abbildung 136 dargestellt, für das „Cluster: Verschneide Pfosten“ als „Dispatch pattern“ verwendet werden, um die Punkte der Zielboxen „Boxen“ zu filtern. Die Punkte der Boxen können mithilfe der „Deconstruct Brep“-Komponente erzeugt werden. In „Liste A“ können so alle äußeren Punkte ausgegeben werden. Diese Punkte werden anschließend mit der „Project Along“-Komponente auf die x-y-Ebene projiziert. Die Ebene ist dabei in der Komponente gespeichert. Als „Direction“ werden die Vektoren der beiden Pfosten verwendet. Durch das Projizieren entlang der Richtungsvektoren werden die entsprechenden Schnittpunkte auf der Schnittebene der beiden Pfosten erzeugt. Dadurch, dass beide Profile immer gleich rotieren, ist die Schnittebene immer die x-y-Ebene. Sollen die Pfosten unabhängig voneinander rotiert werden, muss zunächst die Schnittebene berechnet werden. Die Begründung dieser Einschränkung kann dem Kapitel 4.2 entnommen werden. Die einzelnen Punkte werden anschließend zu einer Zielbox zusammengesetzt. Es ist zu beachten, dass es sich nicht weiter um eine Box handelt, da die Form nicht weiter einem Quader entspricht. Vielmehr liegen an dieser Stelle Prismen vor. Damit die „Box Morph“-Komponente für Prismen funktioniert, muss eine „Twisted Box“ vorliegen. Mit der „Twisted Box“-Komponente kann solch eine spezielle Box über die acht Eckpunkte generiert werden. Diese Boxen können als „PfostenBoxen“ weiter genutzt werden.

Wie in Abbildung 133 ersichtlich ist, werden die Zielboxen der Pfosten und Riegel zusammengeführt. Anschließend werden die eben erzeugten Boxen in der „Box Morph“-Kompo-

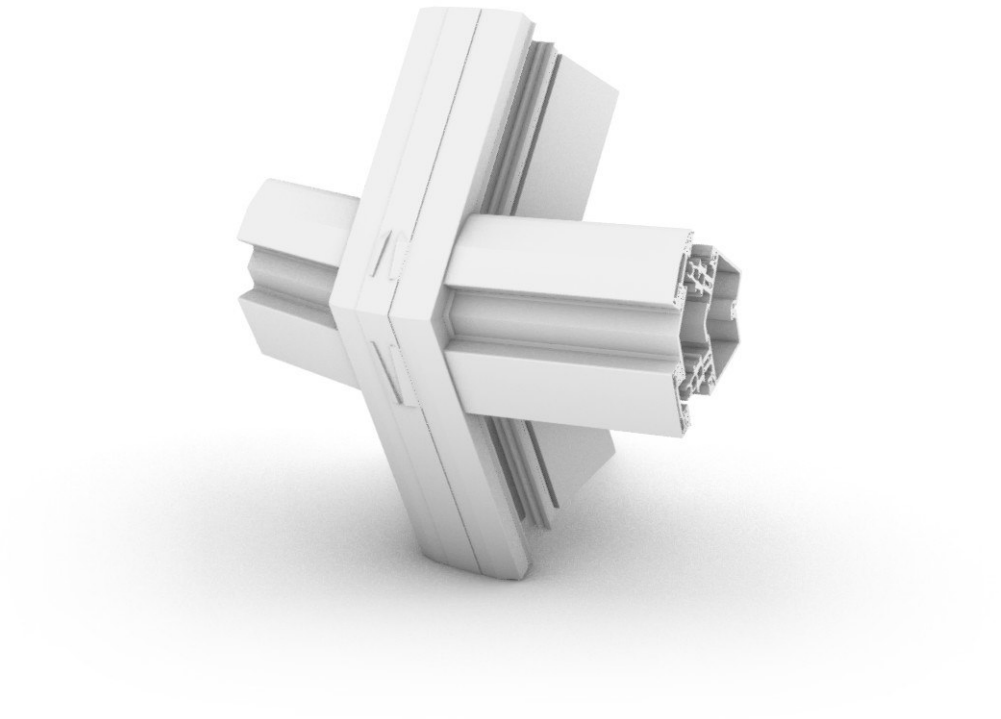


Abbildung 137: Rendering der Einschubprofile
Quelle: Screenshot aus Rhinoceros 6

nente als „Target“ genutzt. Die unveränderten, in Kapitel 6.1.2 erzeugten Referenzboxen werden als „Reference“ eingesetzt und die in Kapitel 6.1.1 erzeugten Referenzprofile als „Geometry“. Aus dem Cluster werden die positionierten Profile als „Geometry“ 133.1 ausgegeben. Das Ergebnis kann in Abbildung 137 betrachtet werden. In der Abbildungen wird verdeutlicht, dass das Verschneiden der beiden Pfostenprofile problemlos funktioniert, jedoch die beiden Riegel noch nicht an die Pfosten angepasst sind.

6.2 Verschneiden der Profile

Nachdem die Profile erfolgreich positioniert sind, kann mit dem vorletzten Programmschritt begonnen werden. Hierbei werden die Profile miteinander verschnitten. Da die Geometrien der Profile aufgrund des Verschneidens nicht mehr durch Prismen mit acht Ecken dargestellt werden können, kann ab diesem Punkt keine weitere Profilsubstitution stattfinden. Daher muss dieser Vorgang mit den vollständigen Profilgeometrien durchgeführt werden. Da ein Verschneiden von zwei vollständigen Profilen aufgrund der hohen LoD sehr zeitintensiv und fehleranfällig ist, wurden bereits in Kapitel 5.1 Cutter in die Datenstruktur aufgenommen, die die subtrahierende Geometrie stark vereinfachen. So kann die zu subtrahierende Geometrie erfolgreich an die anderen Bauteilkomponenten angepasst werden. Da jede Bauteilkomponente auf eine andere Art verschnitten werden muss, findet

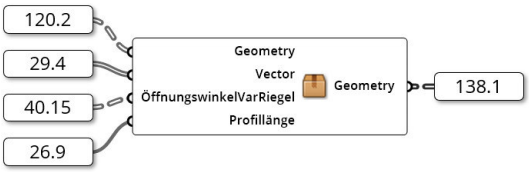


Abbildung 138: Das „Cluster: Verschneiden“
Quelle: In Anlehnung an Grasshopper

der Verschneidevorgang getrennt für die einzelnen Komponenten statt. Dabei ist zu beachten, dass bei den Pfosten alle Bauteilkomponenten, außer dem Tragprofil, bereits in Kapitel 6.1.5 ausreichend verschnitten wurden.

Um den Verschneidevorgang durchzuführen, wurde in der Grasshopper-Definition das „Cluster: Verschneiden“ implementiert. Dieses Cluster hat das Ziel, die positionierten Geometrien miteinander zu verschneiden, sodass Geometrieüberschneidungen verhindert werden. Das Cluster benötigt dazu, wie in Abbildung 138 verdeutlicht, die Profilgeometrie 133.1, die Richtungsvektoren der Profile 29.4, den Öffnungswinkel der Riegelprofile 40.15 und die Profillänge 26.9. Ausgegeben wird die Geometrie des verschnittenen Fassadenknotens 138.1. Der Ablauf des Clusters erfolgt in mehreren Schritten:

1. Öffnen der Datenstruktur und Zugriff auf die einzelnen Komponenten
2. Optimierung der Cuttergeometrien
3. Verschneiden der Fassadenprofile

Diese Schritte sollen im Folgenden erläutert werden.

6.2.1 Öffnen der Datenstruktur

Bevor die Profile miteinander verschnitten werden können, muss zunächst der Zugriff auf alle Komponenten, sowohl Bauteilkomponenten als auch Cutter, ermöglicht werden. Die Umsetzung in Grasshopper ist mit der Gruppe „Datenstruktur“ in Abbildung 139 gegeben. Da die Datenstruktur der „Geometry“ 133.1 für die Erzeugung der Referenzboxen in Kapitel 6.1.2 gruppiert wurde, muss die Struktur zunächst geöffnet werden. Dieser Vorgang wird durch das „Cluster: ÖffneDatenstruktur“ vorgenommen. Dabei wird analog zu Kapitel 5.2.4 vorgegangen. Das heißt, dass die einzelnen Hierarchiegruppen über die „Ungroup“-Komponente geöffnet werden. Anschließend können zunächst die Cutter aus der Datenstruktur herausgefiltert werden. Hierfür wird die „Split Tree“-Komponente mit der Maske {(0 to 3);6;*;*} verwendet, da sich die Cutter an Position 6 der Bauteilkomponenten befinden. Somit werden als „Positive“ die Cutter und als „Negative“ alle weiteren Komponenten

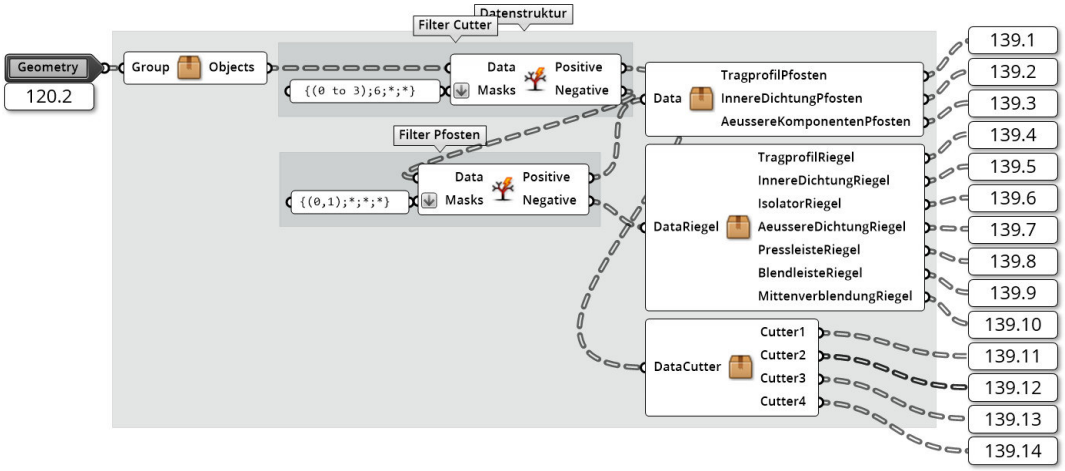


Abbildung 139: Die Gruppe „Datenstruktur“
Quelle: In Anlehnung an Grasshopper

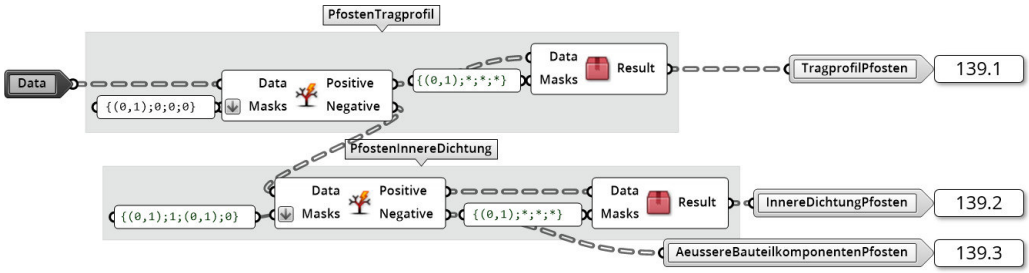


Abbildung 140: Das „Cluster: PfostenKomponenten“
Quelle: In Anlehnung an Grasshopper

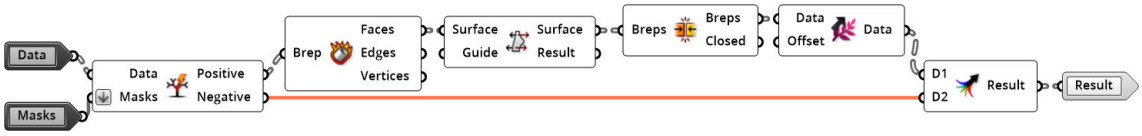


Abbildung 141: Das „Cluster: BREPRepair“
Quelle: Screenshot aus Grasshopper

ausgegeben. Die weiteren Bauteilkomponenten werden zudem mit einer weiteren „Split Tree“-Komponente in Pfosten und Riegel geteilt. Hierfür kommt die Maske „(0,1);*;*;*“ zum Einsatz, da die Bauteilpositionen 0 und 1 beide Pfosten besitzen. Um den Zugriff auf alle Bauteilkomponenten umzusetzen, sind drei Cluster implementiert.

Das erste „Cluster: PfostenKomponenten“ besteht, wie in Abbildung 140 zu sehen, aus zwei „Split Tree“-Komponenten, die mithilfe der Masken {(0,1);0;0;0} alle Tragprofile der Pfosten und {(0,1);1;(0,1);0} alle inneren Dichtungen von den restlichen Bauteilkomponenten trennt. Die restlichen Komponenten sind hierbei der Isolator, die Press- und die Blendleiste, folgend äußere Bauteilkomponenten des Pfostens genannt. Wie sich im Laufe der praktischen Ausarbeitung gezeigt hat, machen die boolschen Operationen in Grasshopper den Anschein, bei einigen Geometriekörpern fehleranfällig zu sein. Beispielsweise hat die

„Union“-Methode das Verhalten der „Difference“-Methode, und umgekehrt, gezeigt. Eine Vermutung des Autors ist, dass die Normalenrichtungen der Surface-Flächen der beiden zu verschneidenden Körper entgegengesetzt sind. Dies könnte daraus resultieren, dass die Querschnittsflächen in unterschiedliche z-Richtungen extrudiert wurden. Diese Vermutung konnte weder mit der Grasshopper Komponente „Face Normal“, noch mit weiteren 3-D-Modellierungsprogrammen, wie etwa dem Programm Blender der Blender Foundation, belegt werden. Jedoch konnte das Fehlverhalten durch das Drehen der Flächen von einem der beiden Körper behoben werden. Dieser Vorgang wird in dem „Cluster: BREPRepair“ vorgenommen. Dieses Cluster ist durch das rote Clustersymbol erkennbar. Der Inhalt ist in Abbildung 141 gegeben. Für diesen Vorgang werden zunächst die zu reparierenden Komponenten aus einer Datenstruktur „Data“ durch eine „Split Tree“-Komponente auf Grundlage einer Maske „Masks“ herausgefiltert. Folgend wird die Geometrie mit der „Deconstruct Brep“-Komponente in die einzelnen Flächen geteilt. Die Flächen können anschließend mit der „Flip“-Komponente gedreht werden und durch „Brep Join“ zu einem geschlossenen Brep zusammengesetzt werden. Abschließend wird die eingehende Datenstruktur wiederhergestellt.

Das „Cluster: RiegelKomponenten“ und das „Cluster: Cutter“ funktionieren analog zum „Cluster: PfostenKomponenten“. Hierbei wird die Datenstruktur in die einzelnen Bauteilkomponenten mithilfe der „Split Tree“-Komponente aufgeteilt. Eine weitere Verwendung des „Cluster: BREPReipair“ ist an dieser Stelle noch nicht durchgeführt.

Durch die eben erläuterten Vorgänge, kann, wie in Abbildung 139 zu sehen ist, auf die einzelnen Bauteilkomponenten 139.1 - 139.14 zugegriffen werden. Dies ist von essenzieller Bedeutung, um die einzelnen Fassadenprofile zu einem kompletten Fassadenknoten zusammenzufügen. Es ist damit zu begründen, dass jede Komponente auf eine andere Art und Weise verschnitten wird.

6.2.2 Optimierung der Cutter

Als letzte Vorbereitung vor dem Verschneiden müssen die Geometrien der Cutter optimiert werden. Um dies zu erreichen, werden die vier Cutter in vier verschiedenen Clustern optimiert. Als Optimierung werden dabei folgende Maßnahmen getroffen:

- Anpassen der Datenstruktur
- Korrekturen der Geometrie oder der Position
- Reparatur der Geometrie mithilfe des „Cluster: BREPRepair“
- Verschneiden der Cuttergeometrie mit anderen Cuttern

All diese Maßnahmen sind erforderlich, damit die boolschen Operationen erfolgreich berechnet und somit die gewünschten Geometrien erzielt werden können.

Wie in Abbildung 142 zu sehen ist, werden in vier verschiedenen Clustern die Cutter optimiert. Aus den vier Cuttern 139.11 - 139.14 können mit den Richtungsvektoren der Profile 29.4, der Profillänge 26.9 und dem Öffnungswinkel der Riegel 40.15 sechs verschiedene Cutter-Datenstrukturen „CutterData“ 142.1 - 142.6 erzeugt werden.

Abbildung 143 zeigt, wie im ersten Cluster die Datenstruktur der Cutter an die zu schneidende Geometrie angepasst wird. Hierfür werden die Cutter-Pfade mit dem „Path Mapper“ auf die zu schneidenden Geometrien angepasst und als 143.1 weitergeleitet. Die Vektoren werden entsprechend der Cutter-Pfade modifiziert und mit den halben Profillängen verrechnet. Durch diese Verschiebung wird erzielt, dass sich die Geometrien eindeutig schneiden. Das Berechnungsergebnis wird in 143.2 weitergeleitet beziehungsweise als „Motion“ für weitere Cluster verwendet. Abbildung 144 zeigt den zweiten Teil des ersten Clusters. Es ist zu sehen, dass die Geometrie 143.1 entsprechend der berechneten „Motion“ 143.2 verschoben wird. Anschließend werden sich als fehlerhaft erwiesene

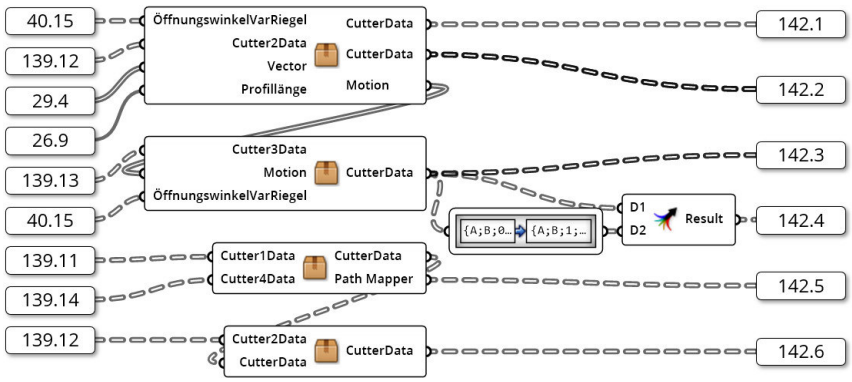


Abbildung 142: Optimierung der Cutter
Quelle: In Anlehnung an Grasshopper

Cuttergeometrien mit dem „Cluster: BREPRepair“ korrigiert und entweder als „CutterData“ 142.2 ausgegeben oder nicht benötigte Cutter herausgefiltert 142.1. Hierfür wird die Maske mit einem zu Q16 analogen Skript erzeugt. Dieses Skript Q25 „Script: KorrekturVarRiegel I“ ist dem Anhang beigefügt. Nach Abbildung 152 wird der Cutter 142.1 zum Schneiden der Tragprofile und Isolatoren der Riegel genutzt. Der Vergleich vor und nach dem Verschneiden dieser Komponenten ist in Abbildung 145 dargestellt. Die ungefilterten „CutterData“ 142.2 werden für die inneren Dichtungen der Riegel und die Mittenblenden der variablen Riegelprofile genutzt. Die durch das Verschneiden erzeugte Geometrie ist in

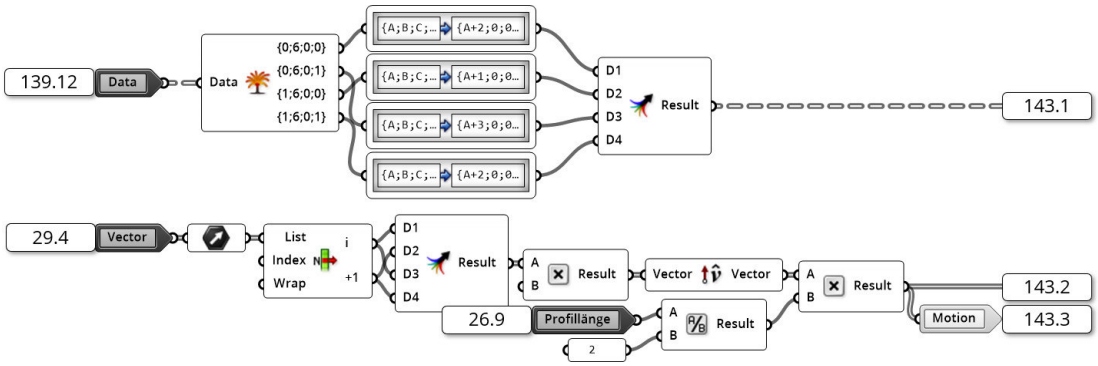
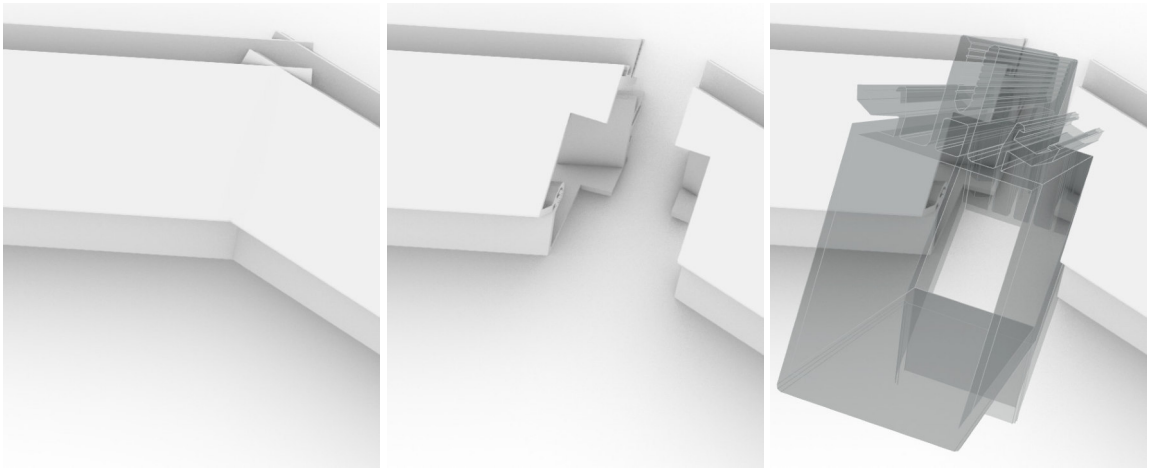


Abbildung 143: Optimierung der Cutter - Das erste Cluster (1)
Quelle: In Anlehnung an Grasshopper

Abbildung 144: Optimierung der Cutter - Das erste Cluster (2)
Quelle: In Anlehnung an Grasshopper



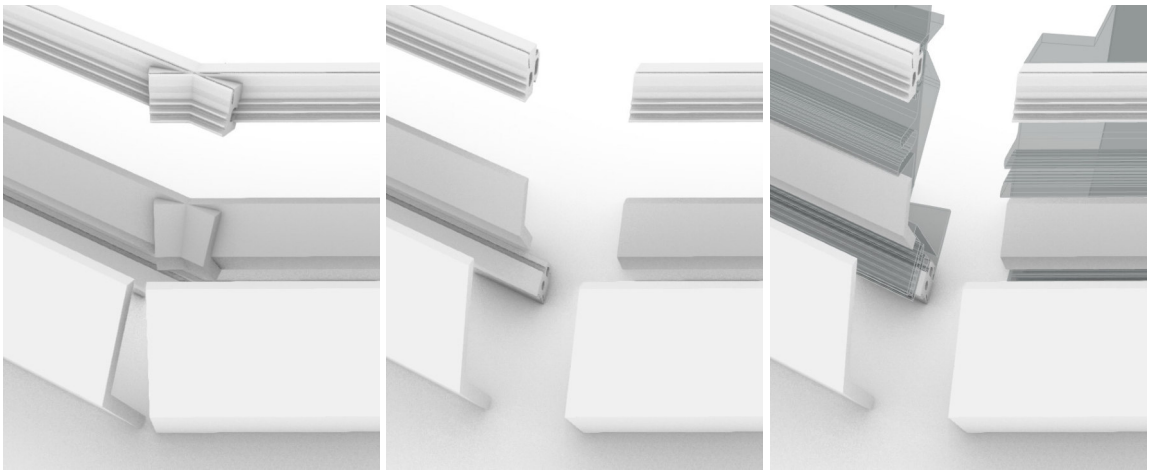
(a) (b) (c)

Abbildung 145: Tragprofile und Isolatoren der Riegel vor (a) und nach (b,c) dem Verschneiden mit (c) und ohne (b) angedeutetem Pfosten (grau)
Quelle: Screenshot aus Rhinoceros 6

Abbildung 146 gegeben. Dass hierbei kein Filtern der Datenstruktur notwendig ist, wird damit begründet, dass diese Bauteilkomponenten immer aus der gleichen Stückanzahl bestehen.

Das zweite Cluster benötigt die Cuttergeometrie „Cutter3Data“ 139.13, die Verschiebungsvektoren „Motion“ vom ersten Cluster und den Öffnungswinkel der Riegel 40.15. Wie in Abbildung 147 dargestellt, werden die Cutter analog zum ersten Cluster optimiert. Das bedeutet, dass zunächst die Datenstruktur angepasst, anschließend die Geometrien verschoben und abschließend mit dem „Cluster: BREPRepair“ korrigiert werden. Die durch das zweite Cluster entstehenden Cutter werden für die äußeren Bauteilkomponenten, also die äußeren Dichtungen, sowie Press- und Blendleiste der Riegel genutzt. Die daraus resultierenden Geometrien sind in Abbildung 148 gegeben.

Für den dritten Cutter muss die Geometrie des ersten Cutters mit der Geometrie des vierten verschnitten werden. Dieser Vorgang ist in Abbildung 149 dargestellt. Neben dem Verschneiden der beiden Cutter finden analog zu den ersten beiden Clustern Datenstrukturoperationen und Geometriekorrekturen durch das „Cluster: BREPRepair“ statt. Die erzeug-



(a) (b) (c)

Abbildung 146: Innere Dichtung und Mittenblenden der Riegel vor (a) und nach (b,c) dem Verschneiden mit (c) und ohne (b) angedeutetem Tragprofil (grau)
Quelle: Screenshot aus Rhinoceros 6

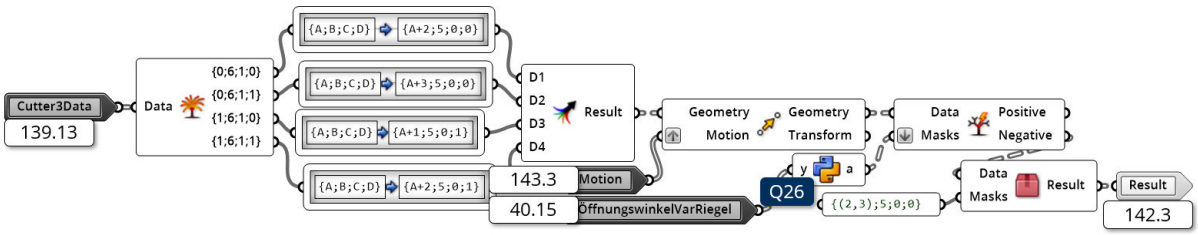
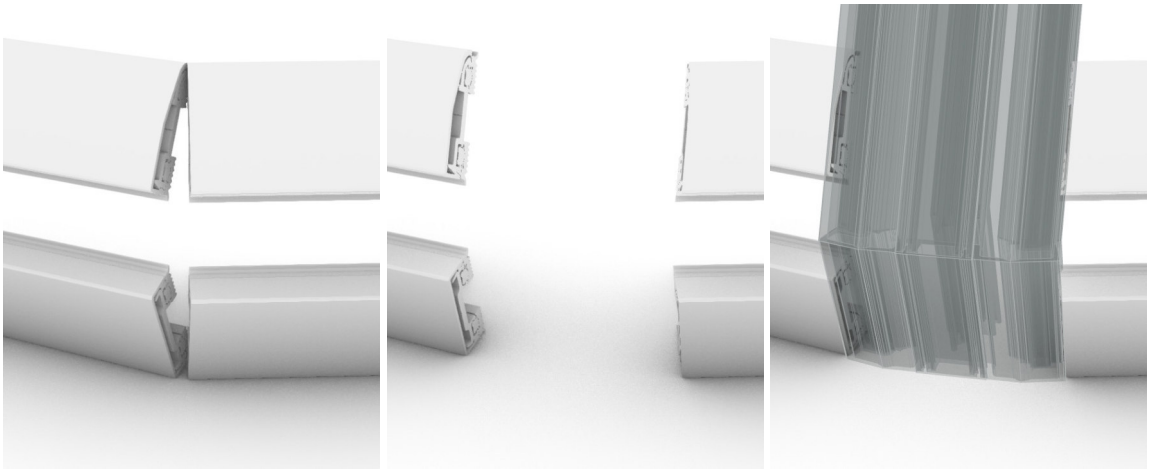


Abbildung 147: Optimierung der Cutter - Das zweite Cluster
Quelle: In Anlehnung an Grasshopper

ten Geometrien 142.5 werden für die inneren Dichtungen der Pfostenprofile verwendet. Die so erzeugten Geometrien sind zusammen mit den Ergebnissen des letzten Clusters in Abbildung 151 dargestellt.

Das vierte, in Abbildung 150 dargestellte Cluster ist analog zu den ersten drei Clustern aufgebaut. Wie beim dritten Cluster wird im vierten Cluster die Cuttergeometrie mit dem ersten Cutter verschnitten und anschließend verschoben. Die Verschiebung ist im Fall des vierten Cutters sehr gering, da der Cutter an der richtigen Position ist. Jedoch treffen ohne die Verschiebung parallele Flächen aufeinander, sodass diese keinen eindeutigen Schnitt aufweisen. In der Abbildung 151 ist das Ergebnis des Clusters dargestellt.



(a) (b) (c)

Abbildung 148: Äußere Bauteilkomponenten der Riegel vor (a) und nach (b,c) dem Verschneiden mit (c) und ohne (b) angedeutetem Pfosten (grau)
Quelle: Screenshot aus Rhinoceros 6

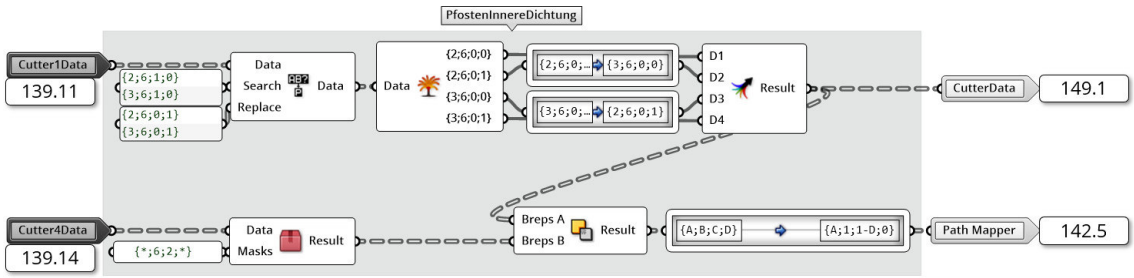


Abbildung 149: Optimierung der Cutter - Das dritte Cluster
Quelle: In Anlehnung an Grasshopper

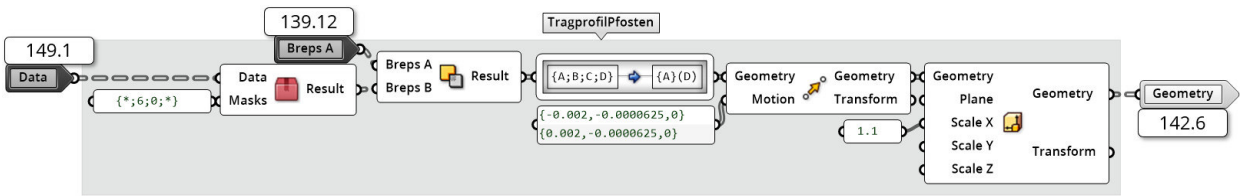
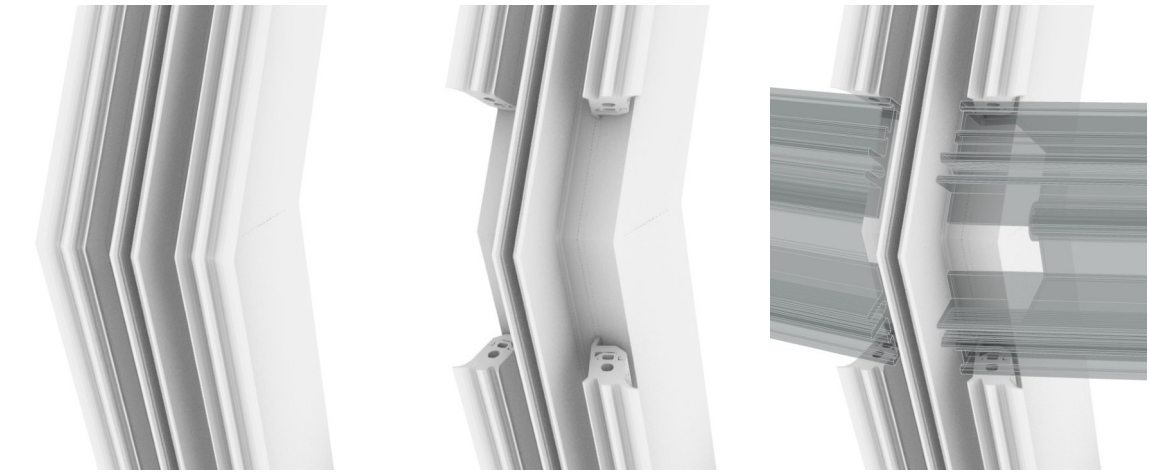


Abbildung 150: Optimierung der Cutter - Das vierte Cluster
Quelle: In Anlehnung an Grasshopper

In Abbildung 152 wird gezeigt, wie die einzelnen Geometrien und Cutter verschnitten werden. Hierfür werden die „Solid Difference“-Komponenten genutzt. Diese führen eine boolsche Operation aus. Es ist zu erkennen, dass die Cutter aus Abbildung 142 von den Bauteilgeometrien aus Abbildung 139 subtrahiert werden. Die Ergebnisse der einzelnen Verschneidungen sind in den Abbildung 145, 146, 148 sowie 151 dargestellt.



(a) (b) (c)

Abbildung 151: Tragprofil und innere Dichtung des Pfostens vor (a) und nach (b,c) dem Verschneiden mit (c) und ohne (b) angedeutetem Riegeltragprofil (grau)
Quelle: Screenshot aus Rhinoceros 6

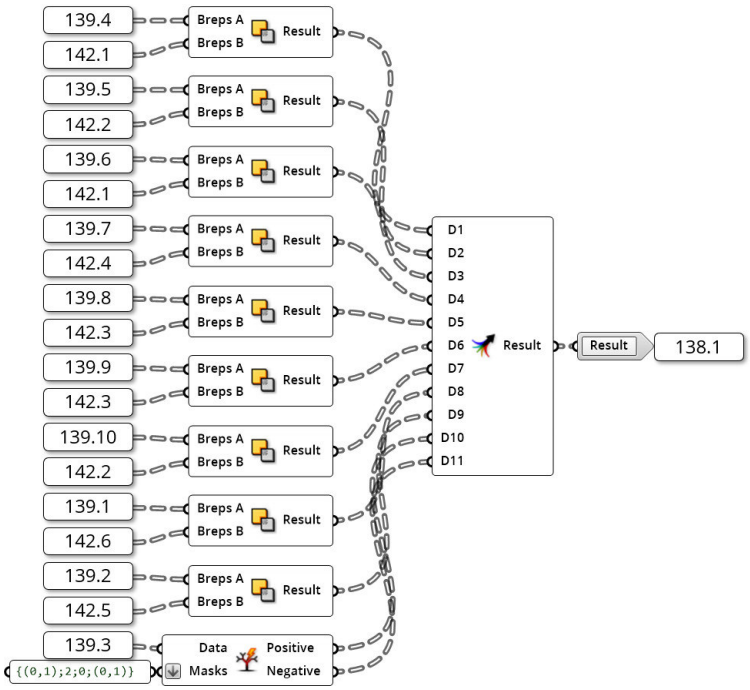


Abbildung 152: Verschneiden der Geometrie mit den boolschen Operationen
Quelle: In Anlehnung an Grasshopper

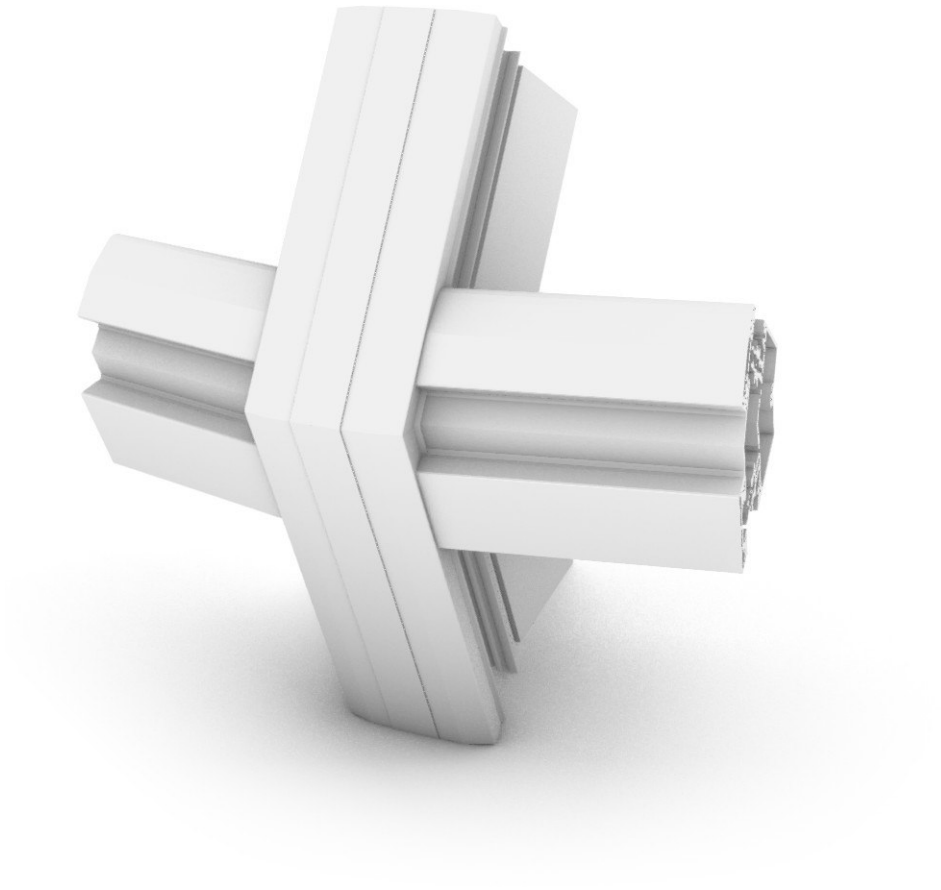


Abbildung 153: Der fertige Fassadenknoten
Quelle: Screenshot aus Grasshopper

Alle weiteren Bauteilkomponenten müssen nicht verschnitten werden. Dabei handelt es sich um die äußeren Komponenten der Pfostenprofile 139.3. Dies hat den Vorteil, dass von außen der Eindruck entsteht, dass die Profile durchgängig sind. Die Abbildung 145, 146, 148 und 151 verdeutlichen sehr anschaulich, wie die unterschiedlichen Bauteilkomponenten eines Pfosten-Riegel-Fassadenknotens geschnitten werden müssen. Dabei ist auch erkennbar, dass die Schnittführung sehr komplex ist, sodass ein manuelles Schneiden der Profile mit großen Ungenauigkeiten einhergeht.

Damit ist die Erzeugung der Knotengeometrie abgeschlossen. Die Daten werden als 138.1 zurückgegeben. In Abbildung 153 ist eine Darstellung des fertigen Fassadenknotens gegeben.

6.3 Baken der Geometrie

Unter Baken versteht man das Abspeichern der innerhalb der Grasshopper-Definition erzeugten Geometrie. Dazu werden die Geometrien an Rhinoceros 6 übertragen. Geometrie, die den Bake-Vorgang durchlaufen hat, kann nicht weiter durch die Parameter der Grasshopper-Definition manipuliert werden. Dies ist darin begründet, dass nur die reinen Geometriedaten und einige wenige Metadaten, wie etwa Objektnamen, gesichert werden.

Für den Bake-Vorgang wird das in Abbildung 154 dargestellte „Cluster: Bake“ implementiert. Das Cluster benötigt die zu bakende Geometrie 138.1 und den Boolean „Bake?“ 26.10. Letzterer gibt an, ob beim Berechnen des Knotens der Bake-Vorgang ausgeführt werden soll oder nicht. Standardmäßig ist der Boolean auf FALSE gestellt, sodass kein Bake-Vorgang stattfindet. Innerhalb des Clusters wird zunächst ein Datenbaum mit allen möglich vorkommenden Ebenennamen gefiltert. Dieser Baum wird im „Cluster: LayerNames“ erzeugt und mithilfe der Pfade im Datenbaum der Geometrie gefiltert. Hierfür wird die „Tree Branch“-Komponente verwendet. Mit den Ebenennamen werden Ebenen erstellt. Dazu wird die „Create/Modify Layers“-Komponente des „Human“[5]-Grasshopper-Plug-Ins verwendet. Bei der Komponente ist zu beachten, dass auch Sublayers – also Unterebenen – erstellt werden können. Dazu muss der Ebenenname eine entsprechende Semantik aufweisen. Die Semantik ist dabei:

$$LayerName :: SubLayerName :: SubSubLayerName :: [...]$$

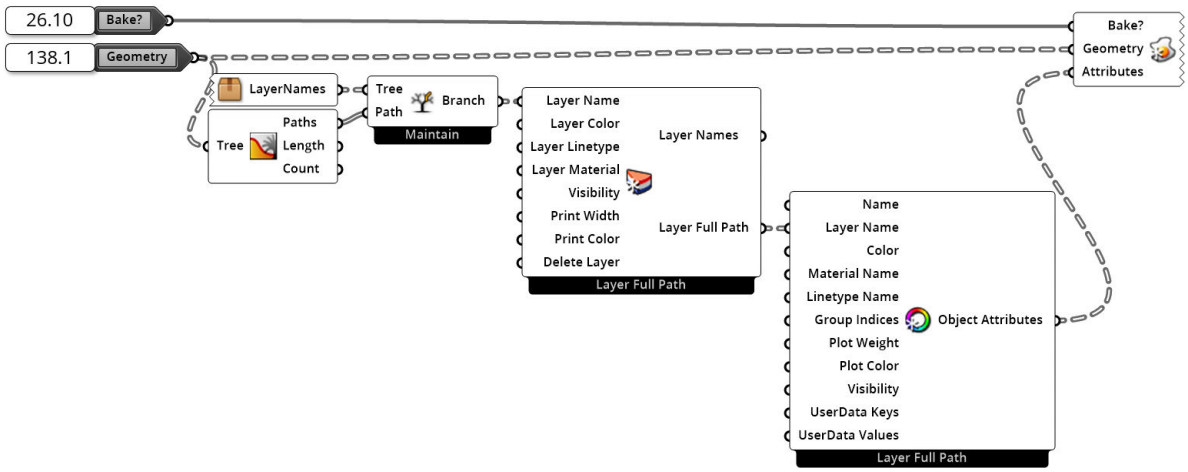


Abbildung 154: Das „Cluster: Bake“
Quelle: In Anlehnung an aus Grasshopper

[5]Das Plug-In „Human“ stammt vom Autoren „andheum“ und ist unter https://www.food4rhino.com/app/human (aufgerufen am 28.09.2019) verfügbar.

Abbildung 155: Die Ebenenstruktur beim Baken
Quelle: Screenshot aus Rhinoceros 6

Zudem muss der Ebenenname im Folgenden als „Layer Full Path“ behandelt werden und die „CreateAttributes“[5]-Komponente („Human“-Plug-In) auch auf „Layer Full Path“ umgestellt werden. Mit den Ebenennamen, der Geometrie und dem „Bake?“-Boolean kann die Geometrie mit der „Bake Geometry“[5]-Komponente gebaket werden.

Mithilfe der Sublayer-Struktur ist es möglich, die verschiedenen Bauteilkomponenten, wie in Abbildung 155 dargestellt, in einzelnen Ebenen, sortiert nach Bauteilen zu speichern. Somit können die einzelnen Bauteile über die einzelnen Ebenen erreicht werden.

7 Digitale Produktion des parametrischen Fassadenknotens

Um den parametrischen Fassadenknoten mithilfe von digitalen Fertigungsmethoden herstellen zu können, ist es notwendig die auszuführenden Schnitte in G-Code umzuwandeln. Der G-Code kann weiter genutzt werden, um eine CNC-Maschine oder einen Roboterarm anzuleiten. Im Rahmen dieser Arbeit soll aufgezeigt werden, dass die Erzeugung eines solchen G-Codes für den parametrischen Fassadenknoten möglich ist. Für diese Arbeit wird angenommen, dass mithilfe des G-Codes die Programmierung einer CNC-Anlage beziehungsweise eines Roboters möglich ist, sodass auf die weiterführende Programmierung nicht weiter eingegangen wird.

7.1 Programmierung von CNC-Maschinen

Bereits im Rahmen der Projektarbeit konnte die grundlegende Programmierung von CNC-Maschinen erläutert werden. Dabei wurden zwei sehr relevante Programmierarten herausgearbeitet. Zum einen existieren hierfür praxisrelevante CNC-Programmierungen und zum anderen die CNC-Programmierung nach DIN 66025-1:1983-01, die unter anderem als G-Code bekannt ist. In den folgenden zwei Kapiteln sollen die Erkenntnisse der Projektarbeit kurz zusammengefasst werden.

7.1.1 Praxisrelevante CNC-Programmierung

Die Programmierung von CNC-Maschinen wird in der Praxis entweder im Büro oder innerhalb einer Werkstatt vorgenommen. Dem Programmierer stehen dabei Hilfsmittel zur Verfügung, wie eine benutzerfreundliche Programmierumgebung, WOP oder Teach-In-Funktionen. (Kief, Roschiwal und Schwarz 2017: S. 610)

Der Bergiff werkstattorientierte Programmierung (WOP) beschreibt eine leicht verständliche, grafisch animierte Programmierumgebung. Die Erstellung des Programmcodes erfolgt hierbei halbautomatisiert. Daraus resultiert eine Reduzierung der Anforderungen an die Programmierkenntnisse des Bedieners. (Kief, Roschiwal und Schwarz 2017: S. 611)

Eine Programmierung von CNC-Maschinen kann auch ohne Programmierkenntnisse erfolgen. Hierfür stehen Methoden, wie Teach-In und Playback-Verfahren, zur Verfügung. Der Maschinenbediener fährt hierfür alle Positionen des zu fertigenden Werkstücks manuell

an. Die CNC-Maschine speichert sich diesen Bewegungsablauf und kann diesen replizieren. Hierbei wird in eine Speicherung der einzelnen Positionen oder des gesamten Bewegungsablaufs unterschieden. Eine CNC-Maschine kann zwischen einzelnen Positionen selbstständig einen Pfad wählen. (Kief, Roschiwal und Schwarz 2017: S. 611 f.)

Komplexe, dreidimensionale Fertigungsaufgaben werden meist im Büro vorbereitet. Dabei kommen CAD- und CAM-Systeme zum Einsatz. Computer Aided Drafting beziehungsweise Computer Aided Design (CAD) beschreibt das computergestützte Zeichnen beziehungsweise Entwerfen (Kief, Roschiwal und Schwarz 2017: S. 705). Mit Computer Aided Manufacturing (CAM)-Systemen kann die zu fertigende Geometrie generiert und daraus ein Bearbeitungsablauf erstellt werden. Für den Bearbeitungsablauf werden unter anderem die Werkzeugauswahl und die Vorschubgeschwindigkeiten gespeichert. Ein wichtiger Bestandteil eines CAM-Systems ist die Simulation des im NC-Programm definierten Bewegungsablaufs. (Kief, Roschiwal und Schwarz 2017: S. 613, 707)

Bei den genannten computergestützten Programmiersystemen wird nicht die Werkzeugbewegung selbst, sondern vielmehr die zu fertigende Geometrie des Werkstücks programmiert. Das Fertigungsprogramm wird anschließend aus diesen Informationen impliziert. (Kief, Roschiwal und Schwarz 2017: S. 614)

7.1.2 CNC-Programmierung nach DIN 66025

Der Aufbau von Steuerprogrammen von CNC-Maschinen ist in der DIN 66025-1:1983-01 geregelt. Diese DIN ist international in der ISO 6983-1:2009 umgesetzt. Der Code, der durch diese DIN beschrieben ist, wird als NC-Code, DIN/ISO-Programmierung (Kief, Roschiwal und Schwarz 2017: S. 566, 611), G-Code oder DIN-Code (Saenger 2019) bezeichnet.

Nach der DIN 66025-1:1983-01: (S. 2 f.) besteht ein G-Code aus einem Programmstart, einer Folge von Sätzen und einem Programmende. Das Zeichen „%“ leitet den Programmstart ein. Alle vorangestellten Zeichen werden von einer CNC-Maschine nicht ausgeführt. Nach dem Zeichen für den Programmstart, kann ein Programmname vergeben werden. Dieser setzt sich aus einer beliebigen Zeichenkette zusammen. Folgend ist beispielhaft ein Programmstart gegeben. Hierbei gilt zu beachten, dass Kommentare zwischen den Zeichen „(“ und „)“ geschrieben werden.

Quelltext 1: Programmstart

```
1  % Schneidprogramm A (Programmname: "Schneidprogramm A")
2  ...   (Ab hier kann der erste Satz definiert werden)
```

Tab. 4: G-Code: Aufbau von Sätzen

Nr.	Wort	Beschreibung	Bemerkung
1	N	Wort für Satznummer	
2	G	Wort für Wegbedingung	
3	X, Y, Z, U, V, W, P, Q, R, A, B, C	Wörter für Koordinaten	
4	I, J, K,	Wörter für Interpolationsparameter bzw. Parameter für Gewindesteigung	Bezieht sich auf eine Gruppe von Wörtern. Das Wort wird dann direkt hinter der betreffenden Gruppe geschrieben.
5	F, E	Wörter für Vorschub	Bezieht sich entweder auf einzelne (direkt dahinter geschrieben) oder eine Gruppe (hinter der letzten Koordinate geschrieben) von Koordinaten.
6	S	Wort für Spindeldrehzahl	
7	T, D	Wörter für Werkzeug	Nur T für Werkzeug inkl. Korrektur oder T für Werkzeug in Kombination mit D für Korrektur
8	M	Wort für Zusatzfunktion	

Quelle: Eigene Darstellung, Inhalt in Anlehnung an DIN 66025-1:1983-01: S. 3

Nach dem Programmstart werden die Programmanweisungen als Sätze definiert. Jeder Satz repräsentiert eine Programmanweisung. Ein Satz setzt sich wiederum aus mehreren Wörtern zusammen. In der DIN 66025-1:1983-01 wird für die Wörter die in Tab. 4 dargestellte Reihenfolge vorgeschrieben.

Zur Strukturierung können Sätze zu Absätzen zusammengefasst werden. Diese stellen somit Teilprogramme dar. Jeder Absatz wird durch einen Hauptsatz eingeleitet. Dieser wird durch das Zeichen „:“ definiert. (DIN 66025-1:1983-01: S. 3)

Nach DIN 66025-1:1983-01 1983: (S. 3 f.) ist der Aufbau eines Wortes definiert als Adressbuchstabe mit nachgestellter Zeichenfolge. Diese Zeichenfolge kann dabei ein Vorzeichen enthalten, falls für die CNC-Maschine negative Zahlenbereiche definiert sind. Bei einigen Wörtern wird die Zeichenfolge als Parameter verwendet. Etwa der Befehl „X1030“, der eine X-Koordinate mit dem Abstand von 1030,00 mm in positiver x-Richtung vom Ursprung aus beschreibt. Andere Wörter, etwa das G-Wort, können die Zeichenfolge als zweistellige Schlüsselzahl interpretieren und so vordefinierte Funktionen aufrufen. Beispielsweise stellt der Befehl „G71“ die CNC-Maschine in das metrische System (Grundeinheit in der Regel mm) um, wohingegen der Befehl „G70“ das angloamerikanische System (Grundeinheit in der Regel inch) einstellt.

Weitere Erläuterungen sowie ein ausführliches Beispiel zum G-Code können aus der vorangestellten Projektarbeit entnommen werden.

7.2 Vorbereitung der Geometrie

Bevor der G-Code der auszuführenden Schnitte erzeugt werden kann, müssen die zu schneidenden Geometrien vorbereitet werden. Dazu müssen die Profile des Fassadenknotens neu positioniert und rotiert werden, damit die digitalen Profile entsprechend einer realen CNC-Fertigungsanlage platziert sind. Da alle Transformationsinformationen innerhalb der Grasshopper-Definition aus Kapitel 4 und 6 vorhanden sind, werden alle notwendigen Informationen, wie etwa die Eingabeparameter des Nutzers, aber auch die Geometrie der Fassadenprofile mithilfe der „Data Output"- und der „Data Input"-Komponente an eine neue Grasshopper-Definition übertragen. Dazu ist es gegebenenfalls notwendig, eine .ghdata-Datei anzulegen, die die zu übergebenden Daten speichert, und diese gegebenenfalls in der zweiten Komponente auswählt.

Für diese Diplomarbeit wird die Annahme getroffen, dass eine Fertigungsanlage, die vergleichbar mit der Schneideanlage „Schüco Maschine DC 500" der Firma Schüco International KG ist, für die Produktion des parametrischen Fassadenknoten verwendet werden kann. Dazu werden die Profile wie in Abbildung 156 veranschaulicht, auf ein Förderband gegeben, sodass dieses die Profile zur Schneidevorrichtung transportieren kann. In Abbildung 156 sind die einzelnen Profile übereinander dargestellt. In der realen Fertigungsanlage müssen die Profile nacheinander einzeln geschnitten werden. Hierfür ist es notwendig die einzelnen Profile vor dem Schneiden zu unterscheiden. Bei der gewählten Fertigungsanlage kann dieser Schritt über ein Barcode-Scanner erfolgen. Zu diesem Zweck ist es notwendig alle Profile mit einem eindeutigen Identifier zu versehen, der dann über einen Barcode eingelesen werden kann, sodass die Fertigungsanlage das entsprechende Programm laden kann.

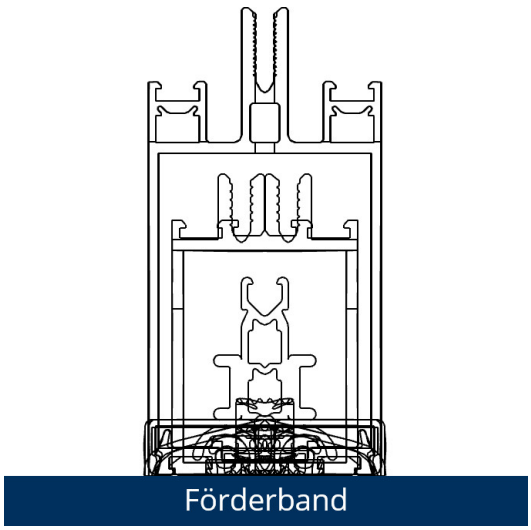


Abbildung 156: Fassadenprofile auf dem Förderband
Quelle: In Anlehnung an Rhinoceros 6

Des Weiteren müssen zunächst die digitalen Profile entsprechend der Abbildung 156 im 3-D-Raum positioniert werden, um die Schnittdaten für die Maschine zur Verfügung zu stellen. Hierfür wird die Grasshopperdatei „Schubert05_GCode.gh“ verwendet. Diese ist in Anhang D5 digital beigefügt. Die Informationen aus der ersten Grasshopper-Definition können mit der „Data Output“- und der „Data Input“-Komponente an die zweite Definition übergeben werden. Dazu ist es wichtig, dass zunächst die Datei „Schubert04_parametrischerKnoten.gh“ aus Anlage D4 ausgeführt und bestenfalls im Hintergrund geöffnet wird. Die Inputparameter aus der ersten Grasshopper-Definition können jeder Zeit manipuliert werden. Die geänderten Daten werden anschließend in der zweiten Definition aktualisiert. Wie in Abbildung 157 zu sehen ist, können so die Geometrie des Fassadenknotens 157.5, die fünf vom Nutzer einstellbaren Winkel 157.6 - 157.10, sowie die beiden berechneten Winkel „LeftAngle“ 157.11 und „RightAngle“ 157.12 importiert werden. Des Weiteren sind vom Nutzer einige Attribute wählbar. Dadurch kann in den vier Dropdown-Menüs 157.1 - 157.4 eingestellt werden, welche Schnittdaten behandelt werden sollen. Mithilfe des „Bake - Schnitte“-Buttons 157.13 können die Schnittgeometrien in Rhinoceros 6 gespeichert werden.

Um die Profile für die Fertigungsmaschine positionieren zu können, müssen als erstes die Rotationen der Fassadenprofile rückgängig gemacht werden. Um dies zu erreichen, wird der in Abbildung 158 dargestellte Bereich implementiert. Grundlegend besteht dieser Teil des Algorithmus aus vier Clustern und zwei „Rotate Axis“-Komponenten. Die Cluster dienen hierbei dazu, die Informationen der Winkel 157.6 - 157.10 und der Rotationsachsen an die Datenstruktur der Geometrie 157.5 anzupassen. Diese liegt weiterhin in der in Kapitel 3.4.2.2 beschriebenen Datenstruktur vor. Dabei ist wichtig, dass alle Pfade dieser Struktur mit den Rotationsinformationen beschrieben werden. Um dies zu erreichen wird, wie in Abbildung 159 dargestellt, das „Cluster: DatenZuDatenstruktur“ implementiert. Alle in

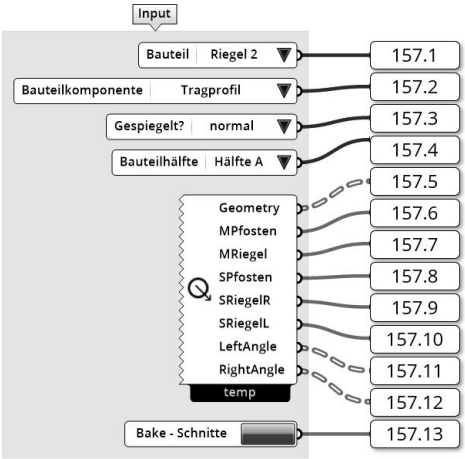


Abbildung 157: Eingabeparameter zur Erzeugung der Schnittgeometrien
Quelle: In Anlehnung an Grasshopper

Abbildung 158 dargestellten Cluster sind Kopien beziehungsweise leichte Abwandlungen dieses Clusters. In Abbildung 159 ist beispielhaft das Oberste dargestellt. Das zentrale Element dieser Cluster ist eine „Series“-Komponente. Durch diese Komponente lässt sich eine Zahlenfolge mit einer gewissen Länge erzeugen. Da die Winkel der Bauteile für alle Bauteilkomponenten verwendet werden sollen, wird die Datenstruktur der Geometrie genutzt, um mit der „Tree Statistics“-Komponente die Anzahl der Bauteilkomponenten je Bauteil zu ermitteln. Diese Information wird als „Count“ verwendet. Da die Winkel eines Bauteils für alle seine Bauteilkomponenten gleich sind, wird der „Step“-Wert, also der Abstand zwischen den Zahlen der Zahlenfolge, auf den Wert 0 gesetzt. Als „Start“ können die einzelnen Winkel 157.6 - 157.10 verwendet werden. Somit erhält man eine Datenstruktur, in der jeder Bauteilkomponente der dazugehörige Rotationswinkel zugeordnet wird. Um diese Zahlenfolgen in die ursprüngliche Form zu bringen, wird die „Match Tree“-Komponente genutzt.

Die beiden oberen Cluster entsprechen genau dem in Abbildung 159 dargestellten Schema. In beiden Clustern werden die fünf nutzerspezifischen Winkelinformationen verarbeitet. Für die Rotationsachsen ist eine kleine Anpassung des Clusters notwendig. Zum En-

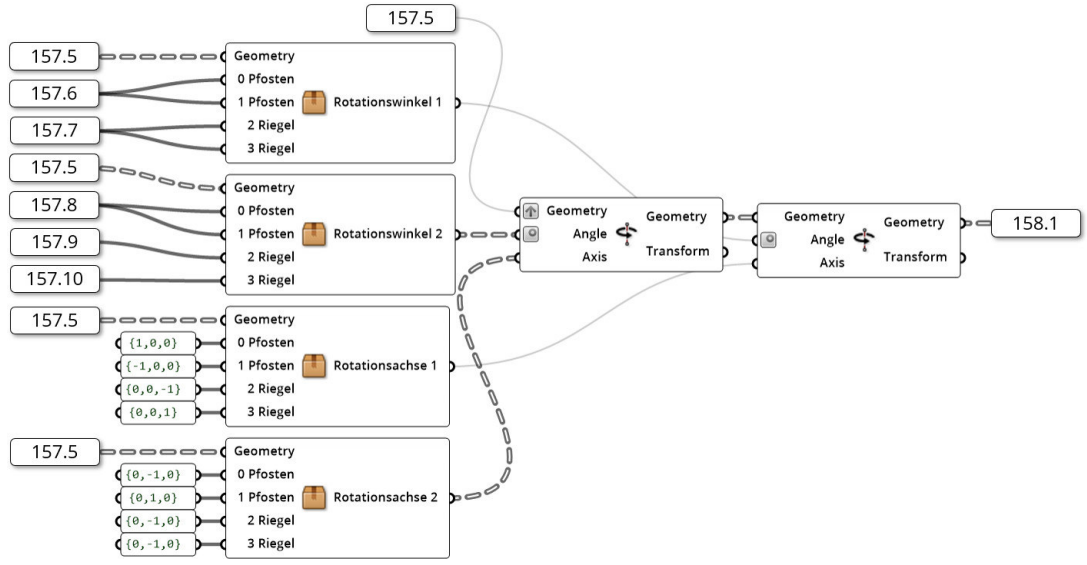


Abbildung 158: Umkehrung der Profilrotationen
Quelle: In Anlehnung an Grasshopper

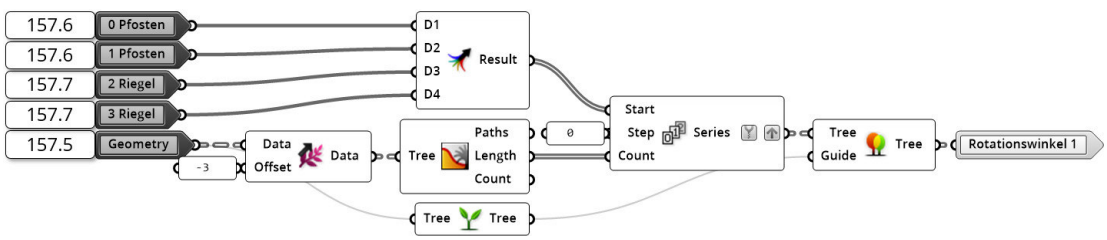
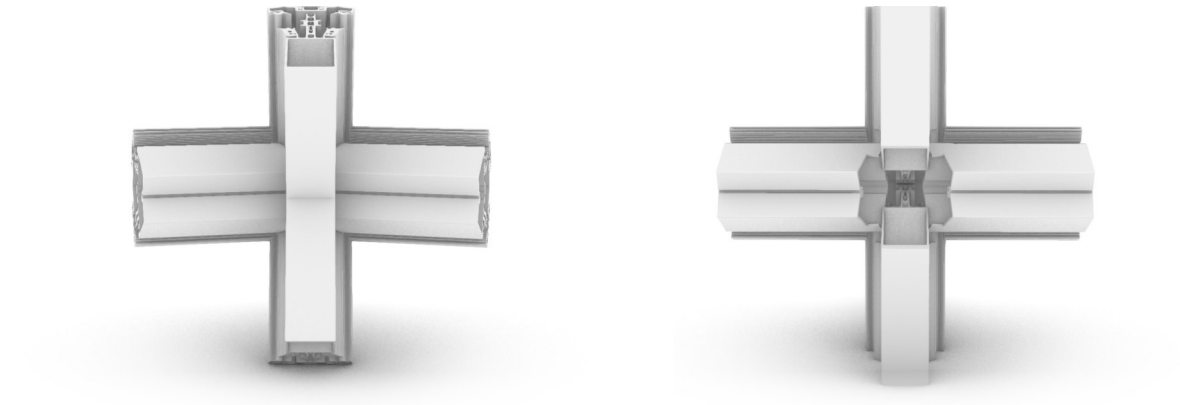


Abbildung 159: Das „Cluster: DatenZuDatenstruktur“
Quelle: In Anlehnung an Grasshopper

de des Clusters werden die Eingabeinformationen genutzt, um Strecken mit der „Line“-Komponente zu erzeugen. Diese Linien können als Achsen in der „Rotate Axis“-Komponente genutzt werden.

Bei der Umkehrung der Rotation ist es wichtig, die vormals zweite Rotation als erstes rückgängig zu machen, da ansonsten die Profile falsch rotiert werden. Die Auswirkungen dieser beiden Rotationen sind in Abbildung 160 gegeben. Dabei wird verdeutlicht, dass sich die Profile am kartesischen Koordinatensystem orientieren und somit orthogonal zueinanderstehen.

Im nächsten Schritt müssen die in 5.2.4 geöffneten variablen Riegelprofile wieder geschlossen werden. Hierfür werden, wie in Abbildung 161 zu sehen, die betroffenen Bauteilkomponenten aus der gesamten Datenstruktur mithilfe der „Tree Split“-Komponenten herausgefiltert und in die einzelnen Bauteilhälften aufgeteilt. Zum einen wird die Datenstruktur der Geometrie 159.1 in die einzelnen Bauteilhälften der Riegel aufgeteilt. Hierzu werden drei



(a) (b)

Abbildung 160: Profile des Fassadenknotens vor (a) und nach (b) der Umkehrung der Rotationen
Quelle: Screenshot aus Rhinoceros 6

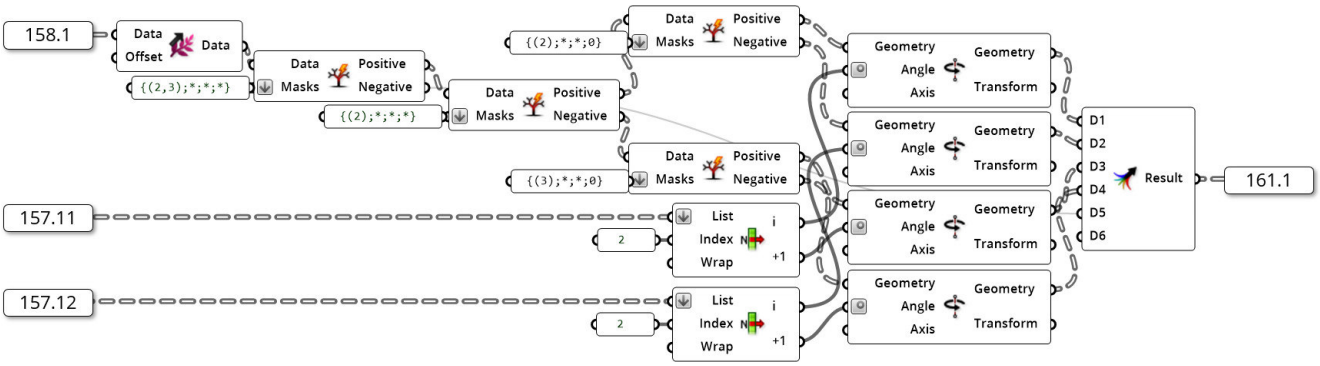


Abbildung 161: Schließen der variablen Riegelprofile
Quelle: In Anlehnung an Grasshopper

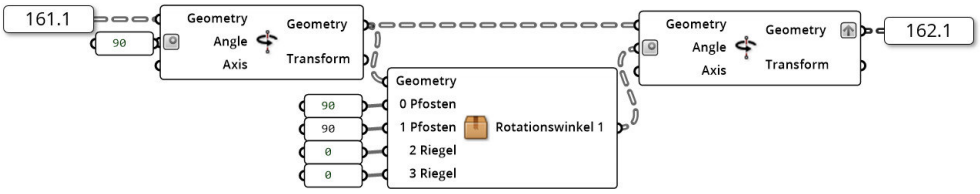


Abbildung 162: Ausrichtung der Profile an der x-Achse
Quelle: In Anlehnung an Grasshopper

„Tree Split"-Komponenten verwendet. Zum anderen können die beiden Winkel „LeftAngle" 157.11 und „RightAngle" 157.12 für die Rotation der einzelnen Bauteilhälften genutzt werden, sodass diese wieder parallel zum kartesischen Koordinatensystem sind. Die hierfür benötigten Rotationsachsen sind direkt in den „Rotate Axis"-Komponenten gespeichert. Die rotierten Profile werden in die bauteilorientierte Datenstruktur gebracht und als 161.1 weiterverwendet.

Da zu diesem Zeitpunkt alle Bauteilkomponenten am kartesischen Koordinatensystem ausgerichtet sind und somit orthogonal zueinanderstehen, können die Geometrien an einer Achse orientiert werden. Die gewählte Achse ist hierbei die x-Achse. Somit stellt diese Achse die Ausrichtung der Fertigungsanlage dar. Um alle Profile entlang dieser Achse zu positionieren, ist es zunächst notwendig, alle Geometrien in die x-y-Ebene zu rotieren. Hierzu werden alle Bauteilkomponenten um 90 ° um die x-Achse rotiert. Dadurch sind die beiden Riegelprofile richtig entlang der x-Achse orientiert. Somit müssen die beiden Pfostenprofile um weitere 90 ° um die z-Achse rotiert werden. Wie in Abbildung 162 dargestellt ist, werden diese beiden Rotationen mit „Rotate Axis"-Komponenten durchgeführt. So sind alle Geometrien 162.1 entlang der x-Achse orientiert.

Als nächsten Schritt müssen die Profile richtig im 3-D-Raum positioniert werden. Hierfür sind die x-, y- und z-Koordinaten anzupassen. Um diese Positionierung vorzunehmen, werden die ausgerichteten Geometrien in Boxen umgewandelt. Dazu kann man die „Bounding Box"-Komponente verwenden. Mithilfe dieser Boxen können die räumlichen Ausprägungen der Profile ermittelt werden. Es ist von Vorteil, die Eckpunkte der Boxen mit der „Deconstruct Brep"-Komponente zu erzeugen. So können mit der „Bounds"-Komponente die minimalen und maximalen Koordinaten bestimmt werden. Dieser Vorgang ist in Abbildung 163 gegeben. Mithilfe dieser Punkte können die Anpassungen an der Position der Geometrien vorgenommen werden. Für die Ausrichtung der x-Koordinate soll für eine Seite des Profils gelten: x = 0. Um weitere Energie für das Schneiden einzusparen, können die gegenüberliegenden Profile mit je drei Schnitten erzeugt werden. Hierfür müssen die Schnitte im Knotenäußeren bei x = 0 liegen, da sich die Schnitte innerhalb einer Ebene befinden. Um dies zu erreichen, muss eines der beiden Profile um den Maximalwert der x-Koordinaten und das Gegenüberliegende um den Minimalwert verschoben werden. Die Auswahl, wel-

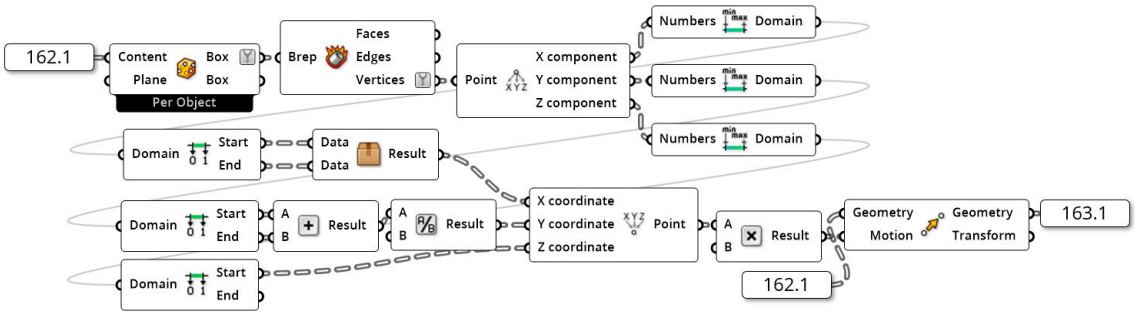


Abbildung 163: Ausrichtung der Profile an der x-Achse
Quelle: In Anlehnung an Grasshopper

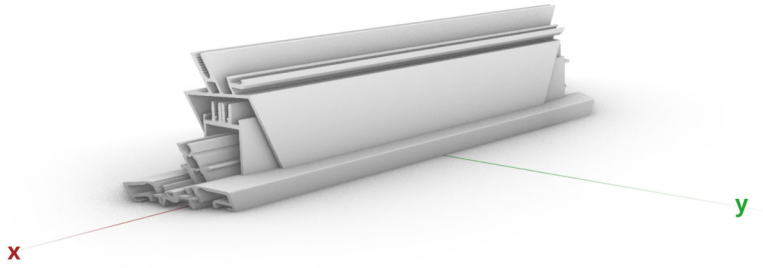


Abbildung 164: Ausrichtung der Profile an der x-Achse
Quelle: In Anlehnung an Rhinoceros 6

ches der beiden Profile um den Maximal- beziehungsweise um den Minimalwert verschoben werden muss, wird in dem „Cluster: Min/Max“ entschieden. Für die y-Koordinate soll gelten, dass die Profile mittig in der Fertigungsanlage positioniert sind. Hierfür kann der Mittelpunkt aus dem Minimal- und dem Maximalwert berechnet werden. Für die z-Koordinate gilt, dass für den untersten Punkt einer Bauteilkomponente gilt z = 0. Somit muss die Geometrie in z-Richtung um den Minimalwert verschoben werden. Die Verschiebungsvektoren werden abschließend mit dem Wert -1 multipliziert und zur Verschiebung der Bauteilkomponenten genutzt. Durch diese Maßnahmen können die Geometrien wie in Abbildung 164 positioniert werden. Hierbei gilt zu beachten, dass neben den Schnitten an den Randbereichen ein dritter Schnitt durch die y-z-Ebene zu führen ist. Dieser ist in der Abbildung 164 nicht dargestellt.

7.3 Generierung der Schnittinformationen

Da die Geometrien der Bauteile für die Fertigungsanlage positioniert sind, kann folgend damit begonnen werden, aus den Geometrien die notwendigen Informationen für die Erzeugung des G-Codes zu extrahieren. Dazu wird die Geometrie 163.1 mit der „Deconstruct Brep“-Komponente in einzelne Punkte zerlegt. Die Umsetzung ist in Abbildung 165 gegeben. Für diese Punkte wird eine Maske erstellt, sodass die beiden Stirnseiten der Profile voneinander getrennt werden. Dabei kann die Tatsache genutzt werden, dass die Punkte der einen Stirnseite alle eine Koordinate von x = 0 aufweisen und alle anderen Punkte die zweite Seite ergeben. So kann mit der Überprüfung der x-Koordinate eine Maske erzeugt werden, die die beiden Seiten trennt. Hierfür ist für das Knoteninnere ein Wertebereich von - 0.05 bis 0.05 zulässig. Mit der „Dispatch“-Komponente können so zwei Listen 165.1 und 165.2 erzeugt werden, die jeweils die Punkte einer Stirnseite beinhalten.

Die beiden Punktlisten 165.1 und 165.2 werden in Abbildung 166 zu Polylines verknüpft und anschließend wird die Datenstruktur bereinigt, sodass sie bauteilorientiert vorliegt. Bevor die Polylines gebaket werden, hat der Nutzer die Möglichkeit in den Dropdown-Menüs 157.1 - 157.4 die zu bakenden Elemente auszuwählen. Nachdem dies erfolgt ist, kann der Bake-Vorgang durch das Betätigen des Buttons „Bake - Schnitte“ 157.13 gestartet werden. Beispielhaft werden folgend die Schnittinformationen des Pfostens 0 für die Bauteilkomponenten Tragprofil, Press- und Blendleiste gezeigt. In Abbildung 167 (a) sind die Schnittkur-

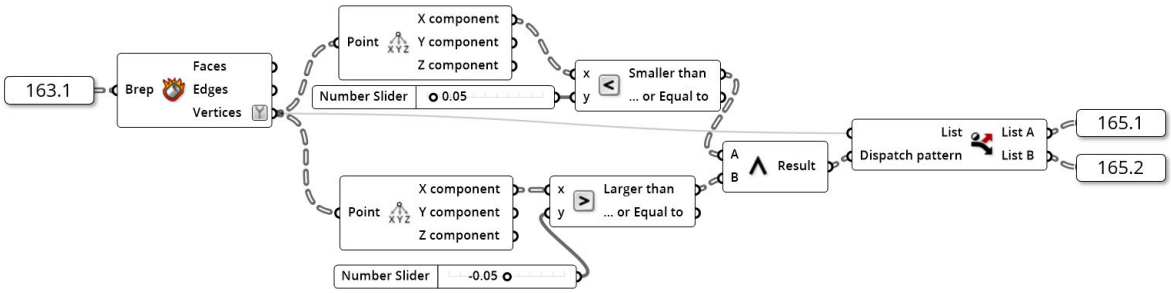


Abbildung 165: Filtern der Schnittkurvenpunkte
Quelle: In Anlehnung an Grasshopper

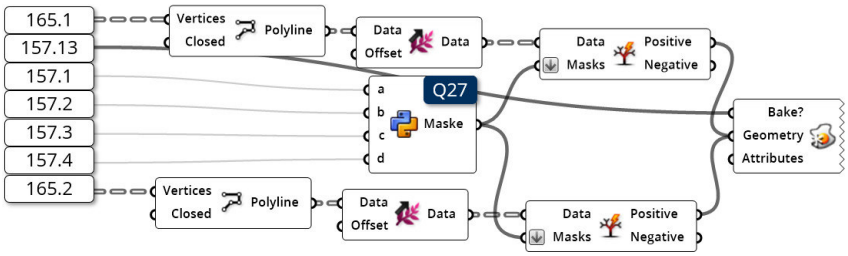
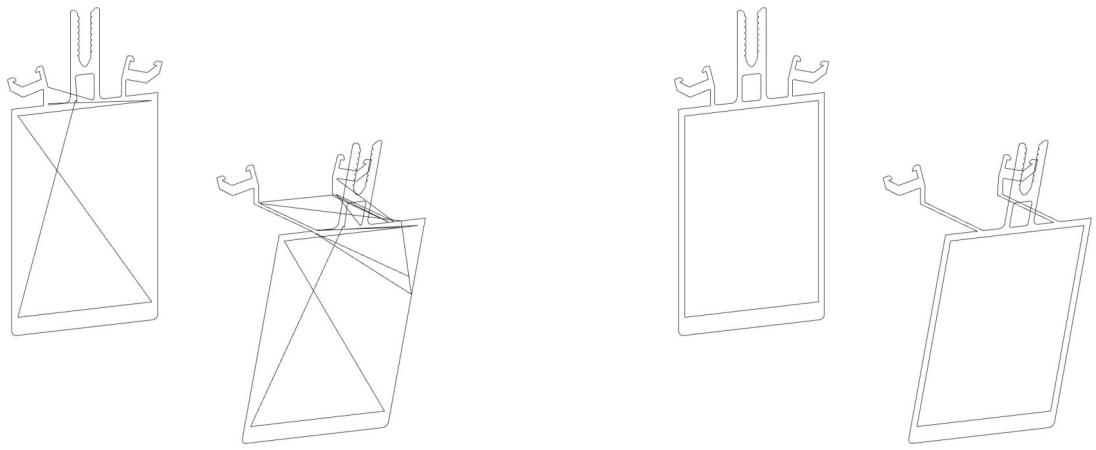


Abbildung 166: Baken der Schnittgeometrien
Quelle: In Anlehnung an Grasshopper



(a) (b)

Abbildung 167: Schnittgeometrien nach dem Baken (a) und nach dem Korrigieren (b)
Quelle: Screenshot aus Rhinoceros 6

ven gegeben. Dabei ist deutlich zu erkennen, dass die Polylines, nicht perfekt geschlossen sind, da einige der Punkte nicht richtig sortiert sind. Jedoch ist in Abbildung 167 (b) erkennbar, dass die Geometrien sich händisch in Rhinoceros 6 schnell bereinigen lassen, sodass geschlossene Polylines vorliegen.

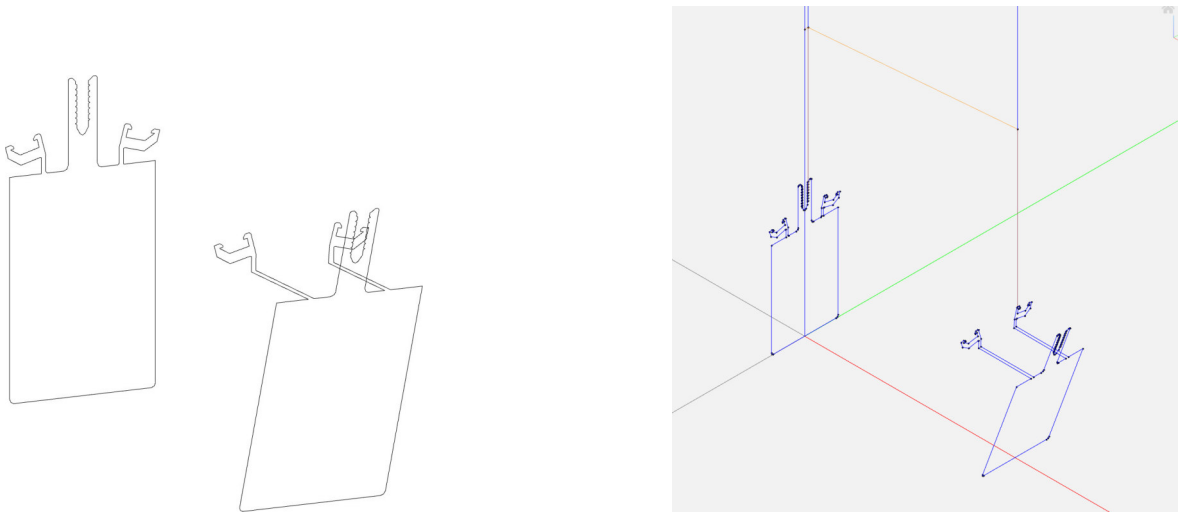
7.4 Generierung des G-Codes

Um eine vollständige Automatisierung des Fertigungsprozesses eines parametrischen Fassadenknotens zu ermöglichen, muss eine Fertigungsanlage auf den Informationen des digitalen Modells basieren. Die notwendigen Vorbereitungen wurden in den Kapiteln 7.2 und 7.3 behandelt. In diesem Kapitel soll gezeigt werden, dass die in Abbildung 167 (b) gezeigte Polyline zur Generierung eines G-Codes geeignet ist. Dieser kann potenziell zur Steuerung einer solchen Fertigungsanlage genutzt werden. Eine explizite Eignung für die Fertigungsanlage „Schüco Maschine DC 500“ ist zu prüfen. Ein Ansatz zur Nutzung des G-Codes ist die Verwendung einer CAM-Software. Da das Ziel dieser Diplomarbeit nicht die Programmierung einer solchen Anlage ist, wird an dieser Stelle nicht weiter auf diesen Aspekt eingegangen. An dieser Stelle soll jedoch aufgezeigt werden, dass es möglich ist, aus dem digitalen Modell des Fassadenknotens einen G-Code zu generieren.

Zu diesem Zweck soll die in Abbildung 167 (b) dargestellte Kurve des Tragprofils verwendet werden. Die Schnittkanten bestehen in der Abbildung aus einer inneren und einer äußeren Kurve. Da die Fertigungsanlage nur einen Schnitt machen muss, wird die innere Kurve entfernt, sodass nur die äußere Kontur erhalten bleibt. Diese ist in Abbildung 168 dargestellt.

Für die Erzeugung des G-Codes können die Koordinaten der Punkte der Polyline als anzusteuernde Positionen verwendet werden. Dieser Vorgang kann prinzipiell in Grasshopper umgesetzt werden, indem die jeweiligen Koordinaten als String ausgegeben werden. Die hierfür notwendige Syntax ist in Kapitel 7.1.2 erläutert. Ein alternativer Ansatz ist die Verwendung von Skripten die den G-Code aus der Geometrie erzeugen. Rhinoceros 6 bietet hierfür unter anderem die Möglichkeit an Skripte in den Sprachen Python und Visual Basic zu nutzen. Für den Anwendungsfall eine 3-D-Kurve in G-Code umzuwandeln, hat der Nutzer „mbele“ am 19.04.2013 in einem Forum der Firma Newfangled Solutions LLC ein passendes Visual Basic Skript geteilt[6]. Mithilfe dieses Skripts konnte die Kurve in einen G-Code umgewandelt werden. Dieser ist in den Anhängen A3 und D7 (digital) gegeben. Mithilfe der Website ncviewer.com wurde der erzeugte Code erfolgreich validiert. Somit konnte gezeigt werden, dass die Daten eines digitalen Modells potenziell zur Steuerung von CNC-Maschinen geeignet ist.

An dieser Stelle ist anzumerken, dass weitere Entwicklungsarbeit in die Ausarbeitung der Schnittstelle zwischen digitalem Modell und Fertigung gesteckt werden muss. Es ist aber auch anzumerken, dass die durch den hohe LoD das digitale Modell auch zur manuellen Steuerung einer CNC-Maschine verwendet werden kann. Wie in Abbildung 169 dargestellt ist, können exakte Werte aus den digitalen Modellen der Bauteilkomponenten gemessen werden. Durch diese Datenbasis kann ein Maschinenbediener die CNC-Maschine manuell programmieren.



(a) (b)

Abbildung 168: Schnittkurve des Pfostentragprofils (a) und Validierung des G-Codes
Quelle: Screenshot aus Rhinoceros 6 (a) und Screenshot von ncviewer.com (b)

[6]Der Link hierzu kann dem Literaturverzeichnis unter dem Eintrag mbele (2013) entnommen werden.

7.5 Vergleich von konventionellen und digitalen Produktionsmethoden

Zum Abschluss dieser Arbeit soll ein Vergleich zwischen konventionellen und digitalen Produktionsmethoden zur Errichtung von parametrischen Fassaden in Pfosten-Riegel-Bauweise gegeben werden. Die meisten dieser Aspekte wurden in den vorangestellten Kapiteln erläutert und sollen an dieser Stelle kurz zusammengefasst werden. Außerdem soll ein kurzes Konzept einer möglichen Umsetzung einer parametrischen Pfosten-Riegel-Fassade gegeben werden.

Vorteilhaft ist an konventionellen Methoden, dass sie derzeit im Vergleich zu den digitalen Verfahren sehr weit verbreitet sind, sodass es eine Vielzahl an Firmen gibt, die zumindest plane Pfosten-Riegel-Fassaden ausführen können. Zudem können baustellenseitig Toleranzen und Differenzen zwischen Soll der Planung und Ist der Ausführung ausgeglichen werden.

Wie Strauß (2013: S. 114) beschreibt, ist die konventionelle Fertigung von komplexen Fassadenknoten geprägt von Ungenauigkeiten. Er beschreibt, dass die meisten Knoten in solch einer Fassade nur durch präzise Schnitte hergestellt werden können. Da mit konventionellen Methoden solche Schnitte nicht mit der benötigten Genauigkeit ausgeführt werden können, entstehen in den Verbindungsstellen häufig ungewollte Leckagen, die nachträglich abgedichtet werden müssen. Abbildung 164 verdeutlicht die Komplexität der Schnittführung für einen Knotenpunkt innerhalb einer Pfosten-Riegel-Fassade.

Des Weiteren wird durch die Trennung der ungestörten Bereiche von den komplex auszuführenden Knotenpunkten die Vorfertigung letzterer ermöglicht. So kann die Produktion wetterunabhängig und mit ausreichender Genauigkeit erfolgen. Damit können die Kosten,

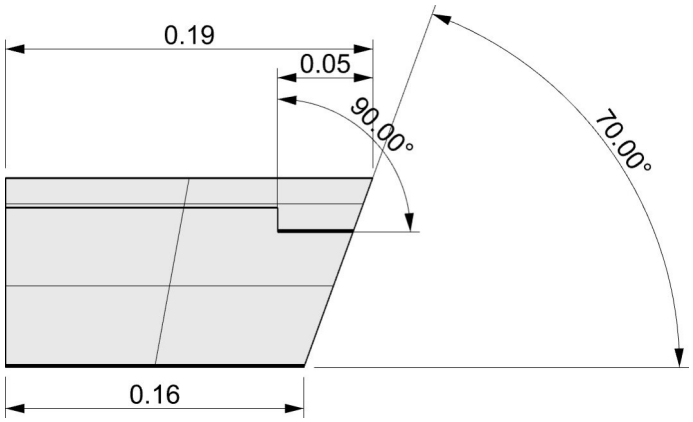


Abbildung 169: Nutzung des digitalen Modells zur Fertigung
Quelle: Screenshot aus Rhinoceros 6

Montagezeiten und der Personalbedarf auf der Baustelle reduziert werden. Durch eine vollständige Digitalisierung des Entwurfs-, Planungs- und Fertigungsprozesses besteht das Potenzial die Komplexität einer Fassade weitgehend Kosten unabhängig zu steigern.

Bis zu diesem Punkt sind jedoch noch einige Hürden zu nehmen, wie etwa die Möglichkeit einer vor-Ort-Robotik, die in der Lage ist Fassaden zu montieren. Im Fall der praktischen Ausarbeitung dieser Arbeit stellt zudem die Schnittstelle zwischen dem Planungs- und dem Fertigungsprozess noch eine Herausforderung dar, sodass die digitalen Informationen direkt zur Fertigung genutzt werden können. Eine derzeitige Umsetzung einer parametrischen Pfosten-Riegel-Fassade kann potenziell durch eine Kombination aus digitaler Vorfertigung und konventioneller Montage erzielt werden. Mithilfe der digitalen Methoden können die Knotenpunkte der Fassade mit hoher Präzision vorgefertigt werden. Das digitale Modell kann dabei genutzt werden um exakte Angaben über die Schnittführung zu geben, sodass eine Programmierung einer Fertigungsanlage möglich ist. Durch die Ausarbeitung von Anschlusspunkten an standardisierte Fassadenprofile, können die Knoten baustellenseitig zu einem gesamten Fassadensystem montiert werden. Dazu muss ein Verbund zwischen Fassadenknoten und den anschließenden Profilen hergestellt werden. Ein Ansatz hierfür ist es, die herstellerseitig bereitgestellten Einschubprofile zu nutzen. Diese können derzeit bereits zum Verbinden von Pfostenprofilen genutzt werden.

8 Schlussbetrachtung

8.1 Zusammenfassung der Arbeit

Im Rahmen dieser Diplomarbeit konnte aufgezeigt werden, dass digitale Produktionsmethoden potenziell für die Errichtung von Bauteilen eines Gebäudes geeignet sind. Durch die neuen Verfahren kann eine Effizienzsteigerung im Bauwesen erzielt werden. Die Vorteile liegen neben der Senkung der Kosten und Zeit auch darin, die Präzision von komplexen Geometrien zu optimieren. So können Fassaden die sich an Freiformflächen orientieren wind- und schlagregendicht ausgeführt werden.

Es wird verdeutlicht, dass auch bereits bestehende Pfosten-Riegel-Fassadensysteme für die Errichtung solcher Fassaden nutzbar sind. Die Arbeit beschreibt, dass die Systeme in kleinen Bereichen erweitert werden müssen, damit alle Geometriestellungen umgesetzt werden können. Für die Umsetzung einer parametrischen Pfosten-Riegel-Fassade ist es von Nöten, die komplexen Knotenpunkte der Profile werkseitig zu produzieren. Durch die Vorfertigung kann die benötigte Präzision erzielt werden, damit die Funktionsweise einer Fassade erfüllt wird. Die Montage der Fassadenprofile und der Fassadenknoten erfolgt dann baustellenseitig.

Eine derzeitige Umsetzung der Digitalisierung im Bauwesen besteht darin kleine, hochkomplexe Bauteile mithilfe von digitalen Produktionsverfahren herzustellen. Mithilfe konventioneller Methoden kann die digitale Produktion zur Errichtung von Bauwerken ergänzt werden.

Des Weiteren konnte in der praktischen Ausarbeitung, in Verbindung mit den Erkenntnissen der vorangestellten Projektarbeit die Nutzung von digitalen Daten zur Erzeugung von digitalen Fassadenknoten validiert werden. Somit wird gezeigt, dass digitale Methoden in der Lage sind, aus einem digitalen Modell alle notwendigen Informationen zu extrahieren und diese für die Produktion zu nutzen. Daraus resultiert ein enormes Potenzial für weitere Effizienzsteigerungen im Bauwesen.

8.2 Ausblick

In dieser Arbeit konnten die Grenzen einer konventionellen Pfosten-Riegel-Fassade zur Errichtung von parametrischen Fassaden ermittelt werden. Es konnte gezeigt werden, dass bereits einige Möglichkeiten vorhanden sind, solch ein Konzept umzusetzen, indem ein bestehendes System um einige weitere Bauteile erweitert wird, wie zusätzliche variable Riegelprofile und Einschubprofile für variable Riegelprofile. Daher sollte für die Weiterentwicklung einer parametrischen Pfosten-Riegel-Fassade an einer Vervollständigung der bestehenden Systeme oder an der Entwicklung eines neuen Systems gearbeitet werden. Das System sollte in der Lage sein deutlich mehr Geometriestellungen des Fassadenknotens zu ermöglichen.

Die vor-Ort-Montage stellt bei einer parametrischen Pfosten-Riegel-Fassade eine große Herausforderung dar. Derzeit ist nur eine konventionelle, zeit- und arbeitskraftintensive Montage möglich. Das Konzept einer parametrischen Pfosten-Riegel-Fassade ist von der Entwicklung einer baustellenseitigen Robotik abhängig, die die Montage der parametrischen Fassade ausführen kann. Bei der Umsetzung einer solchen Robotik ist darauf zu achten, dass eine ausreichende Genauigkeit erzielt werden kann.

Des Weiteren ist eine Schnittstelle zwischen dem digitalen Modell und einer Fertigungsanlage zu schaffen. Durch solch eine Schnittstelle können die Daten des Modells direkt für die Fertigung genutzt werden. Dies bringt den Vorteil mit sich, dass die Produktion automatisiert werden kann. Dadurch können komplexere Knoten mit gleichem Arbeitsaufwand hergestellt werden.

Abschließend lässt sich über das Thema der digitalen Produktion zur Errichtung von parametrischen Fassadenknoten sagen, dass ein großes Potenzial zur Effizienzsteigerung besteht. Dieses Potenzial ist nicht nur in der Pfosten-Riegel-Bauweise vorhanden, wie es in dieser Diplomarbeit aufgezeigt wurde, sondern auch in anderen Fassadensystemen. Möglicherweise führt der Ansatz der digitalen Produktion von Fassaden auch zu einem neuartigen Fassadensystem, das auf die Anforderungen und Möglichkeiten der Digitalisierung spezialisiert ist. Beispielsweise könnten mithilfe von baustellenseitiger Robotik konventionelle Montagemethoden verändert werden, sodass komplexere Anschlusspunkte zeit- und kostensparend sowie mit hoher Präzision errichtet werden können. Mehr gestalterische Freiheiten und eine Steigerung des Wohnkomforts – beispielsweise aufgrund von reduzierten Wärmebrücken – wären die Folge.

8.3 Thesen zur Diplomarbeit

„Entwicklung einer Methode zum Einsatz der digitalen Fertigung im Bauwesen am Beispiel eines parametrischen Fassadenknotens“

cand. ing. Adrian Schubert

These 1: „Im Bauwesen besteht ein großes Potenzial, mithilfe der digitalen Produktion Effizienzsteigerungen zu erzielen.“

These 2: „Die digitale Produktion kann für die Errichtung von Gebäuden genutzt werden.“

These 3: „Parametric Design eignet sich für die Erzeugung von Bauteilen mit einem hohen Level of Detail.“

These 4: „Ein in Parametric Design ausgeführter Entwurf kann durch einen Nutzer intuitiv angepasst und modifiziert werden. Diese Änderungen können direkt auf die Produktion von Bauteilen angewandt werden.“

These 5: „Daten aus einem digitalen Gebäudeentwurf können für die digitale Produktion von Bauteilen genutzt werden.“

These 6: „Durch geeignete Konventionen kann eine Datenstruktur Informationen auf die gespeicherten Datensätze übertragen. Diese Informationen werden dabei nicht explizit gespeichert. “

These 7: „Die Errichtung von Bauteilen mit großem Volumen sind weniger für additive Fertigungsverfahren geeignet als Kleinvolumige.“

Literatur

Aksamija, Ajla et al. (2011). “Parametric Control of BIM Elements for Sustainable Design in Revit - Linking Design and Analytical Software Applications through Customization”. In: *in: Perkins+Will Research Journal Vol. 03.01*, 31 bis 45. URL: https://www.brikbase.org/sites/default/files/PWRJ_Vol0301.pdf (besucht am 08. 09. 2019).

American Institute of Architects (2013). *G202-2013, Building Information Modeiling Protocol Form*. URL: https://www.aiacontracts.org/contract-documents/19016-project-bim-protocol (besucht am 23. 09. 2019).

Baitinger, Martin, Eric Busse und Nils Mauser (2006). “Nachweiskriterien von mechanischen Verbindungen bei Fassadenkonstruktionen in Pfosten- und Riegelbauweise”. Deutsch. In: *Stahlbau 75, Heft 6*, S. 446–453. DOI: 10.1002/stab.200610047.

BIMForum (Sep. 2018). *Level of Development (LOD) Specification Part I and Commentary. For Building Information Models and Data*. Version 2018. URL: https://bimforum.org/wp-content/uploads/2018/09/BIMForum-LOD-2018_Spec-Part-1_and_Guide_2018-09.pdf (besucht am 23. 09. 2019).

Blum, Norbert (2013). *Algorithmen und Datenstrukturen*. Berlin, Boston: Oldenbourg Wissenschaftsverlag, ISBN: 9783486719666.

Cheret, Peter et al. (2015). *Handbuch und Planungshilfe - Baukonstruktion und Baupyhsik*. DOM publishers. ISBN: 978-3-86922-422-0.

Dietzfelbinger, Martin, Kurt Mehlhorn und Peter Sanders (2014). *Algorithmen und Datenstrukturen - Die Grundwerkzeuge*. Springer Vieweg. ISBN: 978-3-642-05472-3. DOI: 10.1007/978-3-642-05472-3.

DIN 66025-1:1983-01 (1983). *Programmaufbau für numerisch gesteuerte Arbeitsmaschinen - Allgemeine*. Berlin: Beuth Verlag GmbH.

DIN EN 13830:2015-07 (2015). *Vorhangfassaden - Produktnorm; Deutsche Fassung EN 13830:2015*. Berlin: Beuth Verlag GmbH.

Fouad, Nabil A. (2013). *Lehrbuch der Hochbaukonstruktionen*. Wiesbaden. URL: http://slubdd.de/katalog?TN_libero_mab215950873.

Heinze GmbH (o.D.). URL: https://www.baunetzwissen.de/glas/tipps/news-produkte/nahezu-farblose-sonnenschutzbeschichtung-3912407.

Heinze GmbH (2019a). *BaunetzWissenGlas - Low-E-Glas*. URL: https://www.baunetzwissen.de/glossar/l/low-e-glas-51407 (besucht am 01. 09. 2019).

Heinze GmbH (2019b). *Floatglas*. URL: https://www.baunetzwissen.de/glas/fachwissen/basisglaeser/floatglas-159089 (besucht am 03. 09. 2019).

Herzog, Thomas, Roland Krippner und Werner Lang (2004). *Fassaden Atlas*. Institut für Internationale Architektur-Dokumentation GmbH & Co. KG. ISBN: 9783764370312.

Hestermann, Ulf und Ludwig Rongen (2018). "Pfosten-Riegel-Fassaden (PRF)". In: *Frick/Knöll Baukonstruktionslehre 2*. Wiesbaden: Springer Fachmedien Wiesbaden, S. 573–600. ISBN: 978-3-658-21913-0. DOI: 10.1007/978-3-658-21913-0_7. URL: https://doi.org/10.1007/978-3-658-21913-0_7.

HGG Group (2019a). *3D-Profilierung*. URL: https://www.hgg-group.com/de/ueber-hgg/3d-profilierung/ (besucht am 17. 09. 2019).

HGG Group (2019b). *MPC | Accurate 3D Cutting Machine for Hollow Sections*. URL: https://www.hgg-group.com/wp-content/uploads/2015/09/MPC-product-sheet-v2_allcompany.pdf (besucht am 17. 09. 2019).

Hofmann, Johann (2017). *Die digitale Fabrik Auf dem Weg zur digitalen Produktion*. Neuerscheinung. Berlin: VDE Verlag. ISBN: 3800744899. URL: http://slubdd.de/katalog?TN_libero_mab216617485.

Kief, Hans B., Helmut A. Roschiwal und Karsten Schwarz (2017). *CNC-Handbuch. CNC, DNC, CAD, CAM, FFS, SPS, RPD, LAN, CNC-Maschinen, CNC-Roboter, Antriebe, Energieeffizienz, Werkzeuge, Industrie 4.0, Fertigungstechnik, Richtlinien, Normen, Simulation, Fachwortverzeichnis*. 30., überarbeitete Auflage. München: Carl Hanser Verlag. ISBN: 978-3-446-45265-7.

Knaack, Ulrich et al. (2007). *Fassaden - Prinzipien der Konstruktion*. 1. Auflage. Birkhäuser Verlag AG. ISBN: 978-3-7643-7961-2.

Knaack, Ulrich et al. (2014). *Fassaden - Prinzipien der Konstruktion*. 3. überarbeitete Auflage. Birkhäuser Verlag AG. ISBN: 978-3-03821-094-8.

Koch, Wilfried (2014). *Baustilkunde - Das Standardwerk zur europäischen Baukunst von der Antike bis zur Gegenwart*. 32. Auflage. München: Prestel Verlag. ISBN: 978-3-7913-4997-8.

mbele (19. Apr. 2013). *Rhino: 2D path to g-code*. URL: https://www.machsupport.com/forum/index.php?topic=24361.0 (besucht am 26. 09. 2019).

Mode Lab (2015). *Der Grasshopper Primer (DE)*. URL: https://www.modelab.is/grasshopper-primer/ (besucht am 25. 08. 2019).

Rodricks, Robin und Andrew Heumann (2018a). *rhino.github.io - Alle Grasshopper Komponenten*. URL: https://rhino.github.io/#System (besucht am 25. 08. 2019).

Rodricks, Robin und Andrew Heumann (2018b). *rhino.github.io - Path Mapper*. URL: http://rhino.github.io/components/sets/pathMapper.html (besucht am 25. 08. 2019).

Rodricks, Robin und Andrew Heumann (2018c). *rhino.github.io - Split Tree*. URL: https://rhino.github.io/components/sets/splitTree.html (besucht am 25. 08. 2019).

Rutton, David (3. Nov. 2013). *DataTree selection rules*. URL: https://www.grasshopper3d.com/forum/topics/datatree-selection-rules?page=6&commentId=2985220%3AComment%3A1888369&x=1#2985220Comment1888369 (besucht am 25. 08. 2019).

Rutton, David (20. Jan. 2015). *They Why and How of Data Trees*. URL: https://www.grasshopper3d.com/forum/topics/the-why-and-how-of-data-trees (besucht am 25. 08. 2019).

Saenger, Anton (Sep. 2019). *CNC Programmierung mit G-Code*. URL: https://www.precifast.de/cnc-programmierung-mit-g-code/ (besucht am 21. 09. 2019).

Samiee, Khashayar (2019). "Tracking trajectories and breaking down trajectory motions of a one-arm robot to assemble low LOD façade elements". Projektarbeit. TU Dresden - Institut für Bauinformatik.

Schiffer, Stefan (2019). *Visuelle Programmierung - Potential und Grenzen*. URL: https://pdfs.semanticscholar.org/1b20/19881a6fd9c9dff1e69201eccd679f09d27f.pdf (besucht am 25. 08. 2019).

Schittich, Christian et al. (2006). *GlasbauAtlas*. 2., überarb. u. erw. Aufl. München: Institut für Internationale ArchitekturDokumentation. ISBN: 3034615531. URL: http://slubdd.de/katalog?TN_libero_mab214004117.

Schmid, Manfred (2015). *Additive Fertigung mit Selektivem Lasersintern (SLS). Prozess- und Werkstoffüberblick*. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-12289-8. DOI: 10.1007/978-3-658-12289-8.

Schüco International KG (Aug. 2011). *architect information 7 - Aluminium-Profilfassaden und Lichtdächer FW 50+ und FW 60+*. URL: http://www.eas-usa.com/Schuco/PDFFacade/Volume7_FW_CurtainWall.pdf (besucht am 01. 09. 2019).

Schüco International KG (2013). *Schüco FW 50 + - Fertigungsunterlagen A-D*.

Schüco International KG (Okt. 2016a). URL: https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&ved=2ahUKEwjYsZeX_dfkAhVOsKQKHZe0DDcQFjAKegQIARAC&url=https%3A%2F%2Fwww.schueco.com%2Fweb2%2Fasset%2Fde-en%2Ffabricators%2Fmf_machinery%2Fcnc_machining%2F24147108%2Fprospect_dc_500.pdf&usg=AOvVaw1wEAZD8n-2czDlmmmOfno2 (besucht am 17. 09. 2019).

Schüco International KG (Juli 2016b). *Schüco Parametric System - 12 Architekten Information*. URL: https://www.schueco.com/web2/blob/21397700/

7420326d8195e39cc4f1e7c7b5a7a613 / 25287 - ainfo12 - schueco - parametric - system - data.pdf (besucht am 01. 09. 2019).

Schuh, Günther, Fritz Klocke und Stephan Ripp (2002). *Integration als Grundlage der digitalen Fabrikplanung*. Bd. 144. 11/12. Düsseldorf: Springer-VDI-Verl., S. 48–51. URL: http://publications.rwth-aachen.de/record/146235.

Stabalux GmbH (Juli 2019). *Stabalux AL - Verarbeitungsrichtlinien*. URL: https://media.stabalux.com/uploads/PDF/de/1.0%20Stabalux_AL_DE.pdf (besucht am 03. 09. 2019).

Strauß, Holger (2013). *AM Envelope - The potential of Addivite Manufacturing for façade construction*. URL: https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&ved=2ahUKEwjorLDEuLbkAhUSy6YKHbBbDhUQFjAKegQIAhAC&url=https%3A%2F%2Fjournals.open.tudelft.nl%2Findex.php%2Fabe%2Farticle%2Fdownload%2Fstrauss%2F476%2F&usg=AOvVaw3vOE5uw1VNwpWkngEIHxyk (besucht am 04. 09. 2019).

Strauß, Holger, Emmer Pfenninger Partner AG und Ulrich Knaack (Juni 2015). "Additive Manufacturing for Future Facades: The potential of 3D printed parts for the building envelope". In: *Journal of Facade Design and Engineering* 3.3-4, S. 225–235. ISSN: 2213-3038. DOI: 10.3233/FDE-150042. URL: https://content.iospress.com/articles/journal-of-facade-design-and-engineering/fde0042 (besucht am 17. 09. 2019).

TU Delft (18. Sep. 2013). *Grasshopper Data Tree Editing*. URL: http://wiki.bk.tudelft.nl/toi-pedia/Grasshopper_Data_Tree_Editing (besucht am 25. 08. 2019).

Weller, Bernhard und Jasmin Fischer (2013). *Untersuchung eines gedämmten Paneels mit integrierter Photovoltaik zur Verwendung in Pfosten-Riegel-Konstruktionen (Gedämmtes PV-Paneel)*. URL: http://www.irbnet.de/daten/rswb/15039013122.pdf (besucht am 26. 08. 2019).

Westkämper, Engelbert et al. (2013). *Digitale Produktion*. Springer, Berlin, Heidelberg. ISBN: 978-3-642-20259-9 (Online). DOI: 10.1007/978-3-642-20259-9.

Anlagenverzeichnis

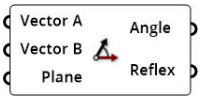
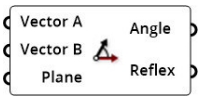
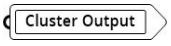
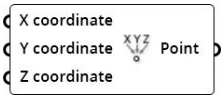
Digital auf der CD enthalten:

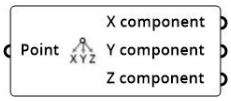
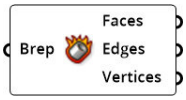
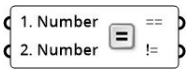
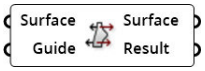
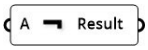
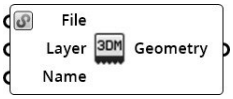
D0	*Schubert00_digitalesVerzeichnis.txt*	Verzeichnis der digitalen Anlagen
D1	*Schubert01_CAD*	Dateiordner mit CAD-Dateien
D2	*Schubert02_Abbildungen*	Dateiordner mit Abbildungen
D3	*Schubert03_Bericht.pdf*	Digitaler Bericht der Diplomarbeit
D4	*Schubert04_parametrischerKnoten.gh*	Grasshopper Definition des parametrischen Fassadenknotens
D5	*Schubert05_GCode.gh*	Grasshopper Definition zur Generierung des G-Codes
D6	*Schubert06_input.xlsx*	Eingabetabelle mit Artikelnummern
D7	*Schubert07_GCode.txt*	G-Code des Pfostentragprofils
D8	*Schubert08_modell.3dm*	Digitales Modell des parametrischen Fassadenknotens
D9	*Schubert09_kurven.3dm*	Digitale Modelle der Schnittkurven

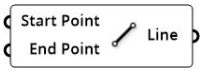
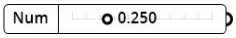
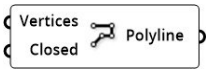
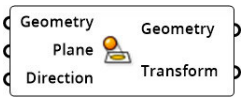
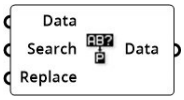
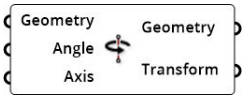
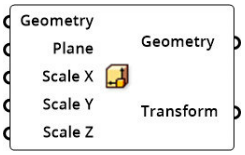
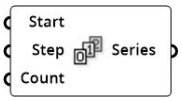
In den Anlagen enthalten:

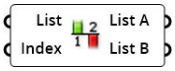
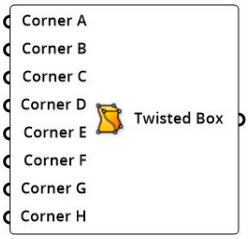
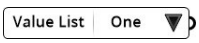
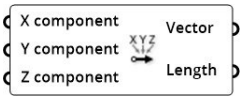
A1:Tab. 5: Übersicht: wichtige Grasshopperkomponenten

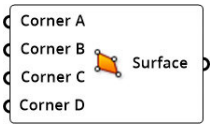
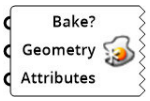
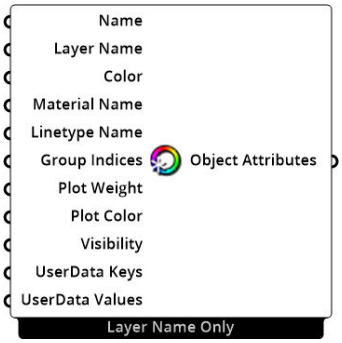
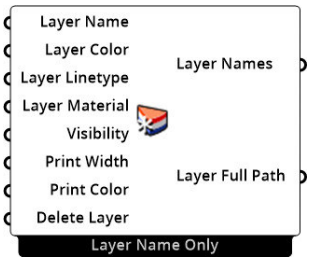
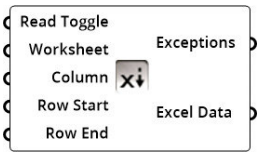
Tab. 5: Übersicht: wichtige Grasshopperkomponenten

Name	Darstellung	Kurzbeschreibung
Grasshopper (ohne Plug-In)		
Absolut		Gibt den Betrag des Inputs zurück
Addition		Addiert zwei oder mehr Werte miteinander
Angle		Gibt den Winkel zwischen zwei Vektoren zurück
Boolean Toggle		Ein Schalter der True oder False ausgibt
Boundary Surfaces		Erzeugt eine Fläche aus einer umgrenzenden Kurve
Bounding Box		Erzeugt eine Box um eine Geometrie
Bounds		Bestimmt den Wertebereich (Domain) des Inputs
Box		Enthält Boxen
Box Morph		Transformiert eine Geometrie auf Grundlage einer Vergleichs- und einer Zielbox
Brep		Enthält Breps
Brep Join		Verbindet mehrere Breps zu einem Brep
Cluster*		Enthält Komponenten
Cluster Input		Übergibt Parameter an ein Cluster
Cluster Output		Gibt Parameter aus einem Cluster aus
Construct Plane		Erzeugt eine Ebene aus zwei Vektoren
Construct Point		Erzeugt einen Punkt (x|y|z)
Cosine		Berechnet den Cosinus (Input standardmäßig in RAD; kann über „Rechtsklick“ auf GRAD umgestellt werden)
Curve		Enthält Kurven
Data		Enthält Daten (ohne weitere Spezifikation)

Data Input		Importiert Daten aus einer anderen Grasshopper-Definition
Data Output		Exportiert Daten für eine andere Grasshopper-Definition
Deconstruct		Zerlegt einen Punkt in seine Koordinaten
Deconstruct Brep		Zerlegt ein Brep in einzelne Flächen, Kanten und Punkte
Deconstruct Domain		Zerlegt einen Wertebereich (Domain) in Min und Max
Degrees		Umwandlung RAD → GRAD
Dispatch		
Division		Dividiert den oberen durch den unteren Wert
Equality		Überprüft zwei Werte auf Gleichheit
Explode Tree		Zerlegt einen Baum in alle seine Äste
Extrude		Erzeugt einen 3-D-Körper aus einer Kurve oder einer Fläche
Flip		Dreht die Normalenrichtung einer Fläche
Gate And		Logisches Und
Gate Not		Logisches Nicht
Gate Or		Logisches Oder
Geometry		Enthält Geometrie
GhPython Script		Diese Komponente kann selbst mit Python programmiert werden
Graft Tree		Fügt eine Hierarchieebene zur Datenstruktur hinzu
Group		Gruppiert alle Elemente in einer Listen
Import 3DM		Importiert Geometrie aus einer .3dm-Datei
Integer		Enthält eine Ganzzahl
Larger Than		Überprüft ob eine Zahl größer als eine andere Zahl ist

Line		Erzeugt eine Linie aus zwei Punkten
List Item		Entnimmt ein Objekt aus einer Liste
Match Tree		Gleicht die Pfade zweier Bäume an
Merge		Führt mehrere Datensätze zusammen
Mirror		Spiegelt eine Geometrie um eine Ebene
Move		Verschiebt eine Geometrie um einen Vektor
Multiplikation		Multipliziert zwei oder mehr Werte miteinander
Number		Enthält eine Fließkommazahl
Number Slider		Ermöglicht das einstellen einer Zahl über einen Schieberegler
Panel		Dient zur Ein- und Ausgabe von Werten oder als „Notizzettel“
Path Mapper		Modifiziert die Datenstruktur
Plane Normal		Erzeugt eine Ebene, wobei der übergebene Vektor dem Normalenvektor entspricht
Point		Enthält Punkte
PolyLine		Erzeugt eine Polyline aus mehreren Punkten
Project Along		Projiziert ein Objekt auf eine Ebene entlang einer Richtung
Replace Paths		Ersetzt Pfade
Rotate Axis		Rotiert eine Geometrie um eine Achse
Scale NU		Skaliert eine Geometrie mit unterschiedlichen Faktoren für unterschiedliche Richtungen
Series		Erzeugt eine Zahlenreihe

Shift List		Verschiebt die Objekte innerhalb einer Liste
Shift Paths		Entfernen oder Hinzufügen von Hierarchieebenen
Simplify Tree		Vereinfacht die Datenstruktur (entfernt nicht benutzte Hierarchieebenen)
Sine		Berechnet den Sinus (siehe Cosine)
Smaller Than		Überprüft ob eine Zahl kleiner als eine andere Zahl ist
Solid Difference		Erzeugt die Boolsche Differenz
Split List		Teile eine Liste an einer definierten Stelle
Split Tree		Teile einen Baum nach einer Maske
Subtraktion		Subtrahiert zwei oder mehr Werte voneinander
Tangent		Berechnet den Tangens (siehe Cosine)
Transform		Transformiert eine Geometrie
Tree Branch		Gibt einen Ast eines Baumes zurück
Tree Statistics		Gibt Metadaten einer Datenstruktur zurück
Twisted Box		Erzeugt eine Twisted Box
Ungroup		Hebt eine Gruppierung auf
Unit Vector		
Value List		Ermöglicht Auswahl aus Liste
Vector		Enthält Vektoren
Vector XYZ		Erzeugt einen Vektor

4Point Surface		Erzeugt ein Drei- oder Viereck
Plug-In: Human		
Bake Geometry		Überträgt eine Geometrie nach Rhinoceros 6
CreateAttributes		Erzeugt Objektattribute
Create/Modify Layers		Erzeugt / Modifiziert Layer in Rhinoceros 6
Plug-In: LunchBox		
Excel Reader LEGACY		Liest eine geöffnete(!) Exceltabelle

* Eigentlich keine Komponente! Kann über „auswählen mehrerer Komponenten → Mittlere Maustaste → Cluster“ erzeugt werden

Quelle: Eigene Darstellung in Anlehnung an Mode Lab 2015 und Rodricks und Heumann 2018a

Quelle der Abbildungen: Screenshots aus Grasshopper

A2: Tabellen der Datei „Schubert06_input.xlsx“

Tab. 6: Wahl InnerProfile

	A	B	C	D	E	F
1	322250	ERROR	ERROR	ERROR	ERROR	ERROR
2	322260	ERROR	ERROR	ERROR	ERROR	ERROR
3	322270	323040_05	323040_10	323040_15	323040_20	323040_25
4	322280	323050_05	323050_10	323050_15	323050_20	323050_25
5	322290	323060_05	323060_10	323060_15	323060_20	323060_25
6	322300	328640_05	328640_10	328640_15	328640_20	328640_25
7	322310	328650_05	328650_10	328650_15	328650_20	328650_25
8	326030	328660_05	328660_10	328660_15	328660_20	328660_25
9	336230	ERROR	ERROR	ERROR	ERROR	ERROR
10	336240	ERROR	ERROR	ERROR	ERROR	ERROR

Tab. 7: Wahl InnerSealing

	A	B	C	D
1	244297	InnerSealing0500+	InnerSealing0010+	InnerSealing1015+
2	244295	InnerSealing0500	InnerSealing0010	InnerSealing1015
3	244293	InnerSealing0500-	InnerSealing0010-	InnerSealing1015-
4	204649	ERROR	ERROR	ERROR
5	224662	ERROR	ERROR	ERROR
6	204553	ERROR	ERROR	ERROR

Tab. 8: Wahl Insulator

	A
1	244327
2	244255
3	244256

Tab. 9: Wahl OuterSealing

	A	B	C	D
1	246476	OuterSealing0500	OuterSealing0010	OuterSealing1015

Tab. 10: Wahl Pressleiste

	A	B	C	D	E	F	G
1	325520	323330	323340	323350	323360	323370	323380
2	161450						

Tab. 11: Wahl Blendleiste

	A	B
1	160620	323390
2	161460	

Tab. 12: Wahl InnerProfileRiegel

	A	B
1	322370	322630
2	322380	322640
3	322460	
4	323840	
5	322390	
6	322400	
7	322410	
8	322420	
9	322430	
10	322440	
11	322450	

Tab. 13: Wahl EinschubprofilePfosten

	A
1	ERROR
2	ERROR
3	322720
4	322730
5	322740
6	322750
7	322760
8	326050
9	336260
10	ERROR

Tab. 14: Wahl EinschubprofileRiegel

	A	B
1	ERROR	VarRiegelA
2	ERROR	VarRiegelB
3	ERROR	
4	Riegel045	
5	Riegel055	
6	Riegel070	
7	Riegel090	
8	Riegel110	
9	Riegel130	
10	Riegel155	
11	Riegel180	

A3: G-Code des Pfostenprofils

```
G49 G40.1 G17 G80 G50 G90 G21 (init)
G0 Z0.2

(...new cut...)
G0 X0 Y0.0025
P0
Z0.592417260882987
F100
G1 Z0.0924(...enter strategy...)
P0
F1000
X0 Y0.002 Z0.0921
X0 Y0.002 Z0.0909
X0 Y0.0025 Z0.0906
X0 Y0.002 Z0.0903
X0 Y0.002 Z0.0891
X0 Y0.0025 Z0.0888
X0 Y0.002 Z0.0885
X0 Y0.002 Z0.0873
X0 Y0.0025 Z0.087
X0 Y0.002 Z0.0867
X0 Y0.002 Z0.0848
X0 Y0.0009 Z0.0828
X0 Y0 Z0.0823
X0 Y-0.0009 Z0.0828
X0 Y-0.002 Z0.0848
X0 Y-0.002 Z0.0858
X0 Y-0.0025 Z0.0861
X0 Y-0.002 Z0.0864
X0 Y-0.002 Z0.0876
X0 Y-0.0025 Z0.0879
X0 Y-0.002 Z0.0882
X0 Y-0.002 Z0.0894
X0 Y-0.0025 Z0.0897
X0 Y-0.002 Z0.09
X0 Y-0.002 Z0.0912
X0 Y-0.0025 Z0.0915
X0 Y-0.002 Z0.0918
X0 Y-0.002 Z0.093
X0 Y-0.0025 Z0.0933
```

```
41  X0 Y−0.002 Z0.0936
42  X0 Y−0.002 Z0.0948
43  X0 Y−0.0025 Z0.0951
44  X0 Y−0.002 Z0.0954
45  X0 Y−0.002 Z0.0966
46  X0 Y−0.0025 Z0.0969
47  X0 Y−0.002 Z0.0972
48  X0 Y−0.002 Z0.0984
49  X0 Y−0.0025 Z0.0987
50  X0 Y−0.0025 Z0.0996
51  X0 Y−0.0035 Z0.1006
52  X0 Y−0.0045 Z0.1006
53  X0 Y−0.005 Z0.1001
54  X0 Y−0.005 Z0.0731
55  X0 Y−0.0053 Z0.0722
56  X0 Y−0.0065 Z0.0716
57  X0 Y−0.0121 Z0.0716
58  X0 Y−0.0126 Z0.0721
59  X0 Y−0.0126 Z0.0773
60  X0 Y−0.0144 Z0.084
61  X0 Y−0.015 Z0.0843
62  X0 Y−0.0163 Z0.084
63  X0 Y−0.0169 Z0.0829
64  X0 Y−0.0167 Z0.0823
65  X0 Y−0.0156 Z0.0826
66  X0 Y−0.0149 Z0.08
67  X0 Y−0.0214 Z0.0783
68  X0 Y−0.0247 Z0.0802
69  X0 Y−0.0235 Z0.0805
70  X0 Y−0.0237 Z0.0811
71  X0 Y−0.0247 Z0.0817
72  X0 Y−0.0265 Z0.0812
73  X0 Y−0.0255 Z0.0776
74  X0 Y−0.0211 Z0.075
75  X0 Y−0.0146 Z0.0768
76  X0 Y−0.0141 Z0.0767
77  X0 Y−0.0141 Z0.0716
78  X0 Y−0.025 Z0.0716
79  X0 Y−0.025 Z0.0016
80  X0 Y−0.0246 Z0.0006
81  X0 Y−0.0237 Z0.0001
82  X0 Y0.0235 Z0.0001
```

```
 83   X0 Y0.0246 Z0.0005
 84   X0 Y0.025 Z0.0016
 85   X0 Y0.025 Z0.0716
 86   X0 Y0.0141 Z0.0716
 87   X0 Y0.0141 Z0.0767
 88   X0 Y0.0146 Z0.0768
 89   X0 Y0.0211 Z0.075
 90   X0 Y0.0255 Z0.0776
 91   X0 Y0.0265 Z0.0812
 92   X0 Y0.0247 Z0.0817
 93   X0 Y0.0237 Z0.0811
 94   X0 Y0.0235 Z0.0805
 95   X0 Y0.0247 Z0.0802
 96   X0 Y0.0214 Z0.0783
 97   X0 Y0.0149 Z0.08
 98   X0 Y0.0156 Z0.0826
 99   X0 Y0.0167 Z0.0823
100   X0 Y0.0169 Z0.0829
101   X0 Y0.0163 Z0.084
102   X0 Y0.015 Z0.0843
103   X0 Y0.0144 Z0.084
104   X0 Y0.0126 Z0.0773
105   X0 Y0.0126 Z0.0721
106   X0 Y0.0121 Z0.0716
107   X0 Y0.0065 Z0.0716
108   X0 Y0.0056 Z0.0719
109   X0 Y0.005 Z0.0731
110   X0 Y0.005 Z0.1001
111   X0 Y0.0045 Z0.1006
112   X0 Y0.0037 Z0.1006
113   X0 Y0.0025 Z0.0996
114   X0 Y0.002 Z0.0993
115   X0 Y0.002 Z0.0981
116   X0 Y0.0025 Z0.0978
117   X0 Y0.002 Z0.0975
118   X0 Y0.002 Z0.0963
119   X0 Y0.0025 Z0.096
120   X0 Y0.002 Z0.0957
121   X0 Y0.002 Z0.0945
122   X0 Y0.0025 Z0.0942
123   X0 Y0.002 Z0.0939
124   X0 Y0.002 Z0.0927
```

```
125  X0 Y0.0025 Z0.0924
126  (...end cut/exit strategy...)
127  G0 Z0.2
128  (...new cut...)
129  G0 X0.1451 Y0.015
130  P0
131  Z0.584335705214648
132  F100
133  G1 Z0.0843(...enter strategy...)
134  P0
135  F1000
136  X0.1452 Y0.0144 Z0.084
137  X0.1452 Y0.0126 Z0.0773
138  X0.1451 Y0.0123 Z0.0718
139  X0.1843 Y0.0121 Z0.0716
140  X0.1843 Y0.0065 Z0.0716
141  X0.1844 Y0.0056 Z0.0719
142  X0.1849 Y0.005 Z0.0731
143  X0.1947 Y0.005 Z0.1001
144  X0.1949 Y0.0045 Z0.1006
145  X0.1949 Y0.0037 Z0.1006
146  X0.1945 Y0.0025 Z0.0996
147  X0.1944 Y0.002 Z0.0993
148  X0.194 Y0.002 Z0.0981
149  X0.1938 Y0.0025 Z0.0978
150  X0.1937 Y0.002 Z0.0975
151  X0.1933 Y0.002 Z0.0963
152  X0.1932 Y0.0025 Z0.096
153  X0.1931 Y0.002 Z0.0957
154  X0.1926 Y0.002 Z0.0945
155  X0.1925 Y0.0025 Z0.0942
156  X0.1924 Y0.002 Z0.0939
157  X0.192 Y0.002 Z0.0927
158  X0.1919 Y0.0025 Z0.0924
159  X0.1918 Y0.002 Z0.0921
160  X0.1913 Y0.002 Z0.0909
161  X0.1912 Y0.0025 Z0.0906
162  X0.1911 Y0.002 Z0.0903
163  X0.1907 Y0.002 Z0.0891
164  X0.1906 Y0.0025 Z0.0888
165  X0.1905 Y0.002 Z0.0885
166  X0.19 Y0.002 Z0.0873
```

```
167   X0.1899 Y0.0025 Z0.087
168   X0.1898 Y0.002 Z0.0867
169   X0.1891 Y0.002 Z0.0848
170   X0.1884 Y0.0009 Z0.0828
171   X0.1882 Y0 Z0.0823
172   X0.1884 Y−0.0009 Z0.0828
173   X0.1891 Y−0.002 Z0.0848
174   X0.1895 Y−0.002 Z0.0858
175   X0.1896 Y−0.0025 Z0.0861
176   X0.1897 Y−0.002 Z0.0864
177   X0.1901 Y−0.002 Z0.0876
178   X0.1902 Y−0.0025 Z0.0879
179   X0.1904 Y−0.002 Z0.0882
180   X0.1908 Y−0.002 Z0.0894
181   X0.1909 Y−0.0025 Z0.0897
182   X0.191 Y−0.002 Z0.09
183   X0.1915 Y−0.002 Z0.0912
184   X0.1916 Y−0.0025 Z0.0915
185   X0.1917 Y−0.002 Z0.0918
186   X0.1921 Y−0.002 Z0.093
187   X0.1922 Y−0.0025 Z0.0933
188   X0.1923 Y−0.002 Z0.0936
189   X0.1928 Y−0.002 Z0.0948
190   X0.1929 Y−0.0025 Z0.0951
191   X0.193 Y−0.002 Z0.0954
192   X0.1934 Y−0.002 Z0.0966
193   X0.1935 Y−0.0025 Z0.0969
194   X0.1936 Y−0.002 Z0.0972
195   X0.1941 Y−0.002 Z0.0984
196   X0.1942 Y−0.0025 Z0.0987
197   X0.1945 Y−0.0025 Z0.0996
198   X0.1949 Y−0.0035 Z0.1006
199   X0.1949 Y−0.0045 Z0.1006
200   X0.1947 Y−0.005 Z0.1001
201   X0.1849 Y−0.005 Z0.0731
202   X0.1845 Y−0.0053 Z0.0722
203   X0.1843 Y−0.0065 Z0.0716
204   X0.1843 Y−0.0121 Z0.0716
205   X0.1451 Y−0.0126 Z0.0721
206   X0.1452 Y−0.0126 Z0.0773
207   X0.1452 Y−0.0144 Z0.084
208   X0.1451 Y−0.015 Z0.0843
```

```
209  X0.145 Y-0.0163 Z0.084
210  X0.145 Y-0.0169 Z0.0829
211  X0.145 Y-0.0167 Z0.0823
212  X0.1451 Y-0.0156 Z0.0826
213  X0.1451 Y-0.0149 Z0.08
214  X0.1445 Y-0.0214 Z0.0783
215  X0.1443 Y-0.0247 Z0.0802
216  X0.1444 Y-0.0235 Z0.0805
217  X0.1444 Y-0.0237 Z0.0811
218  X0.1443 Y-0.0247 Z0.0817
219  X0.1442 Y-0.0265 Z0.0812
220  X0.1442 Y-0.0255 Z0.0776
221  X0.1445 Y-0.0211 Z0.075
222  X0.145 Y-0.0146 Z0.0768
223  X0.145 Y-0.0141 Z0.0767
224  X0.1449 Y-0.0141 Z0.0718
225  X0.1843 Y-0.0141 Z0.0716
226  X0.1843 Y-0.025 Z0.0716
227  X0.1588 Y-0.025 Z0.0016
228  X0.1584 Y-0.0246 Z0.0006
229  X0.1583 Y-0.0237 Z0.0001
230  X0.1583 Y0.0235 Z0.0001
231  X0.1584 Y0.0246 Z0.0005
232  X0.1588 Y0.025 Z0.0016
233  X0.1843 Y0.025 Z0.0716
234  X0.1843 Y0.0141 Z0.0716
235  X0.1449 Y0.0141 Z0.0718
236  X0.145 Y0.0141 Z0.0767
237  X0.145 Y0.0146 Z0.0768
238  X0.1445 Y0.0211 Z0.075
239  X0.1442 Y0.0255 Z0.0776
240  X0.1442 Y0.0265 Z0.0812
241  X0.1443 Y0.0247 Z0.0817
242  X0.1444 Y0.0237 Z0.0811
243  X0.1444 Y0.0235 Z0.0805
244  X0.1443 Y0.0247 Z0.0802
245  X0.1445 Y0.0214 Z0.0783
246  X0.1451 Y0.0149 Z0.08
247  X0.1451 Y0.0156 Z0.0826
248  X0.145 Y0.0167 Z0.0823
249  X0.145 Y0.0169 Z0.0829
250  X0.145 Y0.0163 Z0.084
```

```
251   X0.1451 Y0.015 Z0.0843
252   (...end cut/exit strategy...)
253   G0 Z0.2
254
255   G0 Z0.2
256   M30 (program stop)
```

Q1: Script: FehlermeldungTrigger

```
"""Diese Komponente überprüft die berechneten Winkel auf ihren Gültigkeitsbereich
    Inputs:
        a: Winkel Riegel
        b: Winkel Riegel
        c: Winkel Pfosten
        d: Winkel Pfosten
    Output:
        error: Boolean ob Fehlermeldung ausgegeben werden muss"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if float(a)<= 45 and float(b) <= 45:
    if float(c) <= 40 and float(d) <= 40:
        error = False
    else:
        error = True
else:
    error = True
```

Q2: Script: ErrorMsgBox

```
"""Diese Methode gibt eine Fehlermeldung zurück.
Inputs:
        a: error(bool) vom linken Feld
        b: error(bool) vom rechten Feld
    Output:
    msg: Fehlermeldung"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

msg = "Die Winkel sind möglich!"
if a or b:
msg = "Bitte überprüfen Sie die Winkel!"
```

Q3: Script: Select InnerProfile

```
"""Bestimmt das Profil für die Pfosten!
    Inputs:
        NullL: Profil 0 linker Winkel
        NullR: Profil 0 rechter Winkel
        EinsL: Profil 1 linker Winkel
        EinsR: Profil 1 rechter Winkel
    Output:
        Pfostenprofil: Spaltennummer des Pfostenprofils"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.2"

import rhinoscriptsyntax as rs

#Sollte ein Winkel außerhalb des Definitionsbereichs liegen, dann gib Error
if NullL>40 or NullR>40 or EinsL>40 or EinsR>40:
    raise NameError('Zu großer Winkel!')

#Ermittle den kleinsten und größten Winkel
inputs = [NullL,NullR,EinsL,EinsR]
min = 200
max = -10
for x in inputs:
    if min > x:
        min = x
    if max < x:
        max = x

if min == 200:
    raise NameError('Kein Minimum gefunden!!')
if max == -10:
    raise NameError('Kein Maximum gefunden!!')
#die min() und max() Funktionen rufen durch die Floats der Winkel fehler hervor.
#daher sind die Funktionen oben selbst geschrieben
#min = min(inputs)
#max = max([NullL,NullR,EinsL,EinsR])

#Grenzen (kleinster und größter möglicher Winkel)
```

```
39  W_max = min + (5 - min%5)
40  if W_max > 25:
41      W_max = 25
42  W_min = max - (10 + max%5)
43  if W_min > 25:
44      W_min = 25
45
46  #Sollte der kleinst mögliche Winkel größer als der größte sein,
        dann gib Error
47  #Dies kann vorkommen, falls die Winkel zu weit auseinander liegen!
48  if W_min > W_max:
49      raise NameError('Winkel liegen zu weit auseinander!')
50
51  #Ermittle Pfostenprofil
52  if W_max == W_min:
53      p = W_max
54  else:
55      mittel = (max + min) / 2
56      closest = round(mittel/5) * 5
57      if closest > W_max:
58          p = W_max
59      else:
60          p = closest
61
62  #Wandle Pfostenprofil von GRAD in Spaltennummer um.
63  Pfostenprofil = p / 5 +1
```

Q4: Script: Pick Riegelprofile

```
"""Diese Komponente wählt ein geeignetes Riegelprofil. Dabei wird
    in std. Profile und variable Riegelprofile unterschieden. Als
    Annahme wird ein Winkel von 2 Grad gesetzt, bei dem std. Profile
     mit den std. Dichtungen noch anwendbar sind.
    Inputs:
        t: Tiefe der Riegel
        w: Summe linker und rechter Winkel
    Output:
        s: Spalten-Nr. des Riegels
        z: Zeilen-Nr. des Riegels"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if w < 2 and w > -2:
    s = [1]
    z = [t]
else:
    s = [2,2]
    z = [1,2]
```

Q5: Script: SelectInnerSealing

```
"""Berechnet die Winkel des Profils und der Dichtungen für den Pfosten!
    Inputs:
        Pfostenprofil: Nummer des Pfostenprofils
        PfostenL: linker Winkel der Pfosten 0 und 1 (Tree)
        PfostenR: rechter Winkel der Pfosten 0 und 1 (Tree)
    Output:
        PfostenInnereDichtungL: Spalten/Zeilen-Nr der linken inneren Dichtung
        PfostenInnereDichtungR: Spalten/Zeilen-Nr der rechten inneren Dichtung"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.1"

import rhinoscriptsyntax as rs

#Umwandlung der Profilnummer in zugehörigen Winkel
p = (Pfostenprofil - 1)*5

#Berechne Abweichung der Winkel von Profil
temp_DL = PfostenL - p
temp_DR = PfostenR - p

#Bestimmte zu verwendende Dichtung
if temp_DL > -1.25 and temp_DL < 1.25:
    PfostenInnereDichtungL = 1
else:
    if temp_DL <= -1.25:
        PfostenInnereDichtungL = 2
    else:
        if temp_DL < 10:
            PfostenInnereDichtungL = 3
        else:
            if temp_DL <= 15:
                PfostenInnereDichtungL = 4
            else:
                raise NameError('Fehler Zeile 44')
if temp_DR >-1.25 and temp_DR < 1.25:
    PfostenInnereDichtungR = 1
```

```
38  else:
39      if temp_DR < 0:
40          PfostenInnereDichtungR = 2
41      else:
42          if temp_DR < 10:
43              PfostenInnereDichtungR = 3
44          else:
45              if temp_DR <= 15:
46                  PfostenInnereDichtungR = 4
47              else:
48                  raise NameError('Fehler Zeile 54')
```

Q6: Script: PfadmaskenInnereDichtungRiegel

```
"""Diese Komponente gibt für eingehende Winkel Filtermasken zurück.
    Inputs:
        winkel: Eingehender Winkel
    Output:
        Path: Filtermaske"""

__author__ = "Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

Path = []
#offizielle Unterstützung des variablen Riegels: 38 – 90 Grad
#Annahme: Normales Riegelprofil kann bis +/– 2 Grad mit std. Dichtung aufnehmen
for x in range(0,2):
    if winkel[x] <= 2:
        #Bei normalen Riegelprofilen sind keine 2 Profilhälften erforderlich!
        Path.append("{"+str(x+2)+";1;0;1}")
    #elif x <= 38 Grad: #(Verwenden wenn weiteres Profil eingefügt wird, dass zwischen 0 und 38 Grad funktioniert)
        #Anweisung für neues Profil
    elif winkel[x] > 2:
        Path.append("{"+str(x+2)+";1;1;0}")
```

Q7: Script: InnereDichtungen und Isolator

```
"""Diese Komponente bestimmt die Zeilen-Nummern der Dichtungen und
    des Isolators
    Inputs:
        g: Dicke der Fassadenelemente
    Output:
        DichtungNr: Zeilen-Nummer der Dichtungen
        IsolatorNr: Zeilen-Nummer des Isolators"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if g == 0 or g == 2 or g == 5:
    DichtungNr = 1
elif g == 1 or g == 3 or g == 6:
    DichtungNr = 2
elif g == 4 or g == 7:
    DichtungNr = 3

if g == 0 or g == 1:
    IsolatorNr = 1
else:
    IsolatorNr = int((g+1)/3)+1
```

Q8: Script: Datenstruktur Isolator

```
"""Diese Komponete passt die Datenstruktur der Isolatoren an.
    Inputs:
        winkel: Winkel der Riegel
        INr: Gewählter Isolator
    Output:
        INr_out: Angepasste Datenstruktur der Isolatoren"""

__author__ = "Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

#offizielle Unterstützung des variablen Riegels: 38 – 90 Grad
#Annahme: Normales Riegelprofil kann bis +/– 2 Grad mit std. Dichtung aufnehmen

if winkel <= 2:
    #Bei normalen Riegelprofilen sind keine 2 Profilhälften erforderlich!
    INr_out = INr
#elif winkel <= 38 Grad: #(Verwenden wenn weiteres Profil eingefügt wird, dass zwischen 0 und 38 Grad funktioniert)
    #Anweisung für neues Profil
elif winkel > 2:
    INr_out = [INr,INr]
```

Q9: Script: Select OuterSealing

```
"""Berechnet die Winkel des Profils und der Dichtungen für den Pfosten!
    Inputs:
        Pressleitel: Nummer der Pressleiste
        PfostenL: linker Winkel der Pfosten 0 und 1 (Tree)
        PfostenR: rechter Winkel der Pfosten 0 und 1 (Tree)
    Output:
        PÄDichtungL: Spalten/Zeilen-Nr der linken äußeren Dichtung
        PÄDichtungR: Spalten/Zeilen-Nr der rechten äußeren Dichtung
            """

__author__ = "Adrian Schubert"
__version__ = "V1.1"

import rhinoscriptsyntax as rs

PressleistePfosten == Pressleiste

if PressleistePfosten == 1:
    p = 0
elif PressleistePfosten == 2:
    p = 3.75
elif PressleistePfosten == 3:
    p = 7.5
elif PressleistePfosten == 4:
    p = 15
elif PressleistePfosten == 5:
    p = 20
elif PressleistePfosten == 6:
    p = 30
elif PressleistePfosten == 7:
    p = 35


#Bestimmung der S_OuterSealing
PfostenL = PfostenL - p
PfostenR = PfostenR - p
if PfostenL > -1.25 and PfostenL < 1.25:
    PfostenÄußereDichtungL = 1
else:
```

```
39      if PfostenL < 0:
40          PfostenÄußereDichtungL = 2
41      else:
42          if PfostenL < 10:
43              PfostenÄußereDichtungL = 3
44          else:
45              if PfostenL <= 15:
46                  PfostenÄußereDichtungL = 4
47              else:
48                  raise NameError('Fehler Zeile 46')
49  if PfostenR > -1.25 and PfostenR < 1.25:
50      PfostenÄußereDichtungR = 1
51  else:
52      if PfostenR < 0:
53          PfostenÄußereDichtungR = 2
54      else:
55          if PfostenR < 10:
56              PfostenÄußereDichtungR = 3
57          else:
58              if PfostenR <= 15:
59                  PfostenÄußereDichtungR = 4
60              else:
61                  raise NameError('Fehler Zeile 59')
```

Q10: Script: Riegel Datenstruktur äußere Dichtung

```
1  """Diese Komponente passt die Datenstruktur der äußeren
       Riegeldichtung an.
2      Inputs:
3          winkel: Winkel der Riegel
4          SNr: Spalten Nummer der Dichtung (immer = 1)
5      Output:
6          SNr_out: Ausgabe der korrigierten Spaltennummer"""
7
8  __author__ = "cand. ing. Adrian Schubert"
9  __version__ = "V1.0"
10
11 import rhinoscriptsyntax as rs
12
13 #offizielle Unterstützung des variablen Riegels: 38 – 90 Grad
14 #Annahme: Normales Riegelprofil kann bis +/– 2 Grad mit std.
       Dichtung aufnehmen
15
16 if winkel <= 2:
17     #Bei normalen Riegelprofilen sind keine 2 Profilhälften
           erforderlich!
18     SNr_out = SNr
19 #elif winkel <= 38 Grad: #(Verwenden wenn weiteres Profil eingefügt
       wird, dass zwischen 0 und 38 Grad funktioniert)
20     #Anweisung für neues Profil
21 elif winkel > 2:
22     SNr_out = [SNr,SNr]
```

Q11: Script: Select OuterProfile

```
"""Bestimmt das Pressprofil für die Pfosten!
    Inputs:
        NullL: Profil 0 linker Winkel
        NullR: Profil 0 rechter Winkel
        EinsL: Profil 1 linker Winkel
        EinsR: Profil 1 rechter Winkel
    Output:
        PressleistePfosten: Spaltennummer des Pfostenprofils"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.1"

import rhinoscriptsyntax as rs

#Sollte ein Winkel außerhalb des Definitionsbereichs liegen, dann
    gib Error
if NullL>50 or NullR>50 or EinsL>50 or EinsR>50:
    raise NameError('Zu großer Winkel!')

#Eliminierung von Rundungsfehlern (Damit 5%5 = 0 und nicht 5%5 = 1)
    (Fehler bedingt durch int(5) =/= float(5))
NullL = NullL + 0.00000000000001
NullR = NullR + 0.00000000000001
EinsL = EinsL + 0.00000000000001
EinsR = EinsR + 0.00000000000001

inputs = [NullL, NullR, EinsL,EinsR]
min = 200
max = -10
for x in inputs:
    if min > x:
        min = x
    if max < x:
        max = x

if min == 200:
    raise NameError('Kein Minimum gefunden!!')
if max == -10:
    raise NameError('Kein Maximum gefunden!!')

```

```
39
40  #Grenzen (Maximal mögliche Winkel)
41  #Bem: Profil 2 (3.75 Grad ) wird ignoriert, da keine Neigung
        vorhanden, lediglich etwas breiter
42  if min < 2.5:
43      W_max = 0
44  elif min >= 2.5 and min < 7.5:
45      W_max = 7.5
46  elif min >= 7.5 and min < 10:
47      W_max = 7.5 # Für diesen Bereich stehen keine Dichtungen zur
            Verfügung, um ein größeres Profil zu verwenden
48  elif min >= 10 and min < 15:
49      W_max = 15
50  elif min >= 15 and min < 20:
51      W_max = 20
52  elif min >= 20 and min < 25:
53      W_max = 20 # Für diesen Bereich stehen keine Dichtungen zur
            Verfügung, um ein größeres Profil zu verwenden
54  elif min >= 25 and min < 30:
55      W_max = 30
56  elif min >= 30 and min < 45:
57      W_max = 35
58  elif min >= 45:
59      raise NameError('kleiner Winkel zu groß!')
60  else:
61      raise NameError('Unerwarteter Fehler in Zeile 49')
62
63  #Bem: Profil 2 (3.75 Grad ) wird ignoriert, da keine Neigung
        vorhanden, lediglich etwas breiter
64  if max >= 50:
65      raise NameError('Großer Winkel zu groß!')
66  elif max >= 45:
67      W_min = 35
68  elif max >= 35:
69      W_min = 30
70  elif max >= 30:
71      W_min = 20
72  elif max >= 22.5:
73      W_min = 15
74  elif max >= 15:
75      W_min = 7.5
76  else:
```

```
77      W_min = 0
78
79  if W_max < W_min:
80      raise NameError('Winkel liegen zu weit auseinander!')
81  elif W_max == W_min:
82      p = W_max
83  else:
84      mittel = (max + min) / 2
85      if mittel < 1.875:
86          closest = 0
87      elif mittel < 5.625:
88          closest = 3.75
89      elif mittel < 11.25:
90          closest = 7.5
91      elif mittel < 17.5:
92          closest = 15
93      elif mittel < 25:
94          closest = 20
95      elif mittel < 32.5:
96          closest = 30
97      else:
98          closest = 35
99      if closest > W_max:
100         p = W_max
101     else:
102         p = closest
103
104 #Bestimmung von PressleistePfosten
105 if p == 0:
106     p_out = 1
107 elif p == 3.75:
108     p_out = 2
109 elif p == 7.5:
110     p_out = 3
111 elif p == 15:
112     p_out = 4
113 elif p == 20:
114     p_out = 5
115 elif p == 30:
116     p_out = 6
117 elif p == 35:
118     p_out = 7
```

```
119
120   PressleistePfosten = p_out
```

Q12: Script: Blendleiste

```
1  """Dieses Skript konvertiert die Spalten-Nummern der Pressleiste in
       die Spalten-Nummer der Blendleiste
2      Inputs:
3          x: Spalten-Nummer der Pressleiste
4      Output:
5          a: Spalten-Nummer der Blendleiste"""
6
7  __author__ = "cand. ing. Adrian Schubert"
8  __version__ = "V1.0"
9
10 import rhinoscriptsyntax as rs
11
12 if x == 1:
13     a = 1
14 elif x >= 2:
15     a = 2
```

Q13: Script: Filename

```
"""Gibt den Dateipfad für entsprechenden Input zurück.
    Inputs:
        MainDirectory: Gibt das Hauptverzeichnis des Projektes an
        SubDirectory: Gibt einen Unterordner im Hauptverzeichnis an
            (Ohne abschließendes \)
        Filename: Verwenden, wenn ein genauer Dateiname vorhanden
        ArtNr: Verwenden, wenn nur die Art.-Nr. bekannt
    Output:
        FilePath: Gibt den Dateipfad oder Ordnerpfad (wenn kein
            Filename oder ArtNr) zurück"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

FilePath = str(MainDirectory)

if SubDirectory != "NULL":
    FilePath = str(FilePath)  + str(SubDirectory) + "\\"
if Filename != "NULL" and ArtNr !="NULL":
        raise NameError('Entweder Filename ODER ArtNr')
else:
    if Filename != "NULL":
        FilePath = FilePath + str(Filename)
    if ArtNr != "NULL":
        FilePath = FilePath + str(ArtNr) + ".3dm"
```

Q14: Script: CutterPressleiste

```
"""Diese Komponente wandelt die S\_Pressleiste in die Eckpunkte des
    Cutters der Pressleiste um.
    Inputs:
        S\_PressleistePfosten: Spalten-Nummer der Pressleiste der
            Pfosten
    Output:
        Point: Eckpunkte des Cutters"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if S_PressleistePfosten == 1:
    Points = ["{0,-0.01,0}","{0.025,-0.01,0}","{0.025,-0.1,0}","
        {0,-0.1,0}"]
elif S_PressleistePfosten == 2:
    Points = ["{0,-0.01,0}","{0.03602,-0.01,0}","{0.03602,-0.1,0}",
        "{0,-0.1,0}"]
elif S_PressleistePfosten == 3:
    Points = ["{-0.1,-0.0025,0}","{0.0365,-0.0025,0}","
        {0.03985,-0.02835,0}","{0.03985,-0.1,0}","{-0.1,-0.1,0}"]
elif S_PressleistePfosten == 4:
    Points = ["{0,0.0075,0}","{0.0423375,-0.0075,0}","
        {0.04737,-0.02625,0}","{0.04737,-0.1,0}","{0,-0.1,0}"]
elif S_PressleistePfosten == 5:
    Points = ["{0,-0.0075,0}","{0.04815,-0.0075,0}","
        {0.05345,-0.022,0}","{0.05345,-0.1,0}","{0,-0.1,0}"]
elif S_PressleistePfosten == 6:
    Points = ["{0,-0.0025,0}","{0.051,-0.0025,0}","
        {0.05995,-0.018,0}","{0.05995,-0.1,0}","{0,-0.1,0}"]
elif S_PressleistePfosten == 7:
    Points = ["{0,0.0075,0}","{0.0567,0.0075,0}","
        {0.0687,-0.00965,0}","{0.0687,-0.1,0}","{0,-0.1,0}"]
```

Q15: Script: CutterRiegel

```
"""Diese Komponente gibt die Cuttergeometrie des Riegels zurück
    Inputs:
        y: Winkel
        x: Scheibendicke
    Output:
        a: Punkte der Cuttergeometrie"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if y < 2: #var Riegel
    A = -0.0251
else: #std Riegel
    A = -0.0283

if x == 0 or x == 2 or x == 5:
    B = -0.01225-0.01
elif x == 1 or x == 3 or x == 6:
    B = -0.0111-0.01
elif x == 4 or x == 7:
    B = -0.009-0.01

Points = ["{0.1," + str(B) + ",0}","{" + str(A) + "," + str(B) + "
    ,0}","{" + str(A) + ",0.1,0}","{0.1,0.1,0}"]
```

Q16: Script: FilterCutter

```
"""Komponete gibt Faktor zur Korrektur des etwas breiteren
    variablen Riegels aus
    Inputs:
        y: Winkel
    Output:
        a: Korrekturfaktor"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if y < 2:
    a = "{(2,3);7;(0,1);(1)}"
else:
    a = ""
```

Q17: Script: ÄußereVerschiebung

```
1  """Diese Komponente gibt die Verschiebung der äußeren Komponenten
       zurück.
2      Inputs:
3          g: Dicke der Glasscheibe
4      Output:
5          a: Verschiebung der Äußeren Komponenten"""
6
7  __author__ = "cand. ing. Adrian Schubert"
8  __version__ = "V1.0"
9
10 import rhinoscriptsyntax as rs
11
12 #Midnestabstand
13 c = -0.0261
14
15 if g == 0 or g == 1:
16     a = c
17 elif g > 1 and g <= 4:
18     a = c-0.004
19 elif g > 4:
20     b = c - 0.004 - 0.006*int((g-2)/3)
21     a = b
```

Q18: Script: InnereDichtung

```
"""Diese Komponente dient zur Bestimmung des Rotationswinkels der
    inneren Dichtung
    Inputs:
        S_TragprofilPfosten: Spalten-Nummer des Pfosten Tragprofils
    Output:
        WinkelDichtung: Rotationswinkel der inneren Dichtung"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs
s = S_TragprofilPfosten

w = (s -1)*5

WinkelDichtung = w
```

Q19: Script: ÄußereDichtung

```
"""Diese Komponente dient zur Bestimmung des Rotationswinkels und
    der Translation der äußeren Dichtung
    Inputs:
        S_PressleistePfosten: Spalten-Nummer der Pfosten
            Pressleiste
    Output:
        WinkelDichtung: Rotationswinkel der inneren Dichtung"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs
s = (S_PressleistePfosten-1)

if s == 0:
    w = 0
    t = "0,0,0"
elif s == 1:
    w = 0
    t = "-0.0107,0,0"
elif s == 2:
    w = 7.5
    t = "-0.01275,0.00008,0"
elif s == 3:
    w = 15
    t = "-0.01625,0.0001,0"
elif s == 4:
    w = 20
    t = "-0.02425,0.0025,0"
elif s == 5:
    w = 30
    t = "-0.0293,0.0029,0"
elif s == 6:
    w = 35
    t = "-0.03175,0.0079,0"


WinkelDichtung = w
TranslationDichtung = t
```

Q20: Script: MaskeInputFix

```
"""Komponete gibt Maske zur Korrektur der variablen Riegel aus
    Inputs:
        y: Anzahl der Pfade in Datenstruktur
    Output:
        a: Korrekturmaske"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.1"

import rhinoscriptsyntax as rs

if y == 2:
    a = "{(2,3);1}"
else:
    a = ""
```

Q21: Script: MaskeVerschiebungsVektoren

```
"""Komponete gibt Maske zur Korrektur der variablen Riegel aus
    Inputs:
        count: Anzahl der Pfade in Datenstruktur
    Output:
        mask: Filtermaske"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if count == 4:
    mask = "{(2,3);0;0;0;(2,3)}"
else:
    mask = ""
```

Q22: Script: DickeVarMitten

```
"""Dieses Skript berechnet die Dicke des Paneels der
    Mittelverblednung der variablen Riegelprofile
    Inputs:
        S_Isolator: Spalten-Nummer des Isolators
    Output:
        PaneelDicke: Dicke des Paneels"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.1"

import rhinoscriptsyntax as rs

if S_Isolator == 0:
    PaneelDicke = -0.0375
if S_Isolator == 1:
    PaneelDicke = -0.0375 - 0.004
if S_Isolator >= 2:
    PaneelDicke = -0.0375 - 0.004 - 0.006*(S_Isolator-1)
```

Q23: Script: FilterMittenVarRiegel

```
"""Diese Komponente überprüft die Notwendigkeit von Mittenblenden
    der variablen Riegelprofile
    Inputs:
        winke: Öffnungswinkel der var. Profile
    Output:
        mask: Filtermaske"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.1"

import rhinoscriptsyntax as rs

if winkel >= 38 and winkel <= 90:
    mask = ""
else:
    mask = "{(2);8;0;(0,1)}" #Für das zweite Skript hier {(3)
        ;8;0;(0,1)}
```

Q24: Script: Verschiebung Geometriemitte

```
"""Diese Komponente berechnet den Abstand zum Scheibenmittelpunkt
    Inputs:
        x: Gewählte Elementdicke
    Output:
        a: Abstand"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if x < 2:
    a = (12 - x)/1000
if x >= 2:
    a = ((28 + int((x-2)/3)*6) - (28 + int((x-2)/3)*6 + 2*((x-2)%3)
        )/2)/1000
```

Q25: Script: KorrekturVarRiegel I

```
"""Komponete gibt Faktor zur Korrektur des etwas breiteren
    variablen Riegels
    Inputs:
        y: Winkel
    Output:
        a: Korrekturfaktor"""

__author__ = "Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if y < 2:
    a = "{(2,3);0;(0);(1)}"
else:
    a = ""
```

Q26: Script: KorrekturVarRiegel II

```
"""Komponete gibt Faktor zur Korrektur des etwas breiteren
    variablen Riegels
    Inputs:
        y: Winkel
    Output:
        a: Korrekturfaktor"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

if y < 2:
    a = "{(2,3);5;(0);(1)}"
else:
    a = ""
```

Q27: Script: FilterZuMaske

```
"""Diese Komponente wandelt die Filtereinstellungen in eine Maske
    um.
    Inputs:
        a – d: Filtereinstellungen
    Output:
        Maske: Maske"""

__author__ = "cand. ing. Adrian Schubert"
__version__ = "V1.0"

import rhinoscriptsyntax as rs

Maske = "{"+str(a)+";"+str(b)+";"+str(c)+";"+str(d)+"}"
```