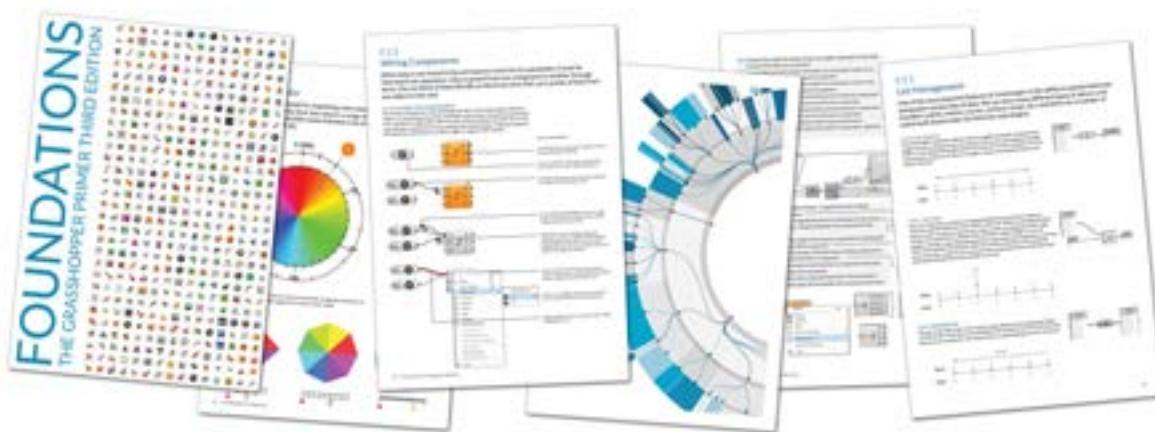


# The Grasshopper Primer (DE)

## Third Edition V3.3



Grasshopper is a graphical algorithm editor tightly integrated with Rhino's 3-D modeling tools, allowing designers to build form generators from the simple to the awe-inspiring.

## WELCOME

You have just opened the third edition of the Grasshopper Primer. This primer was originally written by Andrew O. Payne of Lift Architects for Rhino4 and Grasshopper version 0.6.0007 which, at the time of its release, was a giant upgrade to the already robust Grasshopper platform. We now find ourselves at another critical shift in Grasshopper development, so a much needed update to the existing primer was in order. We are thrilled to add this updated, *and now web-based*, primer to the many amazing contributions put forth by Grasshopper community members.

With an already excellent foundation from which to build, our team at [Mode Lab](#) went to work designing and developing the look and feel of the third edition. This revision provides a comprehensive guide to the most current Grasshopper build, version 0.90076, highlighting what we feel are some of the most exciting feature updates. The revised text, graphics, and working examples are intended to teach visual programming to the absolute beginner, as well as provide a quick introduction to Generative Design workflows for the seasoned veteran. It is our goal that this primer will serve as a field guide to new and existing users looking to navigate the ins and outs of using Grasshopper in their creative practice.

This primer introduces you to the fundamental concepts and essential skill-building workflows to effectively use Grasshopper. Foundations is the first volume in a forthcoming collection of Grasshopper primers. Here's what you can expect to learn from the primer:

- **Introduction** - What is Grasshopper and how is it being used?
- **Hello Grasshopper** - Make your first definition
- **Anatomy of a Grasshopper Definition** - What makes up a definition?
- **Building Blocks of Algorithms** - Start simple and build complexity
- **Designing with Lists** - What's a list and how do I manage them?
- **Designing with Data Tree** - What's a data structure and what do they mean for my process?

- **Appendix** - References and Working files for continued exploration

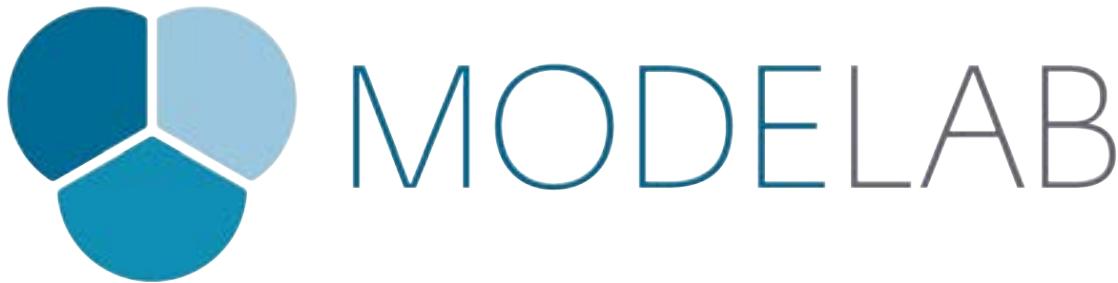
We hope that at the very least this primer will inspire you to begin exploring the many opportunities of programming with Grasshopper. We wish you the best of luck as you embark on this journey.

---

## THE GRASSHOPPER PRIMER PROJECT

The Grasshopper Primer is an open source project, initiated by Bob McNeil, Scott Davidson, and the Grasshopper Development team at [Robert McNeil & Associates](#).

**Mode Lab** authored the Third Edition of the primer. <http://modelab.is>



If you would like to contribute to this project, check out the github project wiki to get started (<https://github.com/modelab/grasshopper-primer/wiki>).

## ACKNOWLEDGEMENTS

A special thanks to David Rutten for the endless inspiration and invaluable work pioneering Grasshopper. We would also like to thank Andrew O. Payne for providing the assets from which this work initiated. Lastly, many thanks to Bob McNeil and everyone at Robert McNeil & Associates for their generous support over the years.

---

## REQUIRED SOFTWARE

### Rhino5

Rhino 5.0 is the market leader in industrial design modeling software. Highly complicated shapes can be directly modeled or acquired through 3D digitizers. With its powerful NURBS based engine Rhino 5.0 can create, edit, analyze, and translate curves, surfaces, and solids. There are no limits on complexity, degree, or size.

<http://www.rhino3d.com/download/rhino/5/latest>

### Grasshopper

For designers who are exploring new shapes using generative algorithms, Grasshopper is a graphical algorithm editor tightly integrated with Rhino's 3D modeling tools. Unlike RhinoScript or Python, Grasshopper requires no knowledge of the abstract syntax of scripting, but still allows designers to build form generators from the simple to the awe inspiring.

<http://www.grasshopper3d.com/page/download-1>

## FORUMS

The Grasshopper forum is very active and offers a wonderful resource for posting questions/answers and finding help on just about anything. The forum has categories for general discussion, errors & bugs, samples & examples, and FAQ.

<http://www.grasshopper3d.com/forum>

The Common Questions section of the Grasshopper site contains answers to many questions you may have, as well as helpful links:

<http://www.grasshopper3d.com/notes/index/allNotes>

For more general questions pertaining to Rhino3D be sure to check out the McNeel Forum powered by Discourse.

<http://discourse.mcneel.com/>

## LICENSING INFORMATION

The Grasshopper Primer is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported license. The full text of this license is available here: <http://creativecommons.org/licenses/by-nc-sa/3.0/us/legalcode>

# Inhaltsverzeichnis

---

- 0. [About](#)
  - 0.1. [Grasshopper - an Overview](#)
  - 0.2. [Grasshopper in Action](#)
- 1. [Foundations](#)
  - 1.1. [Hello Grasshopper!](#)
    - 1.1.1. [Installing and Launching Grasshopper](#)
    - 1.1.2. [The Grasshopper UI](#)
    - 1.1.3. [Talking to Rhino](#)
  - 1.2. [Anatomy of a Grasshopper Definition](#)
    - 1.2.1. [Grasshopper Object Types](#)
    - 1.2.2. [Grasshopper Component Parts](#)
    - 1.2.3. [Data Types](#)
    - 1.2.4. [Wiring Components](#)
    - 1.2.5. [The Grasshopper Definition](#)
  - 1.3. [Building Blocks of Algorithms](#)
    - 1.3.1. [Points Planes & Vectors](#)
    - 1.3.2. [Working With Attractors](#)
    - 1.3.3. [Mathematics, Expressions & Conditionals](#)
    - 1.3.4. [Domains & Color](#)
    - 1.3.5. [Booleans and Logical Operators](#)
  - 1.4. [Designing with Lists](#)
    - 1.4.1. [Curve Geometry](#)
    - 1.4.2. [What is a List?](#)
    - 1.4.3. [Data Stream Matching](#)
    - 1.4.4. [Creating Lists](#)
    - 1.4.5. [List Visualization](#)
    - 1.4.6. [List Management](#)
    - 1.4.7. [Working with Lists](#)
  - 1.5. [Designing with Data Trees](#)
    - 1.5.1. [Surface Geometry](#)
    - 1.5.2. [What is a Data Tree?](#)
    - 1.5.3. [Creating Data Trees](#)
    - 1.5.4. [Working with Data Trees](#)
  - 1.6. [Getting Started with Meshes](#)
    - 1.6.1. [What is a Mesh?](#)
    - 1.6.2. [Understanding Topology](#)
    - 1.6.3. [Creating Meshes](#)
    - 1.6.4. [Mesh Operations](#)
    - 1.6.5. [Mesh Interactions](#)
    - 1.6.6. [Working with Mesh Geometry](#)
- 2. [Extensions](#)
  - 2.1. [Element\\*](#)
    - 2.1.1. [Introduction](#)
    - 2.1.2. [Half Edge Data](#)
    - 2.1.3. [Components](#)
    - 2.1.4. [Exercise](#)
    - 2.1.5. [Architectural Case Study](#)

3. [Appendix](#)

3.1. [Index](#)

3.2. [Example Files](#)

3.3. [Resources](#)

3.4. [About This Primer](#)

# Grasshopper - Eine Uebersicht

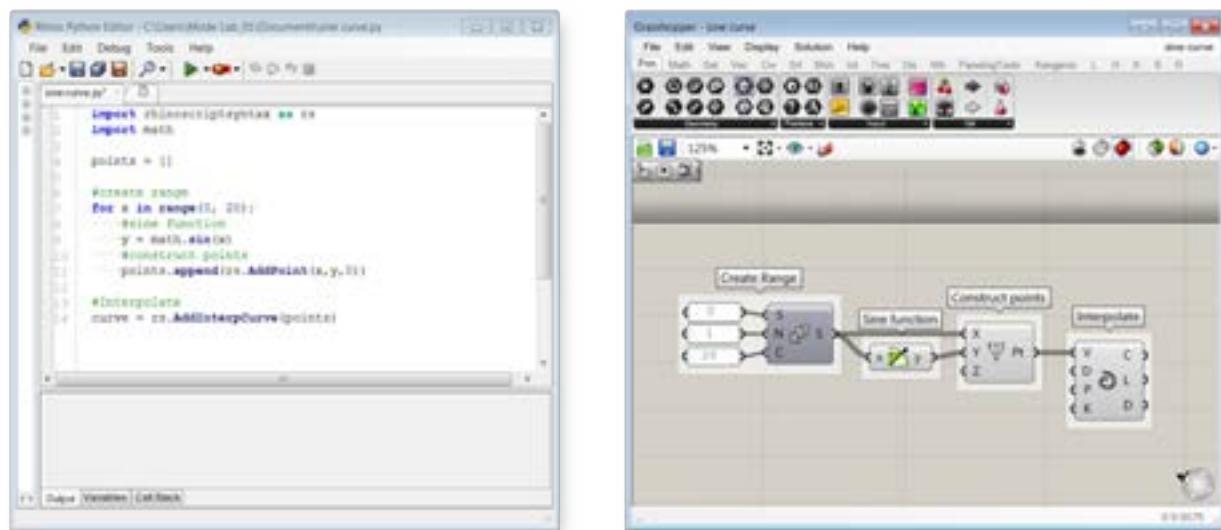
---

**Grasshopper ist ein visueller Programmeditor, der von David Rutten und Robert McNeel & Associates entwickelt wurde. Als Plug-In fuer Rhino3D, ist Grasshopper integriert in eine robuste und vielseitige Modellierungsumgebung. Diese wird von kreativen Profis in verschiedenen Bereichen, wie z.B. Architektur, Ingenieurwissenschaften und Produktdesign genutzt. Zusammen mit Rhino bietet Grasshopper uns die Moeglichkeit praezise parametrische Beziehungen in Modellen zu definieren, die Faehigkeit generative Entwurfsablaeufe zu erkunden und eine Plattform um uebergeordnete Programmierlogik zu entwickeln – und all das in einer intuitiven, graphischen Oberflaeche.**

Die Ursprunge von Grasshopper koennen bis zur Funktionalitaet des "Record History" Knopfes in Rhino3d Version 4 zurueckverfolgt werden. Dieses spezifische Merkmal erlaubte es Anwendern Modellierungsverfahren waehrend des Ablaufes implizit im Hintergrund zu speichern. Falls Du aus vier Kurven mit der eingeschalteten Aufzeichnungsfunktion eine Loftflaeche erstellt hastest, konntest Du die Kontrollpunkte der Kurven bearbeiten und die Geometrie der Flaeche wurde automatisch aktualisiert. Im Jahr 2008 hat David folgende Frage gestellt: "Was wuerde geschehen, wenn man mehr explizite Kontrolle ueber die Aufzeichnungsfunktion haette?". Dies war der Geburtsmoment fuer den Vorlaeufer von Grasshopper - Explicit History. Diese Funktion erlaubte es die gespeicherten Arbeitsablaeufe im Detail zu bearbeiten und befaehigte den Anwender logische Sequenzen zu entwickeln, welche ueber die bestehenden Faehigkeiten von Rhino3Ds eingebaute Funktionalitaet hinausreichen. Sechs Jahre spaeter ist Grasshopper nun eine robuster, visueller Programmieditor, der mit verschiedenen extern entwickelten Plug-Ins erweitert werden kann. Ausserdem wurden dadurch Arbeitsablaeufe von Profis in verschiedenen Industrien grundszaetlich veraendert und eine aktive, globale Gemeinschaft der Anwender beguenstigt.

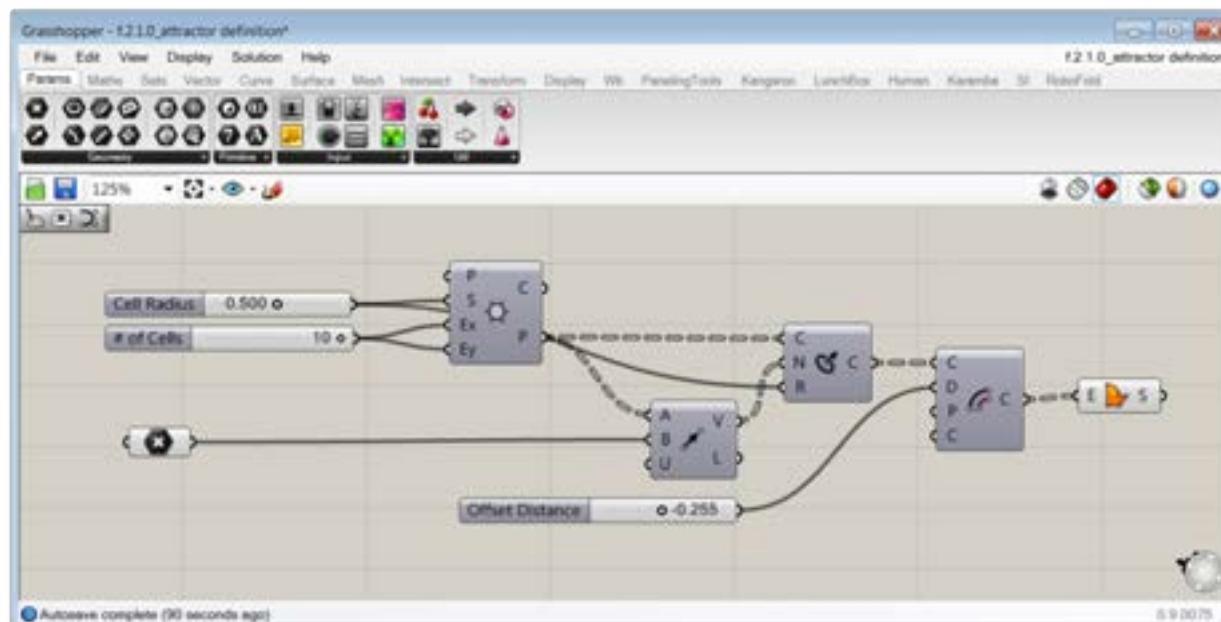
Dieser Primer legt seinen Fokus sowohl auf die Grundlagen um Kernwissen anzubieten, das Du brauchst um in eine regelmaessige Nutzung von Grasshopper einzutauchen, als auch auf einige Sprungbretter fuer die naechsten Schritte in Deiner eigenen kreativen Praxis. Bevor wir in die Beschreibungen, Diagramme und Beispiele einsteigen, welche in der Folge dargeboten werden, sollten wir darueber sprechen, was visuellen Programmierung ist, wie die Grasshopper Benutzeroberflaeche aufgebaut ist, welche grundlegenden Begriffe Grasshopper nutzt und welche "live" Eigenschaften die Rueckkopplung des Ansichtsfensters bzgl. des Benutzererlebnisses hat.

Visueller Programmierung ist ein Paradigma der Computerprogrammierung innerhalb dessen der Benutzer logische Elemente graphisch statt textbasiert manipuliert. Einige der bekanntesten text-basierten Programmiersprachen, wie C#, Visual Basic oder Processing – und naeher an Rhinoceros – Python und Rhinoscript, verlangen von un, dass wir Code schreiben, der an sprachspezifische Syntax gebunden ist. > Im Unterschied dazu erlaubt uns visuelles programmieren funktionale Blöcke in einer Sequenz von Aktionen miteinander zu verbinden. Dabei ist die einzig benoetigte "Syntax" dass die Eingabeparameter Daten des entsprechenden Typs bekommen, und idealerweise, dass das Skript auf das beabsichtigte Resultat hin organisiert ist. Mehr dazu in den Abschnitten ueber die Abgleichung des Datenflusses und das Entwerfen mit Baumstrukturen. Diese Charakteristik des visuellen programmierens senkt die Einstiegsbarriere, welche ueblicherweise bei dem Erlernen einer neuen Sprache - auch bei einer gesprochenen - vorgefunden wird. In den Vordergrund zu stellen ist auch die graphische Benutzeroberflaeche, die Grasshopper fuer Designer in einem vertrauten Umfeld verortet.



Dieses Bild zeigt die Entwicklung einer gezeichneten Sinuskurve in Python und Grasshopper.

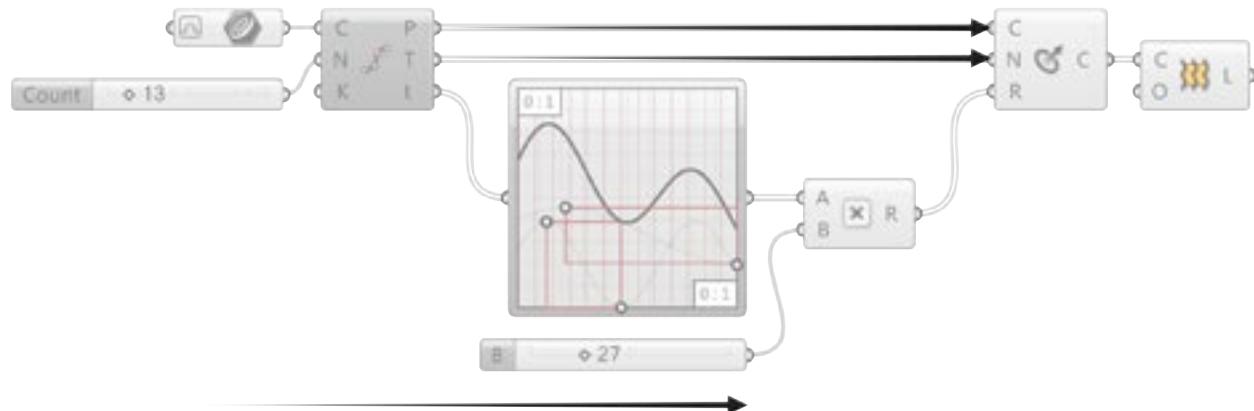
Um bei Grasshopper einzusteigen und seine Fähigkeiten bezüglich visuellen Programmierens nutzen zu können, müssen wir das Programm von der Grasshopper3D.com Webseite laden und auf unserem Computer installieren. Einmal installiert, können wir das Plugin öffnen, indem wir "Grasshopper" in die Rhino Befehlszeile eingeben. Das erste Mal wenn wir diese Eingabe in einer neuen Rhinositzung machen, werden wir die Grasshopper Ladeanzeige sehen, bevor wir das Fenster des Grasshoppereditors sehen. Wir können nun funktionale Blöcke – "Komponenten" genannt- auf der "Leinwand" einfügen, mit Kabeln verbinden und die gesamte "Definition" im .ghx Dateiformat speichern.



Eine Grasshopperdefinition, bestehend aus Komponenten die mit Kabeln auf der Leinwand verbunden sind.

Sobald wir begonnen haben eine Grasshopperdefinition zu entwickeln und Schiebereglern in unserer Leinwand erstellt haben um Geometrien zu kontrollieren, können wir unsere natürliche Intuition nutzen um Verbindungen zwischen den Komponenten zu verstehen, da die Ergebnisse direkt im Rhinosichtsfenster sichtbar werden. Die Verbindung zwischen Grasshopper und Rhino ist grundsätzlich in Echtzeit – wenn wir einen Schieberegler verändern, wird die Folge dieser Handlung im Rahmen der Definition ausgewertet und die Lösung neu berechnet, bevor das Ansichtsfenster aktualisiert wird. Beim Einstieg in Grasshopper ist es von Vorteil, dass die

Vorschau der Geometrie, die wir sehen eine leichtgewichtige Darstellung der Loesung ist und diese automatisch aktualisiert wird. Es ist jetzt wichtig zu wissen, dass sobald Deine Definitionen komplexer werden und auf geschickte Art und Weise den Datenfluss regeln, das Rhinoansichtsfenster den Status der Rechnerloesung wiederspiegeln, um ungewollte Kopfschmerzen zu vermeiden.



Programmfluss von links nach rechts.

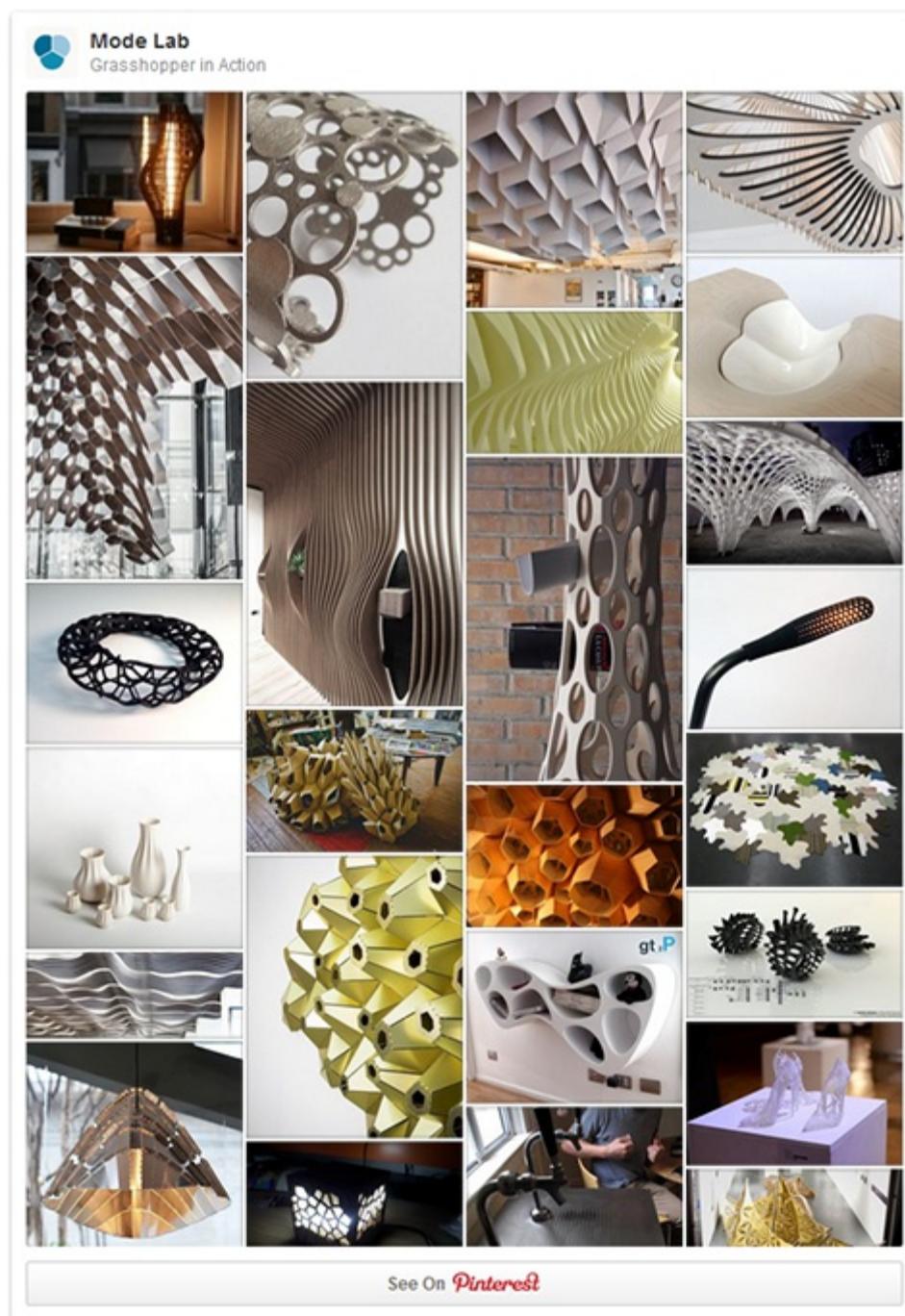
## MERKPUNKTE

- Grasshopper ist ein grafischer Algorithmeneditor, der in das Rhino3D Modellierwerkzeug integriert ist.
- Algorithmen sind schrittweise Prozeduren, die entworfen wurden um eine bestimmte Operation durchzufuehren.
- Du benutzt Grasshopper um Algorithmen zu entwerfen, die dann Aufgaben in Rhino3D automatisieren.
- Ein einfacher Weg um einzusteigen, wenn Du Dir nicht sicher bist, wie eine bestimmte Operation in Grasshopper ausgefuehrt werden kann, waere es den Algorithmus haendisch und schrittweise mit Rhinobefehlen zu erarbeiten.

Sobald Du erstmals beginnst Grasshopper zu erkunden oder Deine Faeigkeiten weiter entwickelst, bist Du ein Teil der weltweiten Grasshoppergemeinschaft. Diese Gemeinschaft besteht aus aktiven Mitgliedern aus vielen verschiedenen Anwendungsgebieten und vielfaeltigen Erfahrungsniveaus. Das Forum auf Grasshopper3D.com ist eine nuetzliche Quelle um Fragen zu stellen, Ergebnisse zu teilen und Wissen zu sammeln. Diese Gemeinschaft hat uns waehrend des Schreibens des Primers sehr geholfen und wir haben uns daran erfreut, wie sich Grasshopper ueber die Jahre entwickelt hat. Herzlich Willkommen!

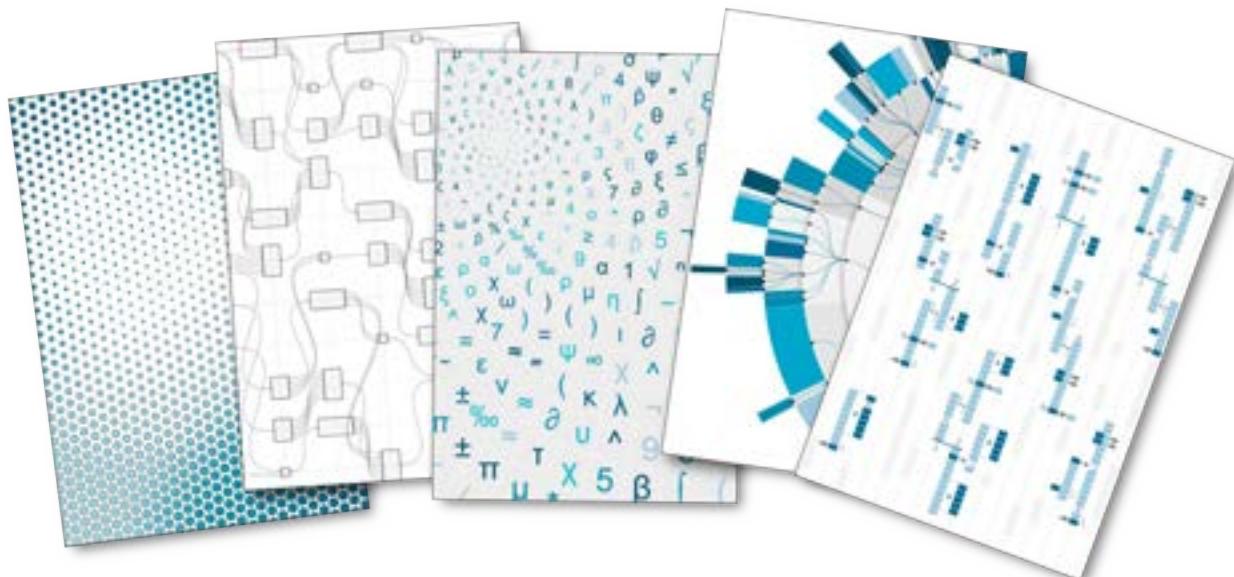
## Grasshopper in Aktion

Folge der Grasshopper in Action Pinnwand auf Pinterest.



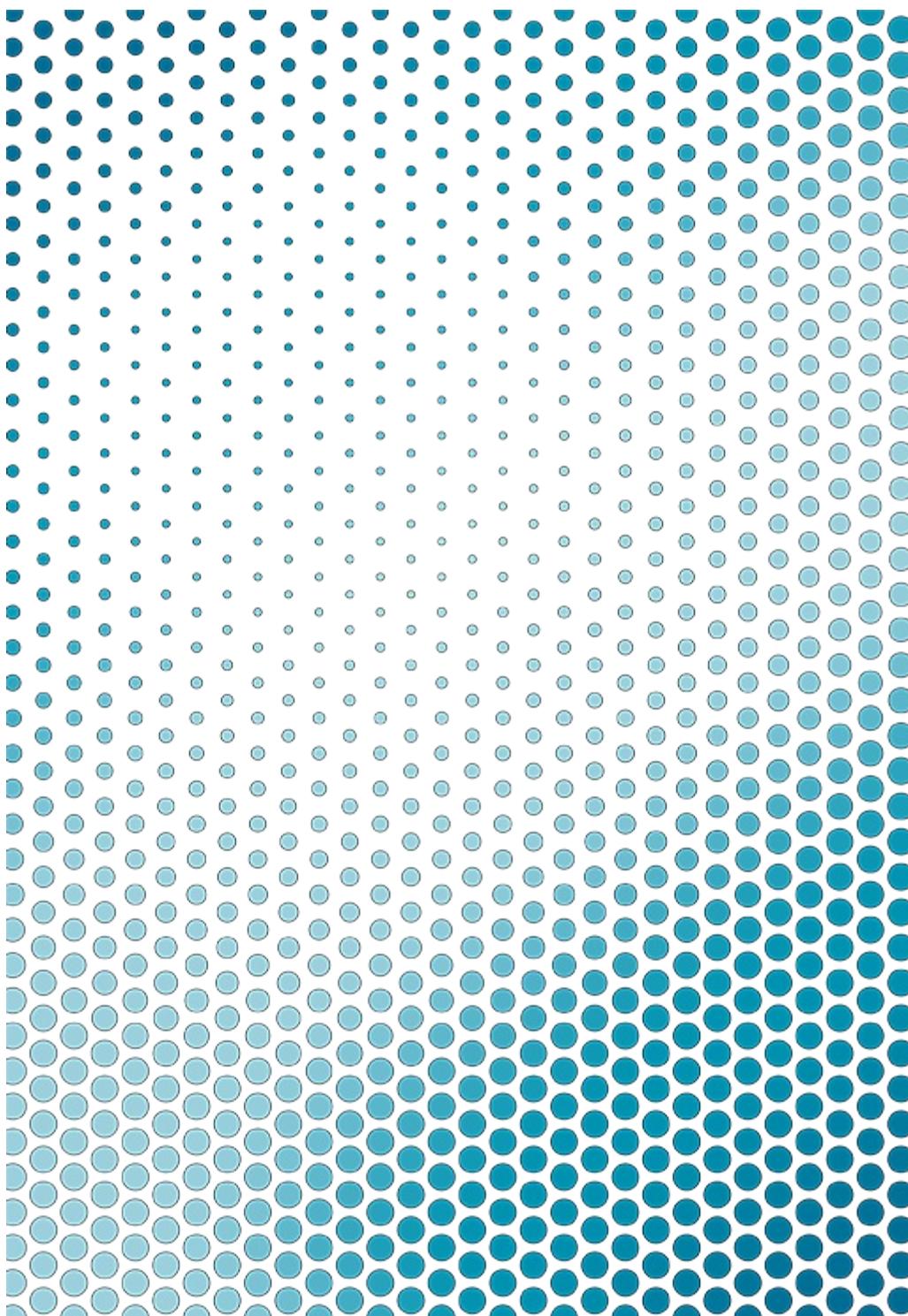
# 1. FUNDAMENT

Ein starkes Fundament ist gebaut um zu bestehen. Diese Ausgabe des Primers fuehrt Schluesselkonzepte ein und erklaert parametrische Modellierung in Grasshopper.



## 1.1. HALLO GRASSHOPPER

**Grasshopper** ist ein graphischer Editor für Algorithmen, der in die **Rhino3D** Modellierwerkzeuge integriert ist. Du benutzt **Grasshopper** um Algorithmen zu entwerfen, die dann Aufgaben in **Rhino3D** automatisieren.



## 1.1.1. INSTALLATION UND START VON GRASSHOPPER

Das Grasshopper plugin wird häufig aktualisiert, weshalb Du sicherstellen solltest, immer die neueste Version installiert zu haben.

Es sei darauf hingewiesen, dass es momentan keine Version von Grasshopper für Mac gibt.

### 1.1.1.1. HERUNTERLADEN

Um das Grasshopper Plug-In herunterzuladen, besuche die Grasshopper Webseite. Klicke auf die Registrierkarte Download am oberen Rand der Seite, und wenn Du auf der nächsten Seite dazu aufgefordert wirst, gebe Deine Emailadresse ein. Nun kannst Du den Link zum Herunterladen mit der rechten Maustaste anklicken und die Option "Ziel speichern unter" aus dem Menü auswählen. Wähle den Speicherort auf Deiner Festplatte (Hinweis: Die Datei kann nicht über eine Netzwerkverbindung geladen werden, deshalb muss die Datei lokal auf Deiner Festplatte gespeichert werden.) und speichere dort die ausführbaren Dateien.



Lade Grasshopper von der [grasshopper3d.com](http://grasshopper3d.com) Webseite herunter.

### 1.1.1.2. INSTALLATION

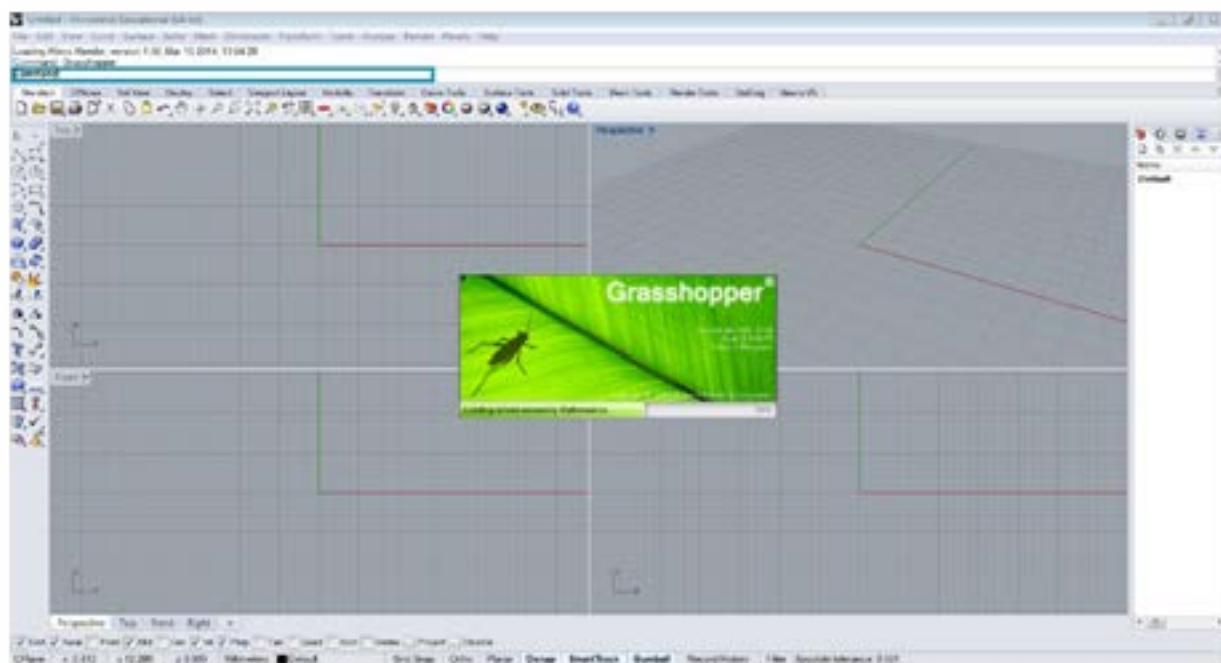
Wähle "ausführen" von den Downloadoptionen aus und folge den Anweisungen im Installationsdialog. (Hinweis: Du musst Rhino 5 bereits auf Deinem Computer installiert haben, damit das Plug-In ausgeführt werden kann.).



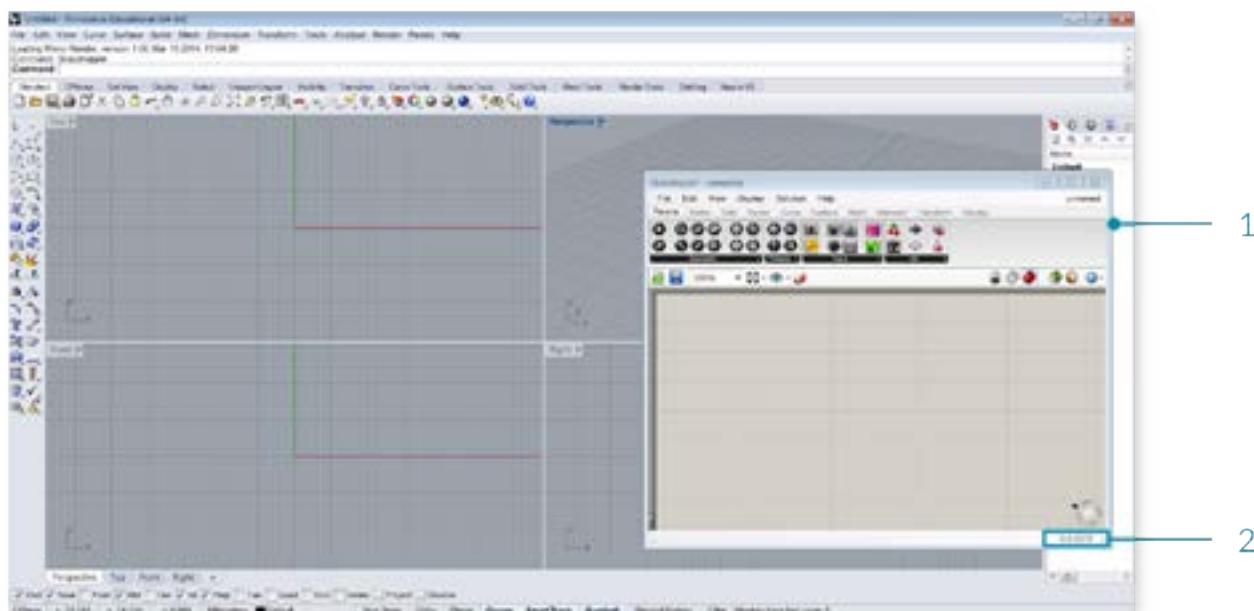
Folge den Anweisungen des Installationsdialogs.

### 1.1.1.3. STARTEN

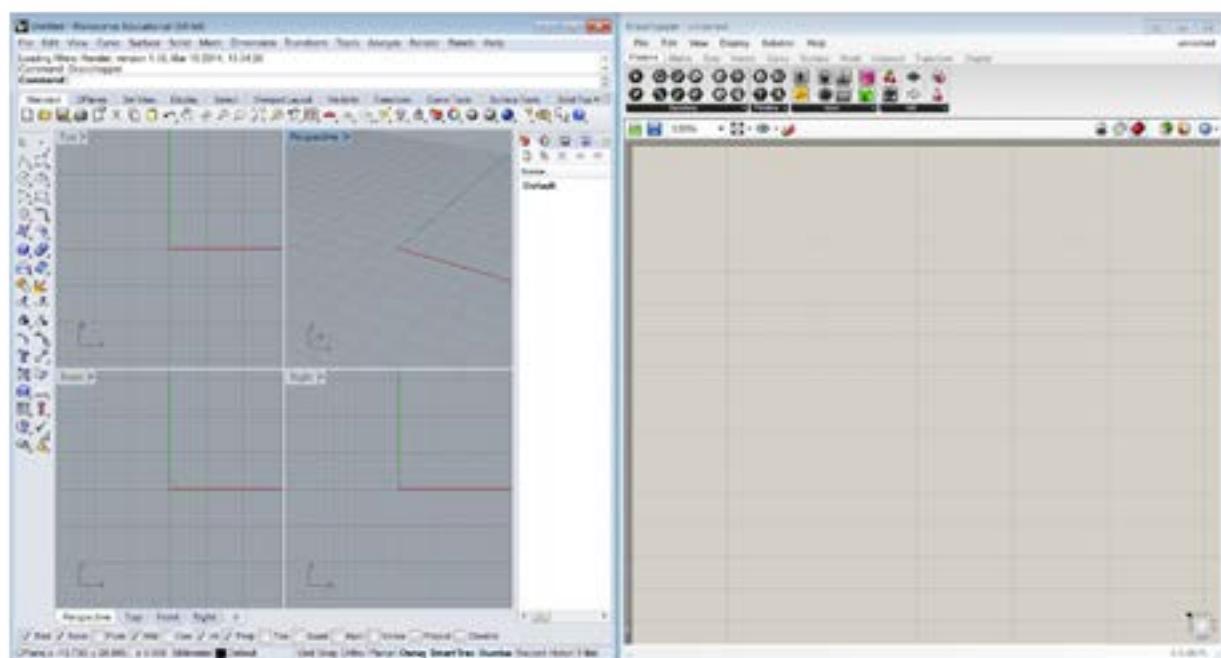
Um Grasshopper zu starten, gebe "Grasshopper" in die Rhino Befehlszeile ein. Wenn Du Grasshopper startest, wirst Du als erstes ein neues Fenster vor dem Rhinofenster schweben sehen. In diesem Fenster kannst Du knotenbasierte Programme erstellen, welche die vielfältige Funktionalität von Rhino automatisieren können. Eine bewährte Methode ist es, die Fenster so nebeneinander anzuordnen, dass sie sich nicht überlappen und Grasshopper nicht die Ansichtsfenster von Rhino versperrt.



Gebe "Grasshopper" in die Rhino Befehlszeile ein, um das Grasshopper Plug-In zu starten.



1. Das Grasshopperfenster schwebt ueber den Ansichtsfenstern von Rhino.
2. Grasshopper zeigt die Versionsnummer am unteren Ende des Fensters an.



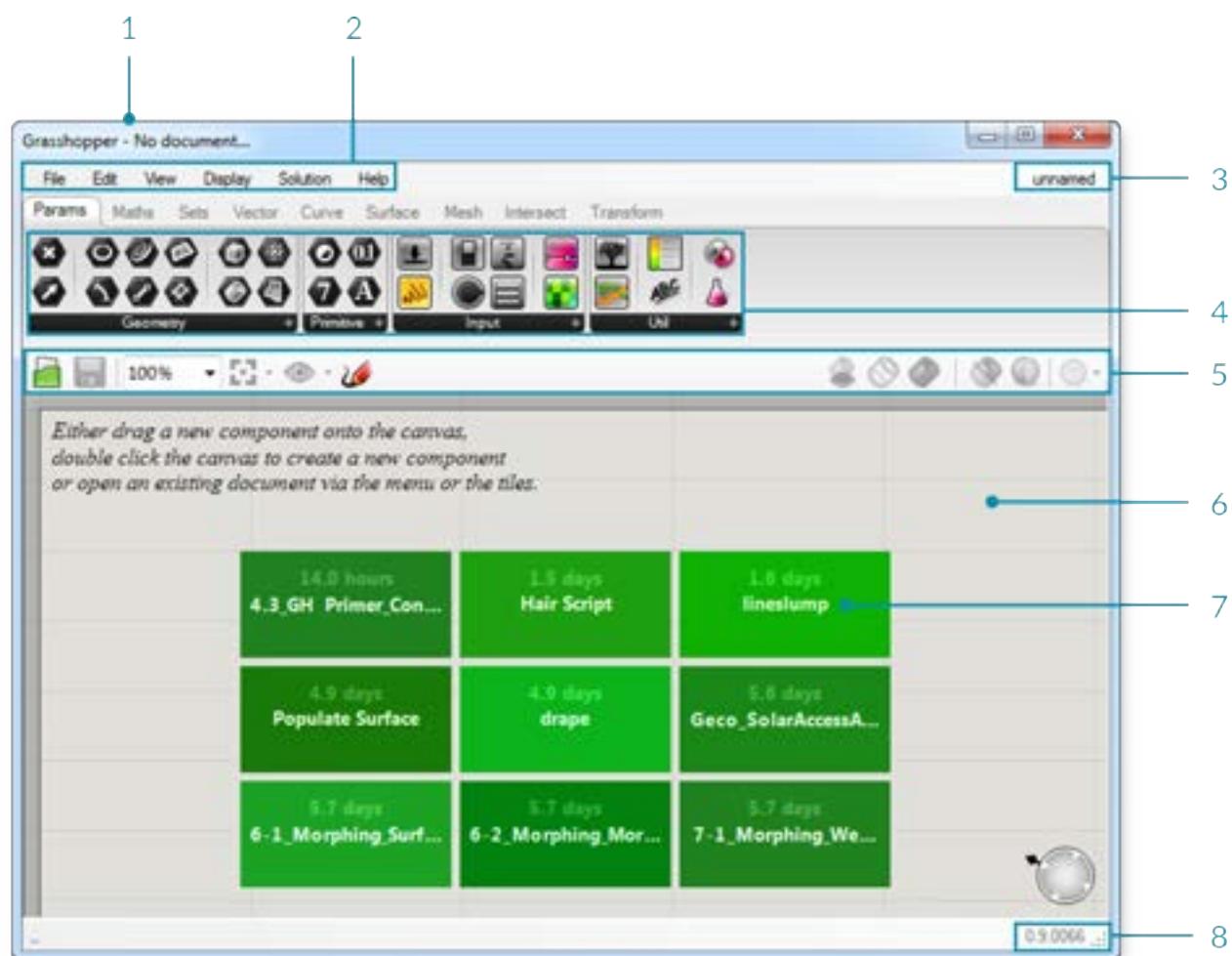
Unterteile den Bildschirm, so dass Grasshopper nicht die Ansichtsfenster von Rhino versperrt. Du kannst dies erreichen, indem Du die Fenster auf gegeneuberliegende Seiten des Bildschirms ziehst, oder indem Du die Windowstaste gedrueckt haeltst und die Cursorpfeile nach links oder rechts drueckst.

## 1.1.2. DIE GRASSHOPPER BENUTZEROBERFLÄCHE

Der visuelle “plug-and-play” Stil von Grasshopper gibt Designern die Fähigkeit kreative Problemlösung mit neuen regelbasierten Systemen in einer fluiden, graphischen Benutzeroberfläche zu kombinieren.

Lass uns damit beginnen die Benutzeroberfläche von Grasshopper zu erkunden. Grasshopper ist eine visuelle Programmieranwendung, die es erlaubt Programme, Definitionen genannt, zu erstellen, indem Komponenten in das zentrale Editorfenster (Canvas genannt) gezogen werden. Die Ausgaben dieser Komponenten werden mit den Eingaben folgender Komponenten verbunden und erzeugen so einen Informationsfluss, der von links nach rechts gelesen wird. Lasst uns mit den Grundlagen anfangen.

Angenommen, Du hast bereits das Grasshopper Plugin installiert (vgl. F.0.0), gib das Wort “Grasshopper” in die Rhino Befehlszeile ein und öffne damit den Grasshopper Editor. Die Grasshopper Oberfläche beinhaltet eine Anzahl an Elementen, die einem Rhino Nutzer sehr bekannt sind. Schauen wir uns die neuen Bestandteile nun an.



1. Fenstertitelleiste.
2. Menüleiste
3. Dateibrowsersteuerung.
4. Komponentenpalette.
5. Canvas Werkzeugeleiste.
6. Canvas.
7. Dieser Bereich, der durch ein Raster von rechteckigen Feldern gekennzeichnet ist, stellt eine Oberfläche bereit mit der die zuletzt geöffneten Dateien geöffnet werden können. Das 3x3 Menü zeigt die zuletzt genutzten Dateien in chronologischer Reihenfolge und wird rote Felder anzeigen, sobald die
8. Version

entsprechende Datei nicht gefunden werden kann (was vorkommt, wenn Du Dateien in einen neuen Ordner verschiebst oder löscht).

8. Die Statusleiste nennt die von Dir genutzte Version von Grasshopper, welche gerade auf Deiner Maschine installiert ist. Falls eine neue Version verfügbar ist, wird ein Pop-Up-Fenster erscheinen und Dir die Anweisungen zum Download der neuesten Version zur Verfügung stellen.

### 1.1.2.1. DIE FENSTERTITELLEISTE

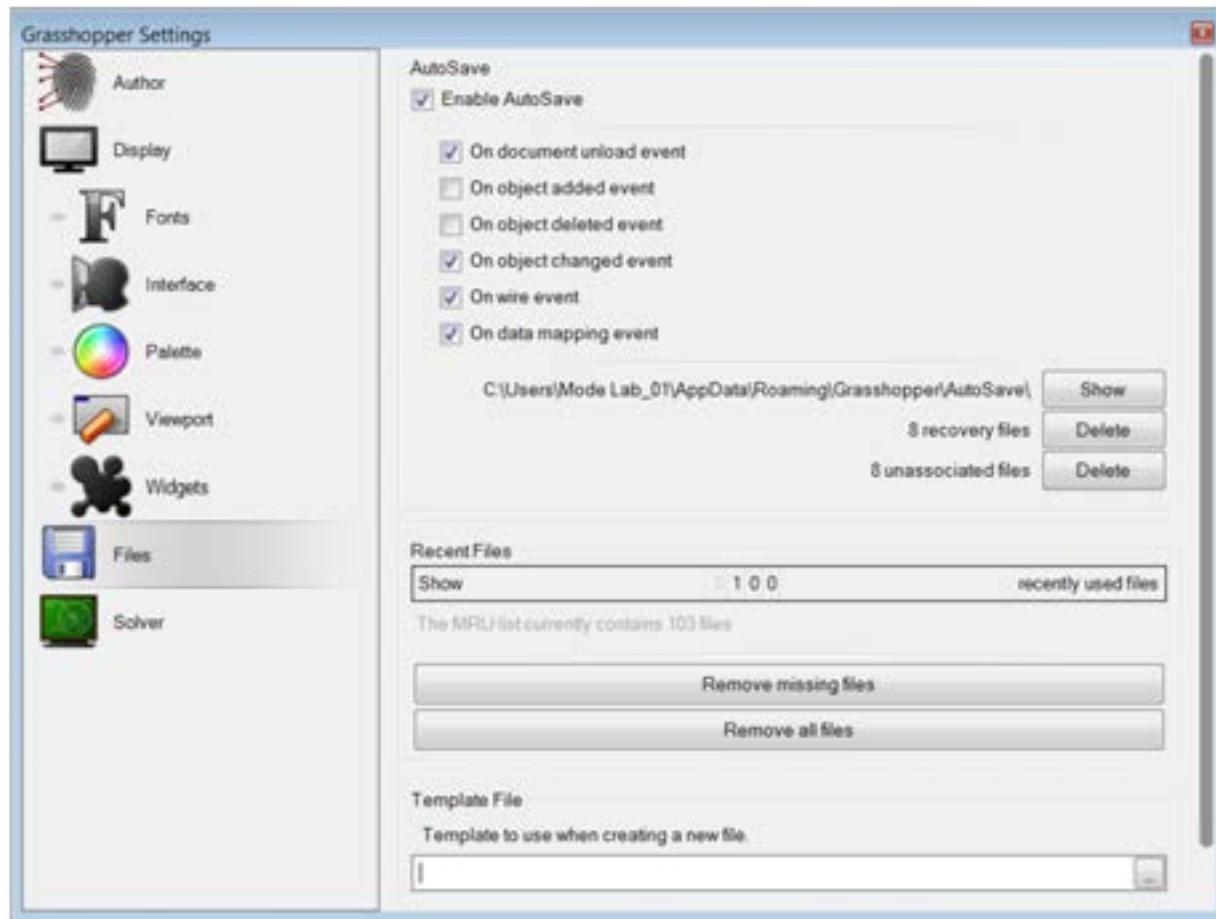
Die Fenstertitelleiste des Editors verhält sich unterschiedlich von den meisten anderen Dialogen in Microsoft Windows. Falls das Fenster nicht minimiert oder maximiert ist, wird der Dialog auf Doppelklick in diese Leiste zu einer Leiste auf Deinem Bildschirm minimiert. Dies ist eine ausgezeichnete Möglichkeit zwischen dem Plugin und Rhino zu wechseln, weil der Editor minimiert wird ohne dass er sich zur Unterseite des Bildschirms bewegt oder hinter einem anderen Fenster verschwindet. Merke, dass, falls Du den Editor schließt, die Grasshopper Geometrievorschau im Rhino Ansichtsfenster verschwindet, die Datei jedoch nicht geschlossen wird. Das nächste Mal, wenn Du den „Grasshopper“ Befehl in die Rhino Befehlszeile eingibst, wird das Fenster im selben Zustand, mit denselben Dateien geöffnet, wieder erscheinen. Das passiert, weil Deine Grasshopper Session aktiv bleibt, bis Rhino geschlossen wird, sobald Du sie einmal von der Rhino Eingabe aus geöffnet hast.

### 1.1.2.2. MENÜLEISTE

Die Titelleiste ist ähnlich wie Windowsmenüleisten, außer der Dateibrowsersteuerung an der rechten Seite (siehe nächster Abschnitt). Das Dateimenü stellt, zusätzlich zu einigen nützlichen Werkzeugen, welche Dich Bilder von Deinem aktuellen Grasshopper Dokument exportieren lassen (siehe Export von schnellen Bildern und Export von Hi-Res Bildern), die typischen Funktionen zur Verfügung (z.B. Neue Datei, Öffnen, Speichern, etc.). Du kannst die verschiedenen Aspekte der Benutzeroberfläche steuern, indem Du die Menüs „View“ und „Display“ verwendest, während das Menü „Solution“ Dich die verschiedenen Eigenschaften der Berechnung der graphischen Lösung durch den Solver verwalten lässt.

Es ist es wert anzumerken, dass viele Einstellungen der Anwendung durch den „Preferences“ Dialog gesteuert werden können, der im „File“ Menü zu finden ist. Der „Author“ Bereich erlaubt es Dir Deine persönlichen Metadaten einzutragen, welche in jedem Grasshopper Dokument gespeichert werden, während der „Display“ Abschnitt Dir die Möglichkeit gibt, die Feinabstimmung von Aussehen und Handhabung vorzunehmen. Der „Files“ Bereich erlaubt es Dir, Dinge zu spezifizieren, wie beispielsweise die Häufigkeit und der Speicherort für Autosavedateien (im Falle des unbeabsichtigten Schließens oder eines Absturzes der Anwendung). Schließlich, im „Solver“ Bereich, kannst Du die Kern- und Drittanbieter-Plugins verwalten, welche die Funktionalität erweitern.

Merke: Sei vorsichtig, wenn Du Tastaturkürzel verwendest, da diese im aktiven Fenster angewendet werden, welches entweder das von Grasshopper oder von Rhino sein kann. Es ist schnell passiert ein Tastaturkürzel anzuwenden, ohne zu merken, dass es im falschen Fenster angewendet wurde und einen ungewollten Befehl ausgelöst hat.



Der „Preferences“ Dialog erlaubt es Dir viele Einstellungen der Grasshopper Anwendung zu steuern.

### 1.1.2.3. DATEIBROWSERSTEUERUNG

Der Dateibrowser erlaubt es Dir schnell zwischen verschiedenen geladenen Dateien umzuschalten, indem diese in der Drop-Down Liste ausgewählt werden. Der Zugang zu geöffneten Dateien durch die Drop-Down Liste des Dateibrowser ermöglicht es Teile von geöffneten Definitionen schnell zu kopieren und einzufügen. Klicke einfach auf den aktiven Dateinamen in der Dateibrowsersteuerung und eine fallende Liste aller geöffneten Dateien wird angezeigt werden (zusammen mit kleinen Vorschaubildern jeder geöffneten Definition). Du kannst ebenfalls Alt+Tab drücken, um schnell zwischen offenen Grasshopper Dokumenten umzuschalten.

Natürlich kannst Du auch durch den Standarddialog „Open file“ gehen, um Grasshopper Definitionen zu laden, auch wenn Du einfach eine Grasshopper Datei auf den Canvas ziehen kannst um eine bestimmte Definition zu öffnen.

Grasshopper ist ein Plugin, das „über“ Rhino arbeitet und deshalb eigene Dateitypen bereitstellt. Der Standarddateityp ist eine binäre Datendatei, die mit der Erweiterung .gh gespeichert wird. Der andere Dateityp ist als Grasshopper XML Datei bekannt und verwendet die Erweiterung .ghx. Der XML (Extensible Markup Language) Dateityp nutzt Tags um Objekte und Objektattribute zu definieren (ähnlich einem HTML Dokument), jedoch programm spezifische Tags, um die Daten innerhalb eines Objektes zu beschreiben. Weil XML Dateien als Textdokumente formatiert sind, kannst Du die Grasshopper XML Dateien in einem Texteditor wie NotePad öffnen, um hinter die Kulissen der Kodierung zu sehen.

Grasshopper hat verschiedene Methoden mit welchen es Dateien öffnen kann, und Du wirst angeben müssen, welche der Optionen Du anwenden möchtest, wenn Du eine bestimmte Methode nutzt.

**Open File:** Wie der Name suggestiert, öffnet diese Option einfach eine beliebige Definition, die Du in den Canvas ziehen kannst.

**Insert File:** Du kannst diese Option benutzen, um eine bestehende Datei in das aktuelle Dokument als lose Komponenten einzufügen.

**Group File:** Diese Methode wird eine Datei in ein bestehendes Dokument einfügen, jedoch die Objekte miteinander gruppieren.

**Cluster File:** Ähnlich zur Option „group function“, wird hier eine Datei in ein bestehendes Dokument eingefügt, wobei die Objekte der Gruppe in einem Clusterobjekt zusammengefügt werden.

**Examine File:** Erlaubt es Dir eine Datei in einem geschlossenen Zustand zu öffnen, was bedeutet, dass Du Dir die Datei ansehen, aber keine Änderungen an ihr vornehmen kannst.

Grasshopper hat auch eine Autosave Funktion, die regelmäßig auf Grundlage der Nutzerinteraktion ausgelöst wird. Eine Liste der Autosave Präferenzen kann im Dateimenü in der Menüleiste gefunden werden. Wenn eine aktive Instanz von Rhino geschlossen wird, erscheint ein Pop-Up Dialog, der abfragt, ob Du Grasshopper Dateien speichern möchtest, die geöffnet waren, als Rhino geschlossen wurde.

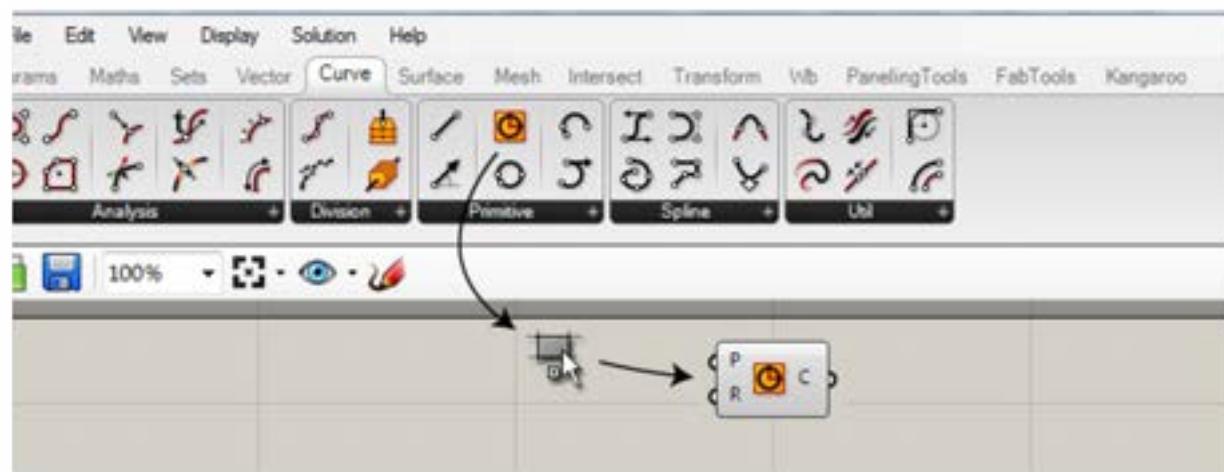
Autosave arbeitet nur, wenn eine Datei mindestens schon einmal gespeichert wurde.



Ziehe Dateien auf den Canvas.

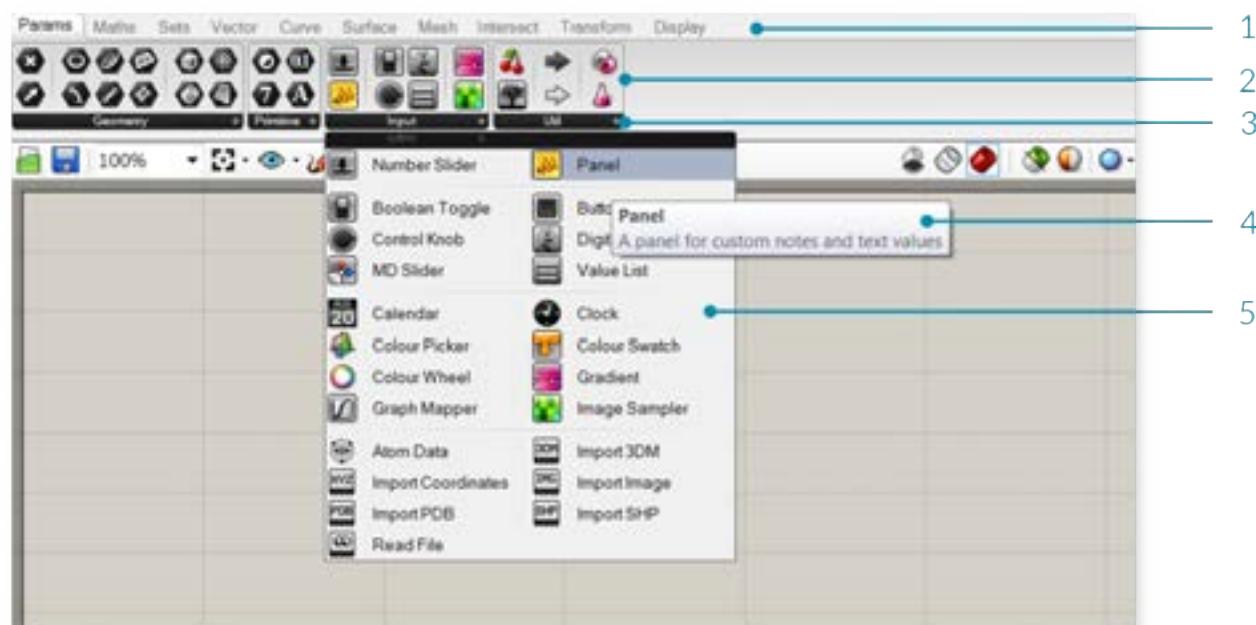
### 1.1.2.4. KOMPONENTEN PALETTEN

Dieser Bereich organisiert Komponenten in Kategorien und Unterkategorien, Kategorien werden als Reiter dargestellt und Unterkategorien in Drop-Down Paneelen. Alle Komponenten gehören bestimmten Kategorien an. Diese Kategorien wurden benannt, um Dir zu helfen bestimmte Komponenten zu finden, wenn Du danach suchst (z.B. "Params" für alle einfachen Datentypen oder "Curves" für alle kurvenbezogenen Werkzeuge). Um Komponenten zum Canvas hinzuzufügen kannst Du entweder die Objekte im Drop-Down Menü anklicken oder sie direkt vom Menü auf den Canvas ziehen.



Ziehe eine Komponente von der Palette direkt auf den Canvas um sie einzufügen.

Da es mehr Komponenten in jeder Unterkategorie geben kann, als in die Palette passen, wird nur eine begrenzte Anzahl in dem entsprechenden Paneel angezeigt. Die Höhe der Komponentenpalette und die Breite des Grasshopperfensters können angepasst werden, um mehr oder weniger Komponenten pro Unterkategorie darzustellen. Um ein Menü aller Komponenten einer bestimmten Unterkategorie zu sehen, klicke einfach auf die schwarze Leiste an der Unterseite des jeweiligen Paneels einer Unterkategorie. So wird ein Drop-Down Menü geöffnet, das Dir den Zugang zu allen Komponenten der Unterkategorie ermöglicht.



1. Kategoriereiter

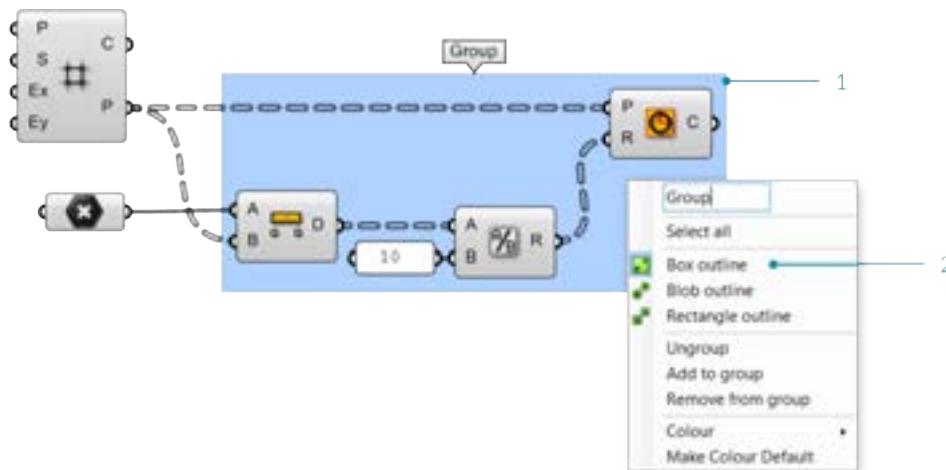
2. Unterkategoriepaneel
3. Klicke auf die schwarze Leiste um das Unterkategoriemenü zu öffnen.
4. Bewege Deine Maus über eine Komponente um eine Kurzbeschreibung zu erhalten.
5. Drop-down Menü.

### 1.1.2.5. DER CANVAS

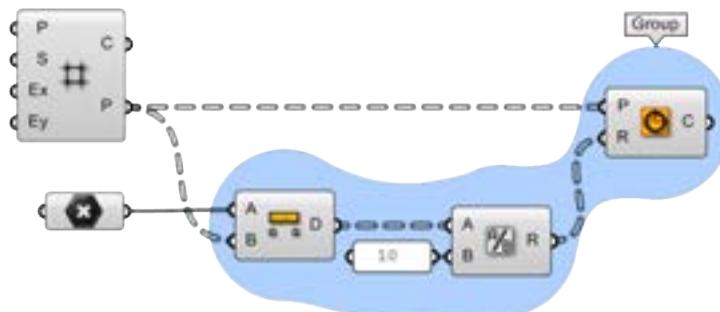
Der Canvas ist die primäre Arbeitsoberfläche um Grasshopperdefinitionen zu erstellen. Hier kannst Du mit den verschiedenen Elementen Deines visuellen Programms interagieren. Du kannst damit beginnen auf dem Canvas zu arbeiten, indem Du Komponenten platzierst und diese mit Kabeln verbindest.

### 1.1.2.6. GRUPPIEREN

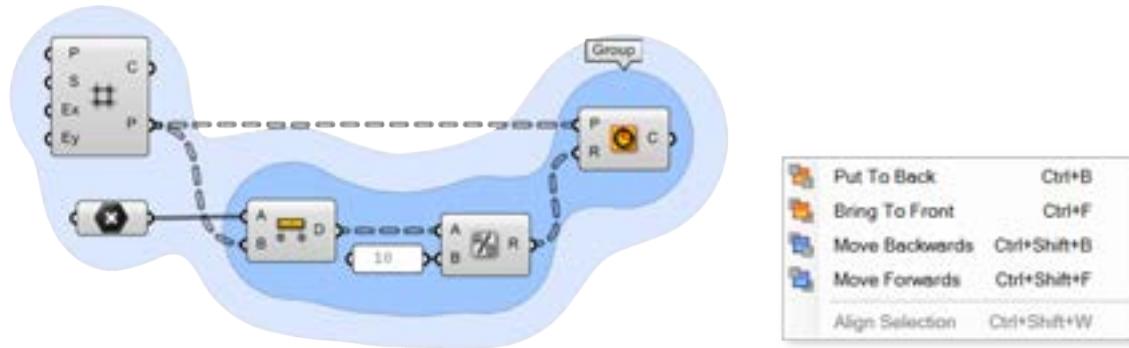
Komponenten auf dem Canvas miteinander zu gruppieren kann insbesondere nützlich sein um die Lesbarkeit und die Verständlichkeit von Definitionen zu verbessern. Gruppieren gibt Dir die Möglichkeit mehrere Komponenten schnell auszuwählen und auf dem Canvas zu bewegen. Du kannst eine Gruppe erstellen, indem Du **Ctrl+G** eingibst, während Du die gewünschten Komponenten ausgewählt hast. Eine alternative Methode kannst Du unter dem „Group Selection“ Knopf des „Edit“ Menüs in der Menüleiste finden. Benutzerdefinierte Parameter für Gruppenfarben, -transparenz, -namen und -konturliniendarstellung können mit einem Rechtsklick auf ein Gruppenobjekt eingestellt werden.



1. Eine Gruppe von Komponenten mit einer Boxlinienkontur gerahmt.
2. Rechtsklicke irgendwo auf die Gruppe um den Namen und die Darstellung der Gruppe zu bearbeiten.



Du kannst eine Gruppe auch mit einem Meta Ball Algorithmus darstellen, indem Du eine Blob Konturlinie auswählst.

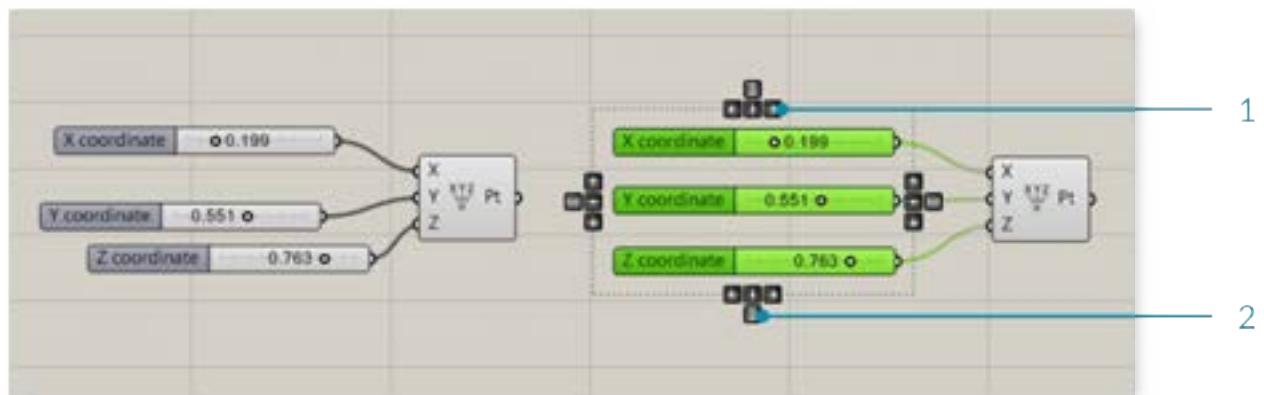


Zwei Gruppen sind hier ineinander verschachtelt, die Farbe (hellblau) wurde bei der äußereren Gruppe verändert um die Lesbarkeit der einzelnen Gruppen zu verbessern. Gruppen werden „hinter“ die Komponenten gezeichnet, die sie beinhalten und, wie in diesem Fall, mit verschiedenen Tiefen. Um dies zu verändern gehe zu Edit > Arrange in der Menüleiste.

### 1.1.1.7. WIDGETS

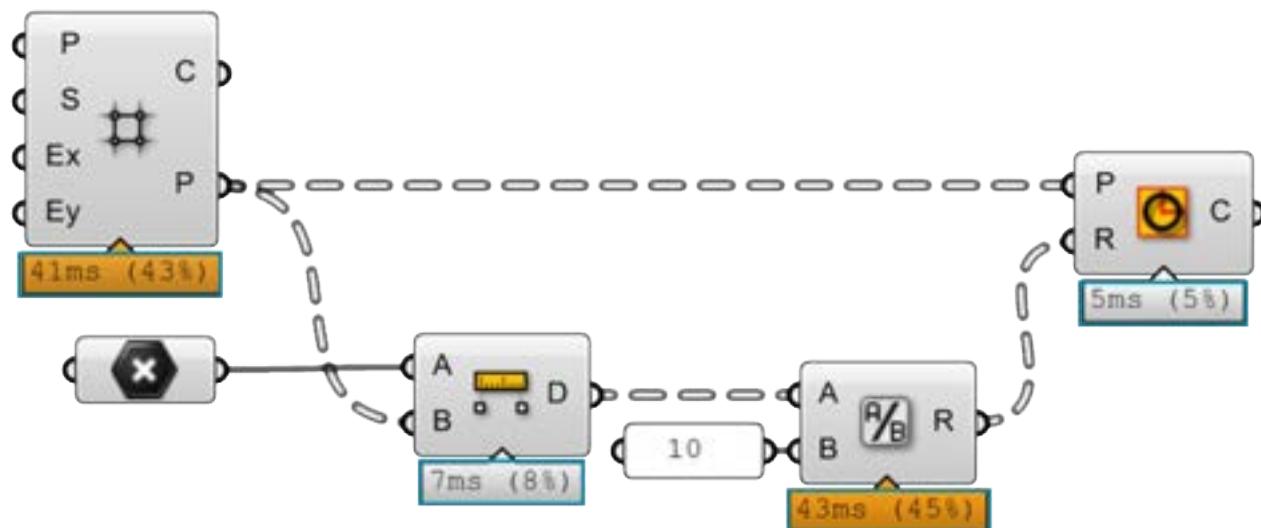
In Grasshopper sind einige Widgets verfügbar, die Dir helfen können nützliche Aktionen auszuführen. Du kannst sie im Display Menü der Menüleiste an- bzw. ausschalten. Nachfolgend werden wir uns einige der am häufigsten benutzten Widgets ansehen.

**Das Align Widget** Ein nützliches Benutzeroberflächen-Widget, welches Dir helfen kann Deinen Canvas in Ordnung zu halten ist das Align Widget. Du kannst es nutzen, indem Du mehrere Komponenten zur gleichen Zeit auswählst und eine der Optionen in der Strichlinie auswählst, welche die ausgewählten Komponenten umgibt. Du kannst die Komponenten links, in der vertikalen Mitte, rechts oder oben, im horizontalen Zentrum, unten oder gleichmäßig verteilt über die Benutzeroberfläche ausrichten. Wenn Du gerade beginnst, wirst Du feststellen, dass diese Werkzeuge Dir manchmal in den Weg kommen (hier kannst Du den Fehler machen alle Komponenten auf einer Stelle zusammenzuführen). Jedoch sind sie mit etwas Übung sehr hilfreich, wenn es darum geht strukturierte Graphen mit einer hohen Lesbarkeit und Verständlichkeit zu erstellen.



1. Rechtsbündiges Ausrichten.
2. Vertikale Verteilung.

**Das Profiler Widget** Der Profiler listet die Laufzeiten für den ungünstigsten Fall für die einzelnen Parameter und Komponenten, um Dir zu erlauben die Engpässe in Netzwerken zu finden und die Laufzeiten von verschiedenen Komponenten miteinander zu vergleichen. Die Widgets sind standardmäßig ausgeschalten.

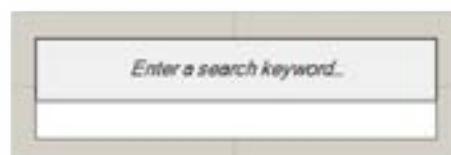


Das Profiler Widget gibt Dir visuelles Feedback, welche Komponenten Deiner Definition längere Laufzeiten verursachen können.

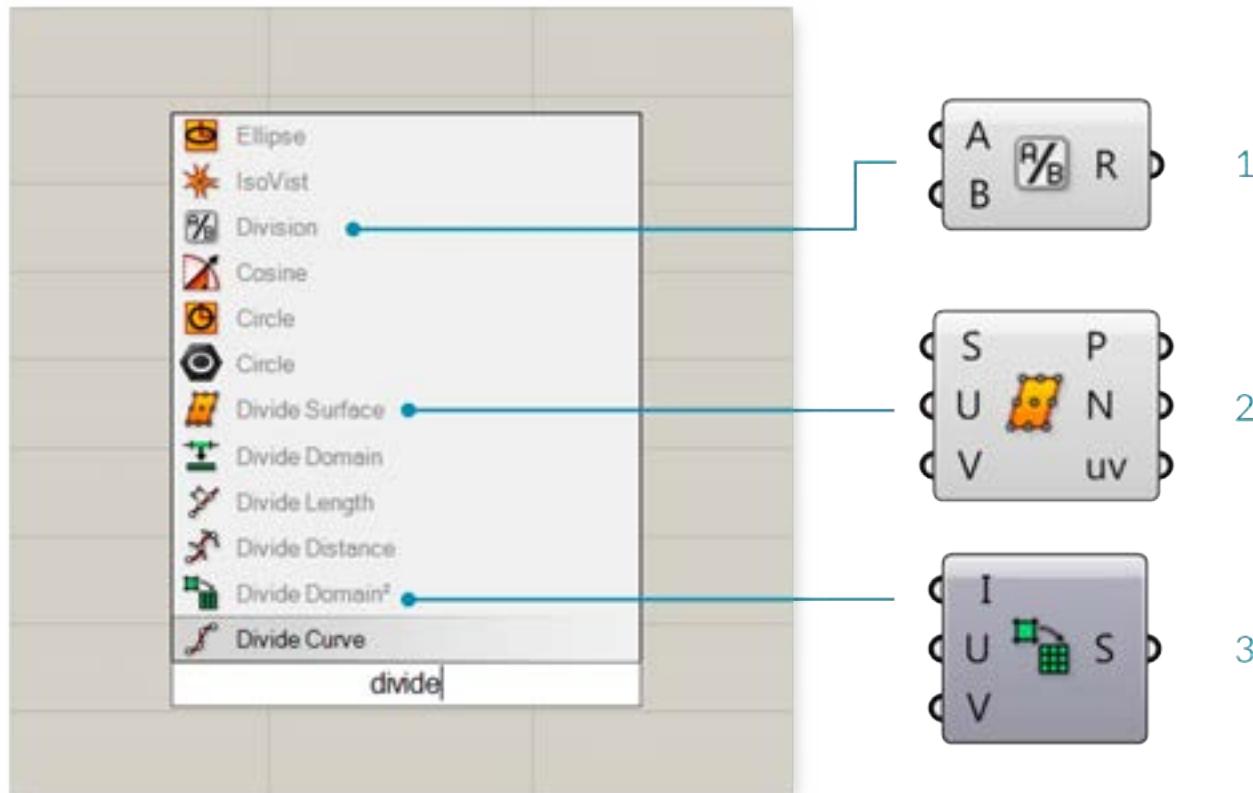
**Das Markov Widget** Dieses Widget nutzt Markovketten um, basierend auf Deinem vergangenen Verhalten, „vorherzusehen“, welche Komponenten Du vielleicht als nächstes nutzen möchtest. Eine Markovkette ist ein Prozess, der aus einer begrenzten Anzahl verschiedener Zustände (oder Ebenen) und bekannten Wahrscheinlichkeiten besteht. Es kann einige Zeit dauern, bis sich dieses Widget an ein bestimmtes Nutzerverhalten gewöhnt hat, aber über einige Zeit sollte es beginnen Dir die Komponenten vorzuschlagen, welche Du als nächstes nutzen möchtest. Das Markov Widget kann, abhängig von Deiner vorausgegangenen Aktivität, bis zu fünf verschiedene Komponenten vorschlagen. Du kannst auf das Markov Widget rechtsklicken, um es in eine bestimmte Ecke des Canvas zu heften oder es komplett zu verstecken (die Standardposition ist in der linken unteren Ecke des Canvas).

### 1.1.2.8. NUTZUNG DER SUCHFUNKTION

Obwohl einige Gedanken in die Platzierung der einzelnen Komponenten in den Komponentenpaneelen gegangen sind um es für den Nutzer intuitiv zugänglich zu machen, ist es manchmal schwierig bestimmte Komponenten, welche tief in den Kategoriepaneelen verborgen sind auszumachen. Glücklicherweise kannst Du auch jede Komponente unter ihrem Namen finden, wenn Du auf einen leeren Bereich im Canvas doppelklickst. Dies wird eine Pop-Up Suchleiste öffnen. Gib einfach den Namen der Komponente ein, nach der Du suchst und Du wirst eine Liste von Parametern und Komponenten erhalten, die Deiner Anfrage entsprechen.



Doppelklicke irgendwo auf den Canvas um die Schlüsselwortsuche für eine bestimmte Komponente aus den Komponentenpaneelen zu aktivieren.



Eine Suche nach "divide" listet eine Vielfalt von Komponenten.

1. Division operator Komponente.
2. Divide Surface Komponente.
3. Divide Domain2 Komponente.

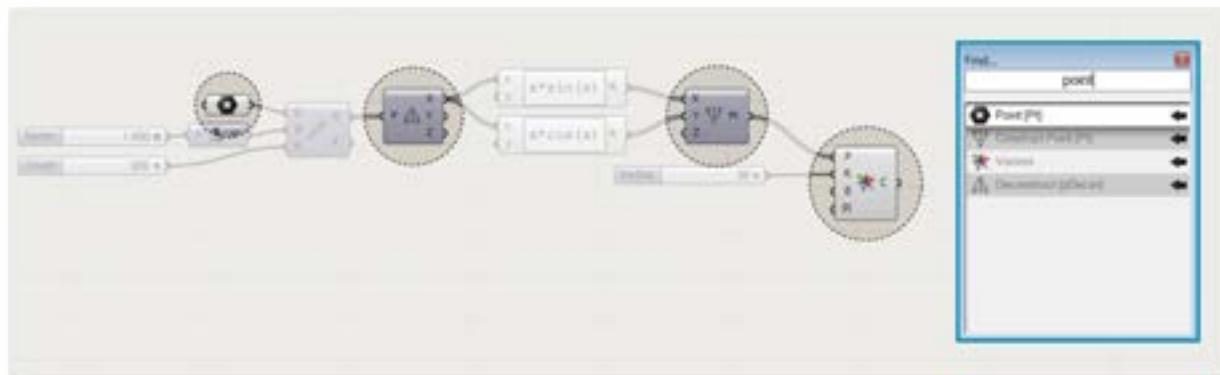
### 1.1.2.9. DIE FIND FUNKTION

Es gibt buchstäblich hunderte (wenn nicht tausende) von Grasshopper Komponenten, welche Dir zur Verfügung stehen und es kann für einen Einsteiger ziemlich entmutigend sein zu wissen wo bestimmte Komponenten in den Komponentenpaletten zu finden sind. Die schnellste Lösung ist es irgendwo auf den Canvas doppelzuklicken und eine Suchanfrage für die Komponente einzugeben. Was jedoch tun, wenn Du eine bestimmte Komponente suchst, die bereits auf dem Canvas platziert wurde? Kein Grund zur Sorge. Mit einem Rechtsklick irgendwo auf den Canvas oder mit einem Druck auf die F3 Taste kannst Du die „Find“ Funktion aufrufen. Beginne einfach, indem Du den Namen der gesuchten Komponente eingibst.

Die „Find“ Funktion verwendet sehr ausgereifte Algorithmen, welche nicht nur nach einer Instanz eines bestimmten Komponentennamens in einer Definition suchen (ein Komponentenname ist der Titel der Komponenten unter der er im Komponentenpaneel gefunden werden kann und kann vom Nutzer nicht geändert werden), sondern auch nach einzigartigen Signaturen, welchen wir bestimmten Komponenten zugeordnet haben können (auch Kosenamen genannt). Die „Find“ Funktion kann auch für die Suche nach Komponententypen auf dem Canvas verwendet werden oder nach Inhalten von Paneelen, Skizzen und Gruppeninhalten suchen. Sobald die „Find“ Funktion einen Treffer gefunden hat, wird es automatisch den Rest der Definition ausgrauen und eine Strichlinie um die hervorgehobene Komponente zeichnen. Wenn mehrere Treffer vorliegen, wird eine Liste der zutreffenden Komponenten für die Suchanfrage in der „Find“ Dialogbox angezeigt und sobald Du mit der Maus über einen Eintrag fährst, wird die entsprechende Komponente im Canvas grün dargestellt.



Indem Du irgendwo auf den Canvas rechtsklickst oder die F3 Taste drückst, kannst Du die „Find“ Funktion aufrufen. Beginne mit der Eingabe des Namens der Komponente nach der Du suchst.



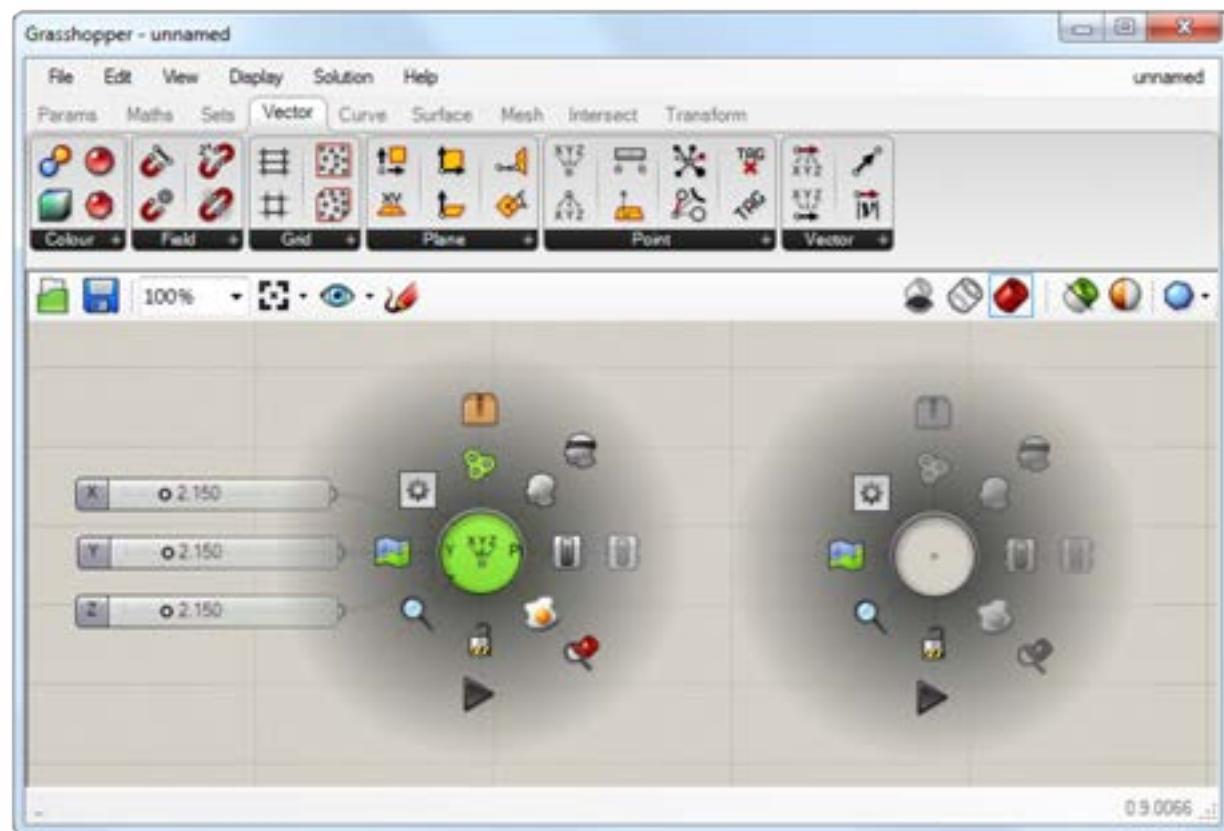
Die „Find“ Funktion kann recht hilfreich sein, wenn Du eine bestimmte Komponente auf dem Canvas lokalisieren möchtest. Rechtsklicke irgendwo auf den Canvas um die „Find“ Dialogbox zu starten.



Ein kleiner Pfeil wird zusätzlich neben den Listeneinträgen angezeigt, der auf die entsprechenden Komponenten auf dem Canvas zeigt. Versuche die „Find“ Dialogbox auf dem Canvas zu bewegen und beobachte die Rotation der Pfeile um den Komponenten zu folgen, auf welche sie zeigen. Klicke auf das gewünschte Suchergebnis um die Komponente (auf dem Canvas) neben der „Find“ Dialogbox darzustellen.

### 1.1.2.10. NUTZUNG DES RADIALMENÜS

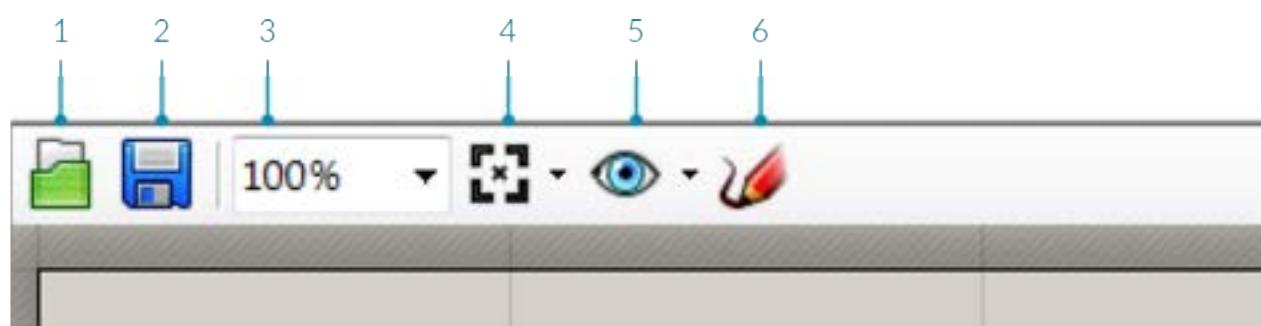
Sobald Du kompetenter im Umgang mit der Grasshopper Oberfläche bist, wirst Du einen bestimmten Arbeitsablauf beschleunigen. Tastaturkürzel sind ein Weg, um dies zu erreichen; jedoch gibt es auch eine andere Funktion, die Dir helfen kann schnellen Zugang zu einer Reihe von nützlichen Werkzeugen zu erlangen – Das Radialmenü der Benutzeroberfläche. Du kannst Das Radialmenü öffnen, indem Du die Leertaste drückst (während sich Deine Maus über dem Canvas oder einer Komponente befindet) oder indem Du die mittlere Maustaste drückst. Das Radialmenü wird verschiedene Werkzeuge bereitstellen, abhängig davon, ob Du das Menü direkt oberhalb einer Komponente oder irgendwo über dem Canvas aufrufst. In dem Bild unterhalb kannst du sehen, dass das Radialmenü im Vergleich zum Aufruf oberhalb des leeren Canvas mehr Funktionen hat, wenn Du es oberhalb einer ausgewählten Komponente aufrufst. Dieses Menü kann die Geschwindigkeit in der Du Grasshopper Dokumente erstellt dramatisch erhöhen.



Das Radialmenü der Benutzeroberfläche ermöglicht es Dir häufig genutzte Menüelemente schnell zu erreichen.

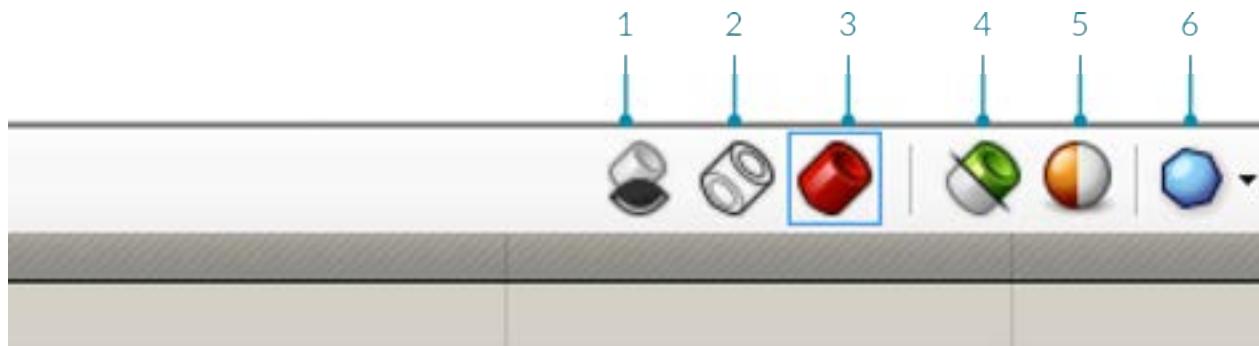
### 1.1.2.11. DIE CANVASWERKZEUGLEISTE

Die Canvaswerkzeugleiste ermöglicht schnellen Zugriff auf eine Anzahl häufig genutzter Grasshopper Funktionen. Alle Werkzeuge sind auch durch die Menüleiste zugänglich und Du kannst die Werkzeugleiste verstecken, wenn Du willst. Die Werkzeugleiste kann im „View“ Reiter der Menüleiste wieder eingeschaltet werden.

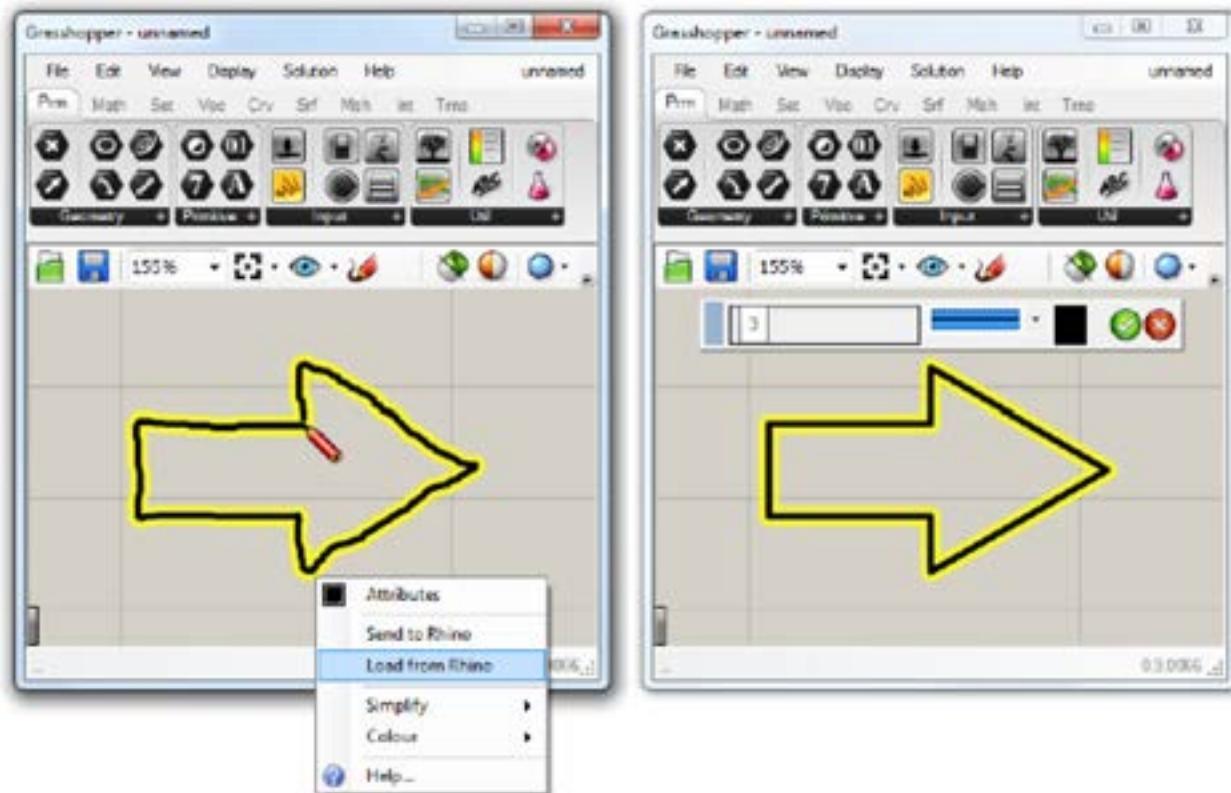


1. **Open File:** Schnellzugriff um eine Datei zu öffnen.
2. **Save File:** Schnellzugriff um eine Datei zu speichern.
3. **Zoom Defaults:** Die Standardeinstellungen für den Zoom, die es Dir erlauben in bestimmten Intervallen herein- und herauszuzoomen.
4. **Zoom Extents:** Zoom auf den gesamten Umfang der Definition. Klicke auf den Pfeil neben dem „Zoom Extends“ Symbol um eines der Untermenüelemente auszuwählen um eine bestimmte Region Deiner Definition anzusehen.
5. **Named Views:** Diese Funktion eröffnet ein Menü, das es ermöglicht jeden Ansichtsbereich Deiner Definition zu speichern und wiederherzustellen.

6. **Das Skizzenwerkzeug:** Das Skizzenwerkzeug arbeitet ähnlich wie das Bleistiftwerkzeug in Adobe Photoshop mit ein paar zusätzlich hinzugefügten Funktionen.

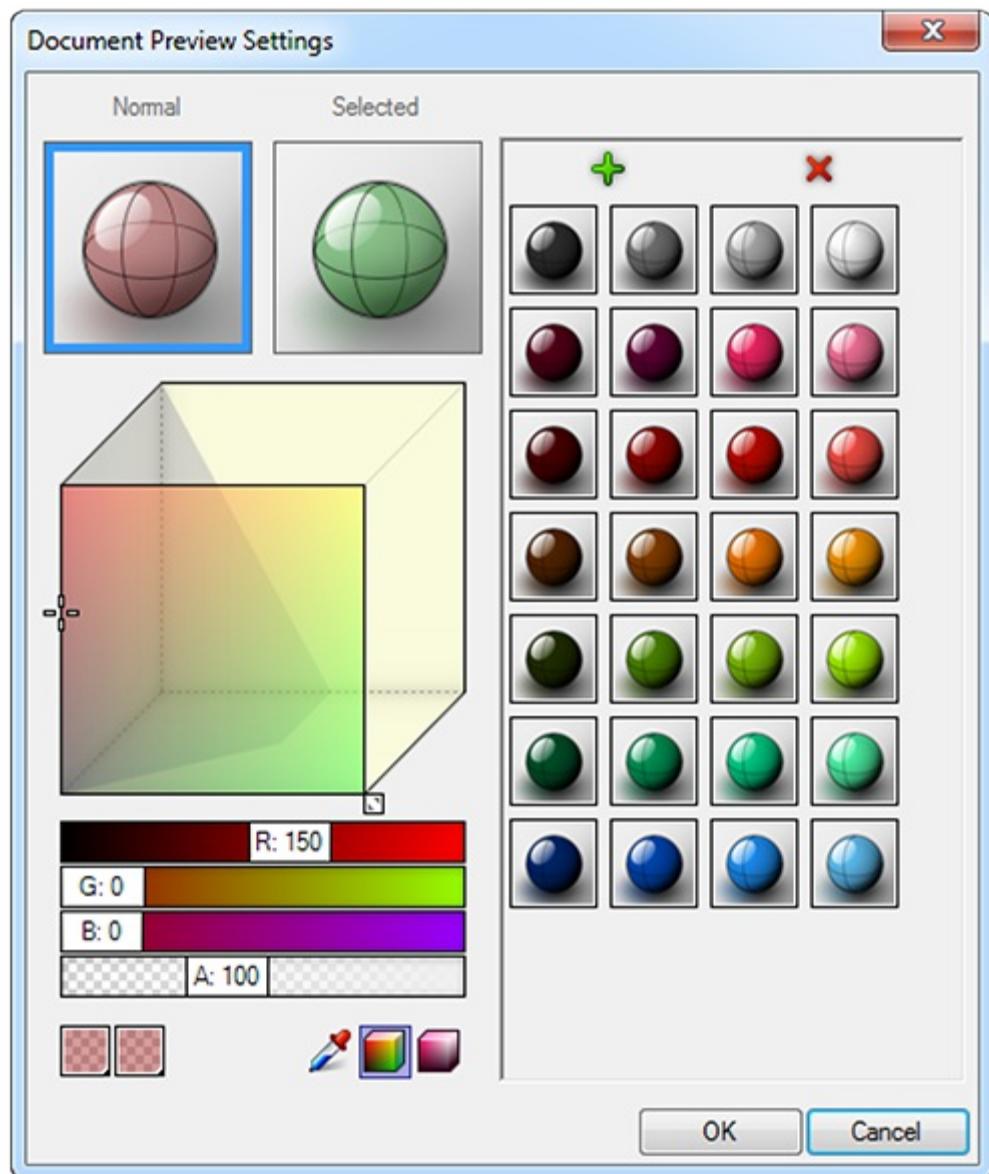


1. **Vorschauinstellungen:** Wenn eine Grasshopper Komponente Geometrien erzeugt, wird die Vorschau dieser Geometrie im Ansichtsfenster standardmäßig dargestellt. Du kannst die Vorschau pro Objekt ausschalten wenn Du diese rechtsklickst und die Vorschauoption deaktivierst, oder global den Vorschauzustand einstellen, indem Du diese drei Knöpfe benutzt.
2. Wire-frame Vorschau.
3. Vorschau ausschalten.
4. Teiltransparente Vorschau (Standard).
5. **Vorschau ausgewählter Objekte:** Wird dieser Knopf gedrückt, wird Grasshopper nur die Geometrien darstellen, die Teil der aktiven Auswahl sind, auch wenn diese Komponenten in ihren Objekteigenschaften die Vorschau ausgeschalten haben.
6. **Dokumentvorschau Einstellungen:** Grasshopper hat ein Standardfarbschema für ausgewählte (teiltransparent grün) und unausgewählte (teiltransparent rot) Geometrien. Es ist möglich dieses Farbschema in den Dokumentvorschau Einstellungen zu überschreiben.
7. **Vorschau Mesh Qualität:** Aus Optimierungsgründen kann die Qualität der Mesh- und Flächendarstellung kontrolliert werden mit der die Geometrien in Rhino dargestellt werden. Höhere Qualitätswerte werden zu höheren Laufzeiten für die Berechnung führen, während niedrigere Einstellungen zu einer weniger akuraten Darstellung führen werden. Es soll angemerkt werden, dass die Geometrie trotz der Einstellung eine hohe Auflösung der Darstellung in Rhino aufweisen wird, sobald sie in das Rhinodokument gebacken wird – die Einstellungen selbst haben nur einen geringen Einfluss auf die Darstellungsleistung und -qualität.



Das Skizzenwerkzeug erlaubt es die Strichstärke, Strichtype und Farbe zu ändern. Mit einem Rechtsklick auf ein ausgewähltes Skizzenobjekt kannst Du wählen ob Du Deine Linien vereinfachen willst um eine glattere Darstellung zu erreichen. Rechtsklicke auf Dein Skizzenobjekt und wähle "Load from Rhino" um eine beliebige 2d Form in die Rhinoszene zu laden. Sobald Du eine Form als Referenz ausgewählt hast und die Eingabetaste drückst, wird Deine Skizzenlinie in die referenzierte Form rekonfiguriert.

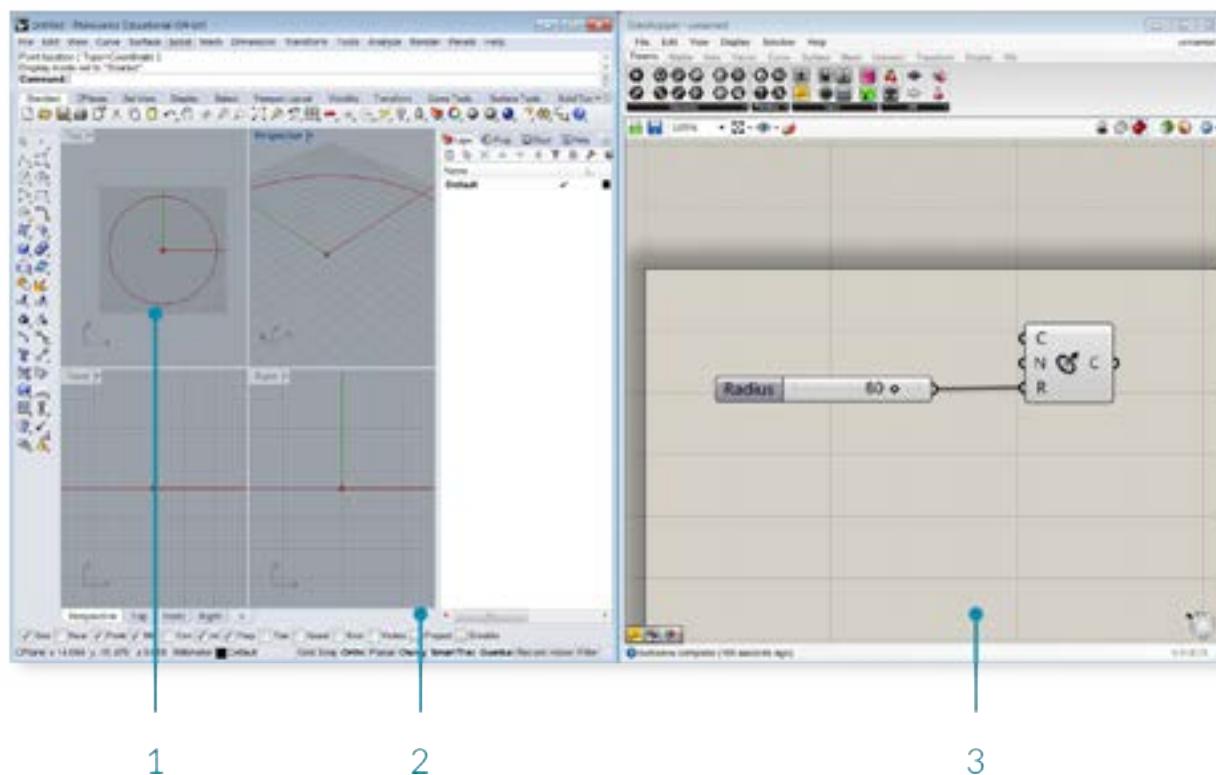
Merke: Dein Skizzenobjekt kann sich von seiner ursprünglichen Position bewegt haben, sobald Du eine Form von Rhino geladen hast. Grasshopper platziert Dein Skizzenobjekt relativ zum Ursprung des Canvas (Ecke links oben) und dem Ursprung der Welt-XY-Achse in Rhino.



Grasshopper hat ein Standardfarbschema für ausgewählte (teiltransparent grün) und unausgewählte (teiltransparent rot) Geometrien. Es ist möglich dieses Farbschema in den Dokumentvorschau Einstellungen zu überschreiben.

### ###1.1.3. MIT RHINO REDEN

Im Unterschied zu einem Rhinodokument, enthaelt eine Grasshopperdefinition keine tatsaechlichen Objekte oder Geometrien. Stattdessen repreasentiert eine Grasshopperdefinition eine Reihe von Regeln und Instruktionen, wie Rhino Aufgaben automatisieren kann.

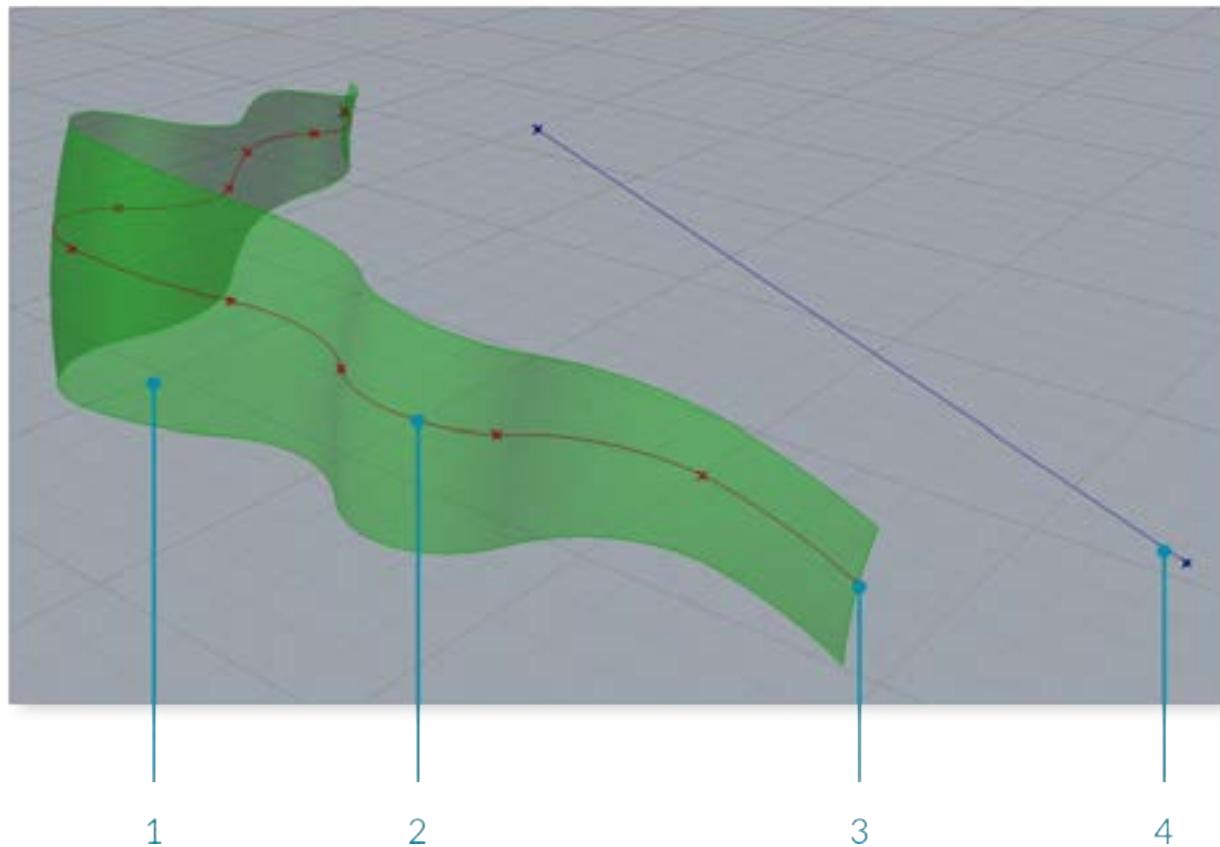


1. Grasshoppervorschau fuer Geometrien.
2. Rhino Darstellungsfenster.
3. Grasshopper Anwendungsfenster.

#### 1.1.3.1. REAKTION DER DARSTELLUNGSFENSTER

Alle Geometrie, die mit den verschiedenen Grasshopperkomponenten erzeugt wird, wird (standartmaessig) im Rhino Darstellungsfenster auftauchen. Diese Vorschau ist nur eine OpenGL Annaeherung der eigentlichen Geometrie, und als solche kann sie nicht als Geometrie im Rhino Darstellungsfenster ausgewaehlt werden (Du musst sie zuerst in die Szene backen). Du kannst die Geometrievorschau ein- bzw. ausschalten, indem Du eine Komponente mit einem Rechtsklick anklickst und den Vorschauabschalten auswaehlst. Die Geometrie im Darstellungsfenster ist farbkodiert um visuelles Feedback zu geben. Das unten gezeigt Bild umreisst das Standartfarbschema.

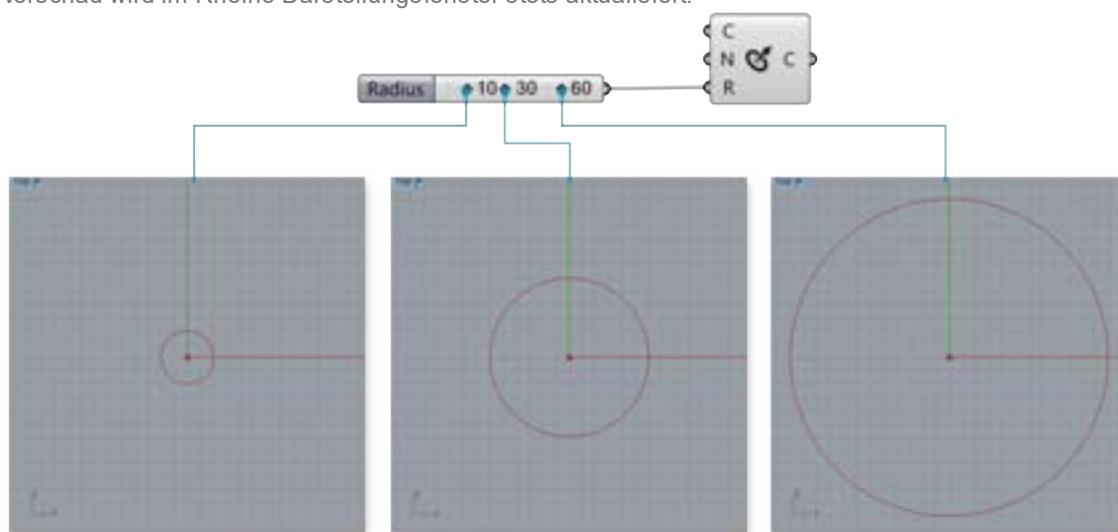
Merk: Dies ist ein Standardfarbschema, welches mit den Dokumentenvorschauinstellungen in der Canvaswerkzeugeiste angepasst werden kann.



1. Gruene Geometrien im Darstellungsfenster gehoert zur gerade ausgewaehlten Komponente.
2. Rote Geometrien im Darstellungsfenster gehoeren zu einer Komponente, die gerade nicht ausgewaehlt ist.
3. Punktgeometrien sind als Kreuz dargestellt, um sie von den rechteckig dargestellten Rhino Punktobjekten unterscheiden zu koennen.
4. Blaue Rueckmeldungen bedeuten, dass Du gerade eine Auswahl im Rhino Darstellungsfenster ausfuehrst.

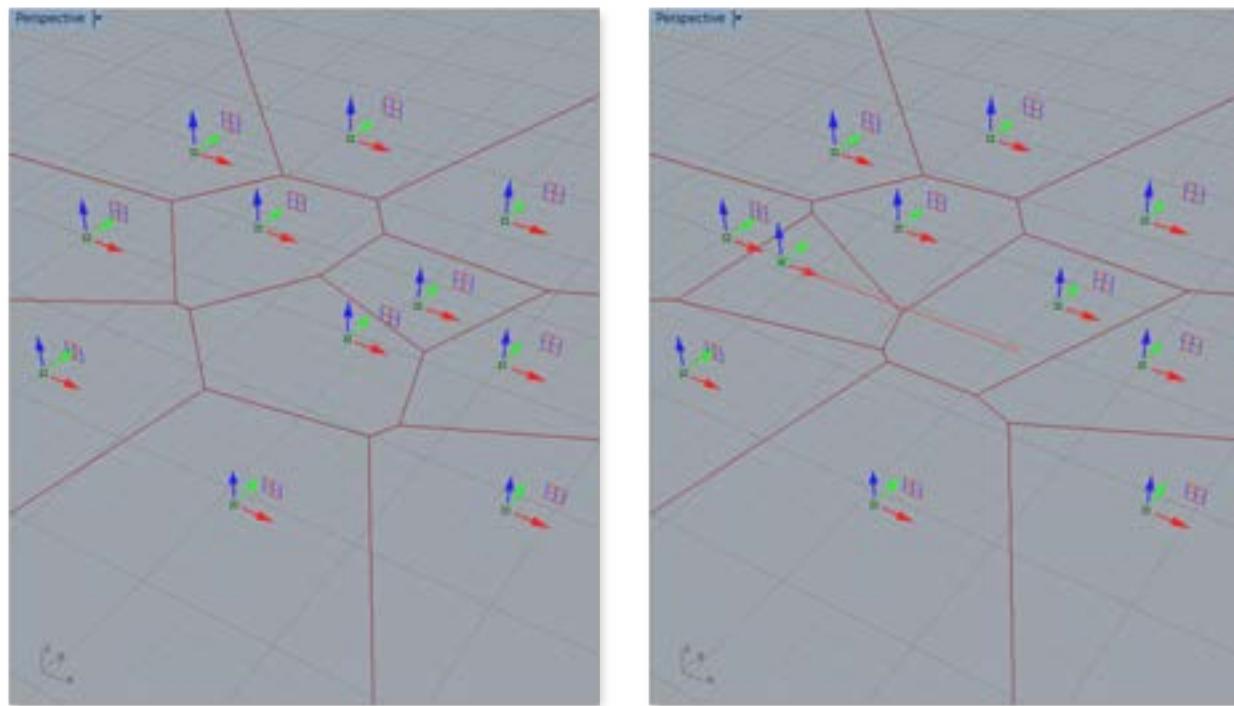
### 1.1.3.2. LEBHAFTE KABEL

Grasshopper ist ein dynamisches Umfeld. Aenderungen, die gemacht werden sind live und ihre Vorschau wird im Rhinoceros-Darstellungsfenster stets aktualisiert.



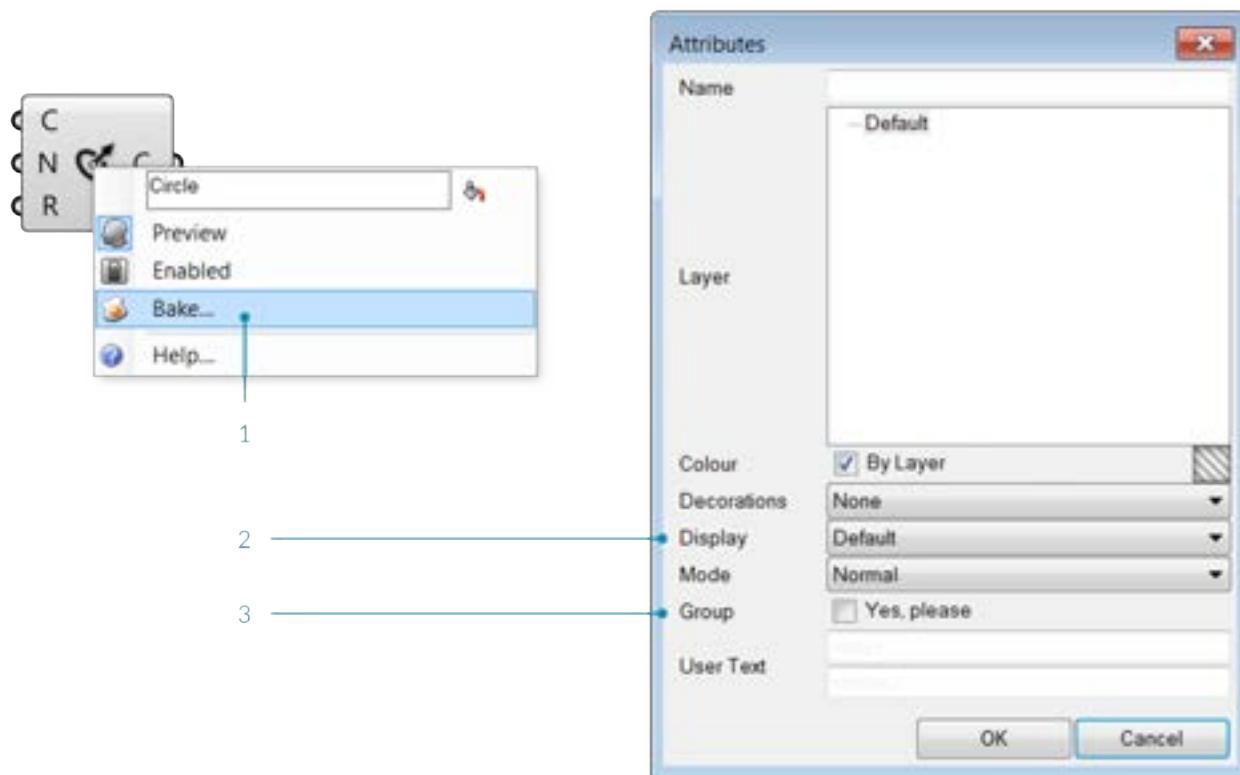
### 1.1.3.3. GUMBALL WIDGET

Wenn eine Geometrie in einen Grasshopper Parameter internalisiert gespeichert wird, erlaubt der Gumball Dir mit der Geomerie im Rhino Darstellungsfenster zu interagieren. Diese Interaktion ist live und Aktualisierungen werden durchgefuehrt, waehrend Du mit dem Gumball arbeitest. Im Gegensatz werden direkt von Rhino referenzierte Geometrien weiter im Rhinodokument existieren und Aktualisierungen werden erst nach dem Aufteten von Aenderungen ausgefuehrt (anstatt waehrend dessen).



### 1.1.3.4. BACKEN VON GEOMETRIEN

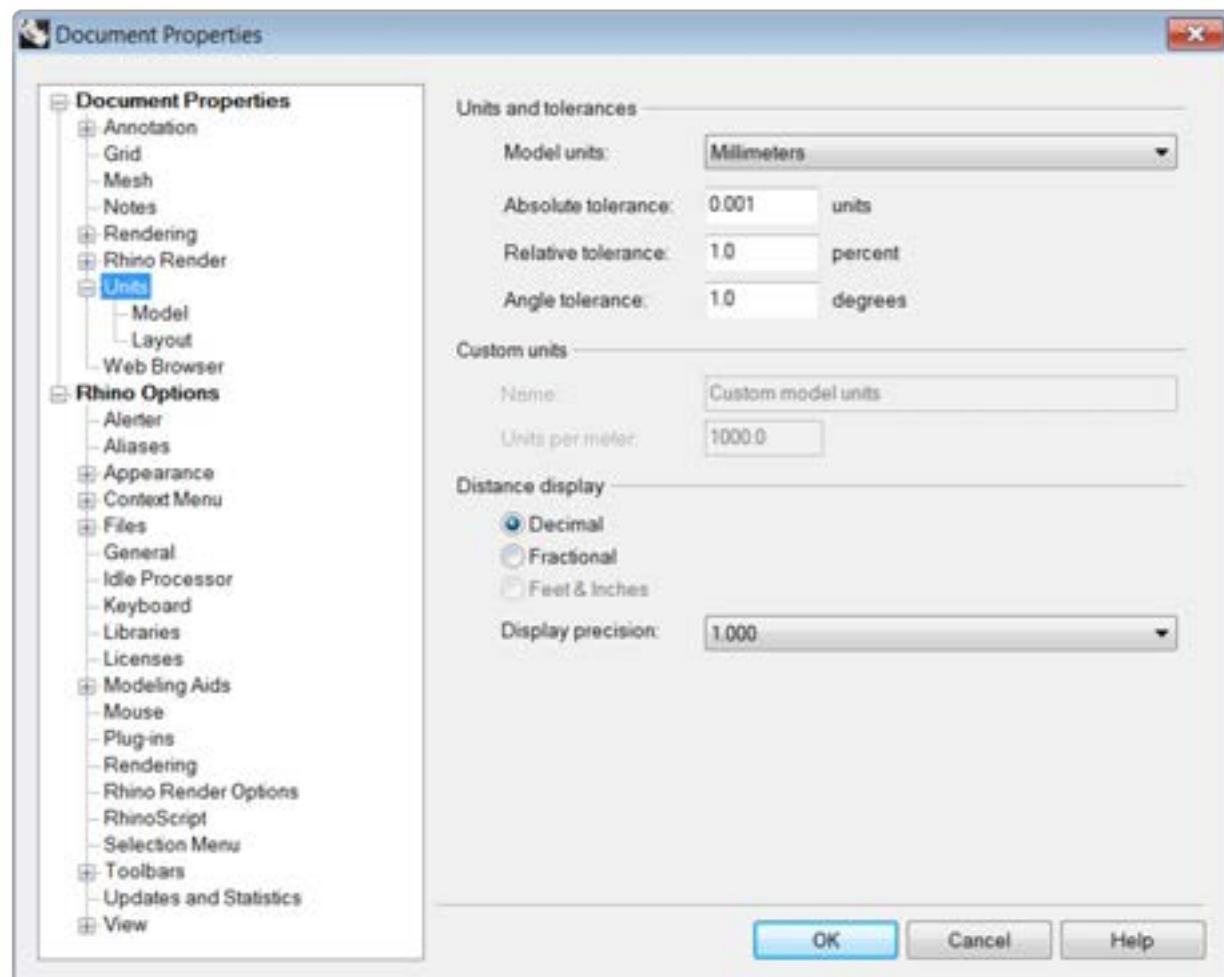
Um mit Geometrien in Rhino zu bearbeiten (auswaehlen, editieren, transformieren, etc.), die in Grasshopper erstellt wurden, musst Du sie "backen". Backen instanziert neue Geometrien im Rhinodokument, basierend auf dem momentanen Zustand des Grasshoppergraphen. Sie wird im Weiteren nicht mehr die weitere Aenderungen der Definition ansprechbar sein.



1. Backe durch Rechtsklick auf eine Komponente und Auswahl der Funktion Bake.
2. Ein Dialog wird auftauchen, der es Dir erlaubt auszuwaehlen, auf welcher Rhinobene die Geometrie eingefuegt wird.
3. Gruppieren von gebackener Geometrien ist ein komfortabler Weg um instantiierte Geometrien zu verwalten, besonders wenn Du viele Objekte in Grasshopper erzeugst.

### 1.1.3.5. EINHEITEN & TOLERANZEN

Grasshopper uebernimmt Einheiten und Toleranzen von Rhino. Um die Einheiten zu aendern, gebe `_Document Properties` in die Rhino Befehlszeile ein, wodurch Du Zugang zum Menu fuer Dokumenteneigenschaften erhaeltst. Waele Einheiten um Einheiten und Toleranzen zu aendern.



Aendere Einheiten und Toleranzen im Rhinomenu fuer Dokumenteneigenschaften.

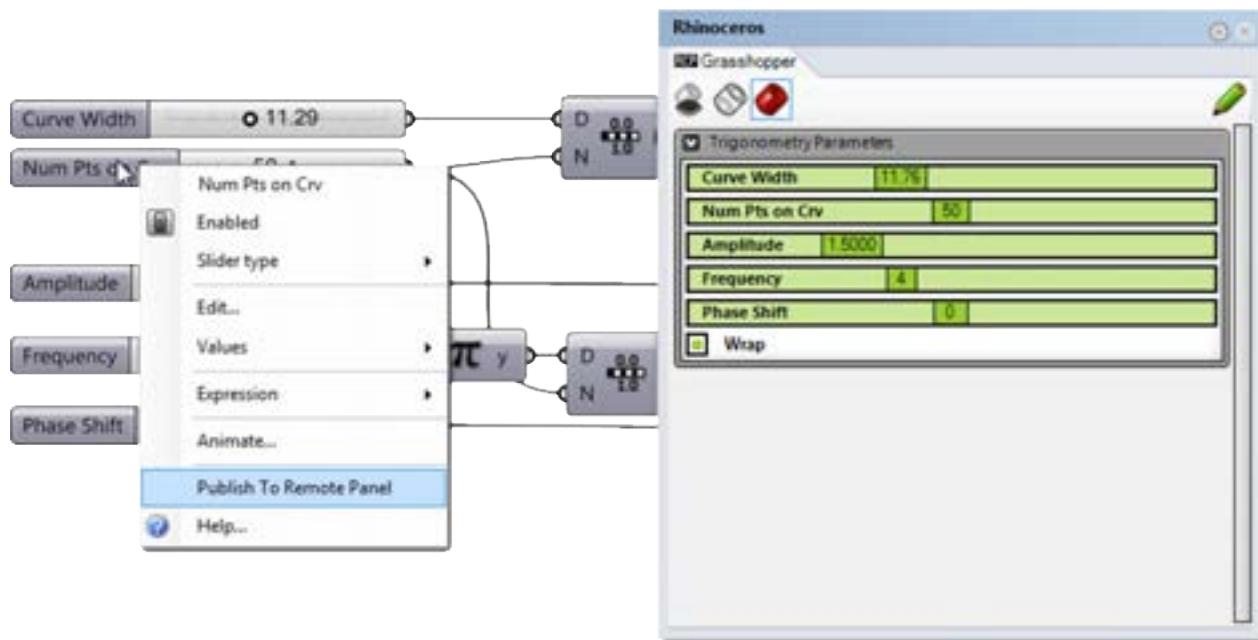
### 1.1.3.6. FERNBEDIENUNGSPANEEL

Wenn Du einmal angefangen hast, ist Grasshopper ein unglaublich maechtiges und flexibles Werkzeug, dass er Dir erlaubt verschiedene Iterationen waehrend des Entwurfsprozesses mit der graphischen Oberflaeche zu erkunden. Jedoch hast Du vielleicht schon gemerkt, dass wenn Du an einem einzelnen Bildschirm arbeitest der Grasshoppereditor einen grossen Teil der Bildschirmoberflaeche einnimmt. Abgesehen von konstantem ein- und auszoomen und verschieben der Fenster auf dem Bildschirm, gibt es noch keine elegante Loesung des Problems. Das heist... bis zur Veroeffentlichung des Fernbedienungspaneels!

Das Remote Control Panel (RCP) stellt eine minimale Oberflaeche zur Kontrolle Deiner Definition bereit, ohne unnoetig viel Platz auf dem Bildschirm einzunehmen. Das RCP kann dargestellt werden, indem Du dem Schalter im Ansichtsmenu der Hauptmenuleiste anklickst. Standardmaessig ist das RCP blank - das bedeutet nicht, dass es keine Informationen ueber Dein aktuelle Grasshopperdokument enthaelt. Um das RCP mit Bentzeroberflaechen(UI)-Elementen wie Schiebereglern, Schaltern und Knoepfen zu belegen, rechtsklicke einfach au das Element und waehle Publish to Remote Panel. Dies wird eine neue Gruppe erstellen und ein synchronisiertes UI-Element im RCP erzeugen. Eine Veraenderung der Werte des UI wird auch die werte in Deinem Graphen aktualisieren und die Geometrien in den Ansichtsfenstern modifizieren, die von diesen Parametern abhaengen. Du kannst mehrere Elemente im RCP veroeffentlichen und die gesamte Oberflaeche belegen. Diese Funktion kannst Du nutzen, um Deine Datei zu kontrollieren, ohne dass Du die Verstrickungen Deines visuellen Graphen in der Nutzeroberflaeche oberhalb Deiner Rhinoansichtsfenster zeigen zu muessen.

Merke: Das RCP wird die Namen der UI Elemente uebernehmen und deren Namen als Label nutzen. Es ist

gute Praxis die Schieberegler und Schalter mit entsprechend verstaendlichen und bedeutungsvollen namen zu aktualisieren. Dies wird die Nutzbarkeit Deines RCP unwahrscheinlich erleichtern.



Um ein UI Element (z.B. Schieberegler, Schalter, Knopf) im Remote Control Panel anzuzeigen, muessen wir es erst dort publizieren.

Die RCP UI kann auch personalisiert werden – um Dir zu erlauben, festzulegen an welcher Stelle Objekte in der Oberflaeche erscheinen, sowie die Namen und Farben verschiedener Gruppen einzustellen. Um das Layout des RCP zu modifizieren musst Du zuerst vom "Working Mode" (die Standard RCP-Ansicht) in den "Edit Mode" umschalten. Du kannst diesen Bearbeitungsmodus erreichen, indem Du auf den gruenen Bleistift in der oberen rechten Ecke des RCP klicks. Sobald Du im Bearbeitungsmodus bist, kannst Du neue UI-Gruppen erstellen, Elemente innerhalb der Gruppen neu ausrichten, Label hinzufuegen, Farben aendern und vieles mehr. Um ein UI-Element zu loeschen, ziehe es einfach ueber die Begrenzung des RCP hinaus. Du kannst die individuellen Werte der parameter nicht aendern, solange Du Dich im Bearbeitungsmodus befindest. Statt dessen, wirst Du wieder auf den gruenen Bleistift klicken muessen, um in den urspruenglichen Arbeitsmodus zurueckzukehren.

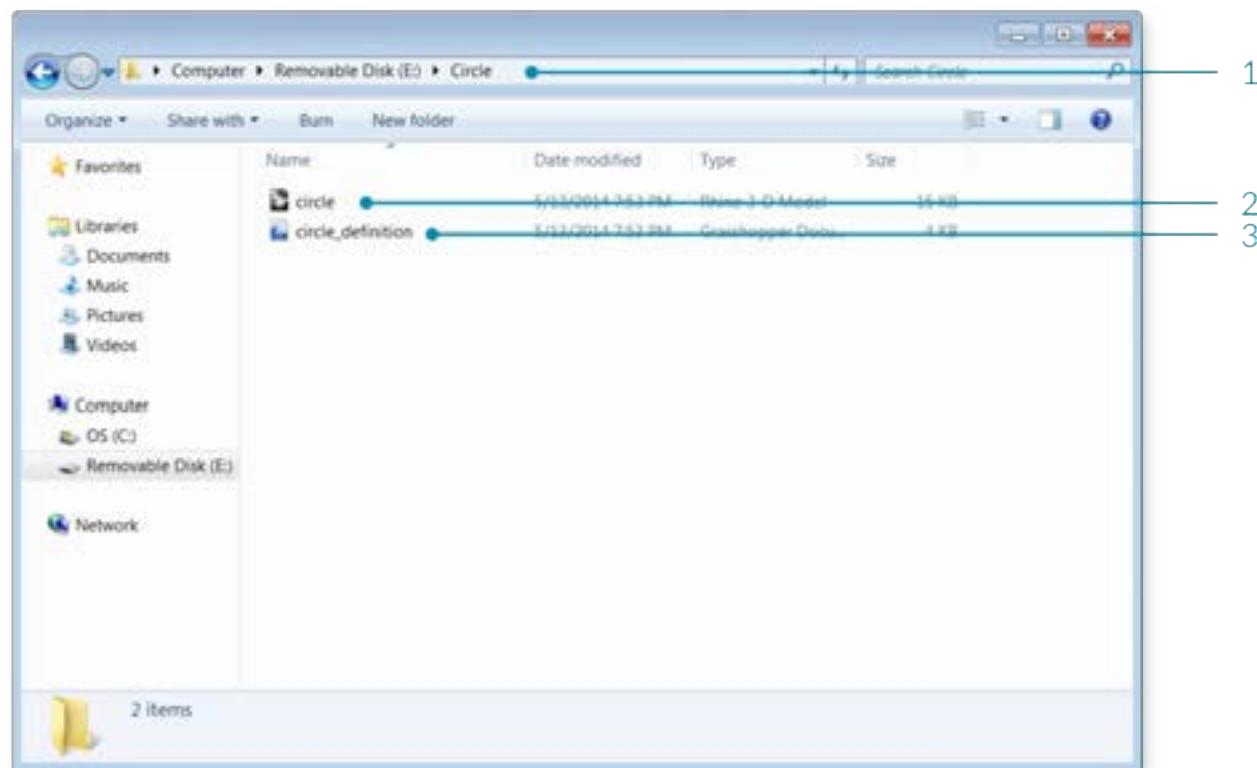
*Das "Remote Control Panel" hat zwei verschiedene Modi: Bearbeitungsmodus (links), der es Dir ermoeglicht das Aussehen und die Handhabung des RCP anzupassen, und den Arbeitsmodus (rechts), in dem Du die eigentlichen Werte des UI bearbeiten kannst.*



Das "Remote Control Panel" im Bearbeitungsmodus hat einen orangenen Hintergrund.

### 1.1.3.7. DATEI MANAGEMENT

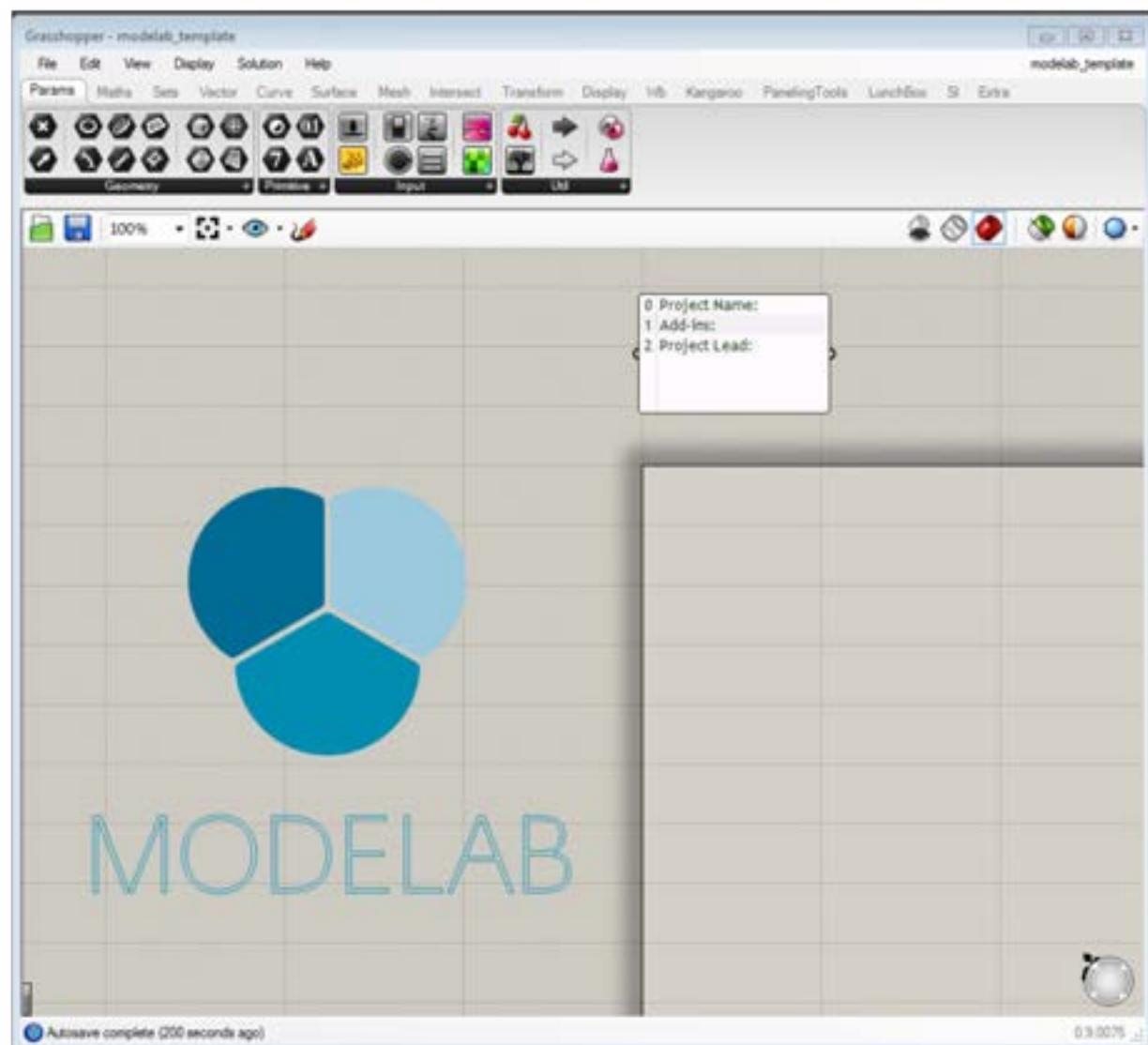
Wenn Deine Grasshopperdatei Geometrien aus Rhino referenziert, musst Du die selbe Datei in Rhino oeffnen, damit die Definition funktioniert. Behalte Deine Datein in einer organisierten Struktur, indem Du die Grasshopper- und Rhinodateien im selben Verzeichnis speicherst und Ihnen Namen gibst, die in Beziehung zueinander stehen.



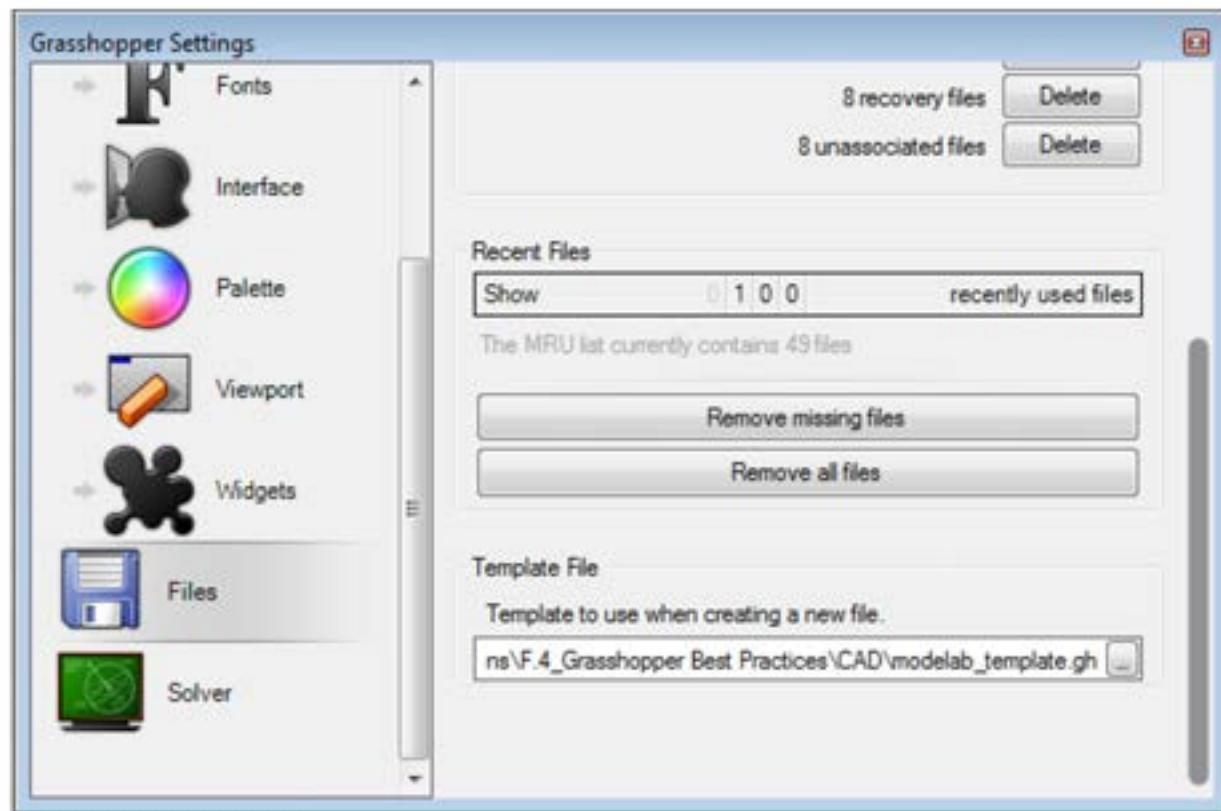
1. Projektverzeichnis.
2. Rhinodatei.
3. Grasshopperdatei.

### 1.1.3.8. TEMPLATES

Die Erstellung und Spezifizierungen einer Templatedatei mit Deinen Grasshopperpräferenzen ist ein angenehmer Weg um neue Grasshopperdefinitionen auf Grundlage Deiner Voreinstellungen zu erzeugen. Das Template kann Grasshopperkomponenten, -paneele und -skizzen enthalten, um Deine Dokumentation zu erleichtern.



Erzeuge eine Templatatedatei und speichere sie.



1. Unter File/Preferences, lade die Datei, die Du gerade als Template erstellt hast. Dein Template wird nun jedes Mal genutzt werden, wenn Du eine neue Datei erstellst.

## 1.2. ANATOMIE EINER GRASSHOPPER DEFINITION

Grasshopper ermöglicht es Dir visuelle Programme zu erstellen, die Definitionen genannt werden. Diese Definitionen bestehen aus Knoten, die mit Kabeln verbunden werden. Die folgenden Kapitel führen Grasshopperobjekte ein und erklären, wie man mit ihnen interagiert, um Definitionen aufzubauen.



## 1.2.1. GRASSHOPPER OBJEKTYPEN

Grasshopper besteht aus zwei primaeren Typen von Benutzerobjekten: Parameter und Komponenten. Parameter speichern Daten, wobei Komponenten Aktionen ausfuehren, welche wiederum Daten erzeugen. Der einfachste Weg, um Grasshopper zu verstehen ist, sich daran zu erinnern, dass wir Daten nutzen werden, um die Eingabe von Aktionen zu definieren (was neue Daten erzeugt, die wir im weiteren Verlauf nutzen koennen).

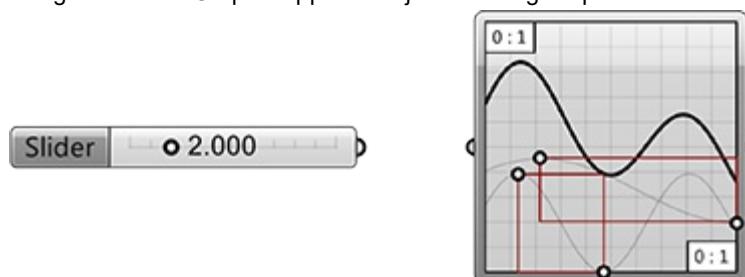
### 1.2.1.1. PARAMETER

Parameter speichern die Daten – Zahlen, Farben, Geometrien , u.a. – die wir durch den Graph unserer Definition senden. Parameter sind Containerobjekte, welche gewoehnlich als kleine rechteckige Kaestchen mit einer einzelnen Eingabe und Ausgabe angezeigt werden. Wir koennen Parameter auch an der Form des Symbols erkennen, da alle Parameterobjekte eine sechseckige Einfassung haben.

Geometrieparameter koennen Geometrien aus Rhino referenzieren oder Geometrien von anderen Komponenten aufnehmen. Punkt- und Kurvenobjekte sind beide Geometrieparameter.

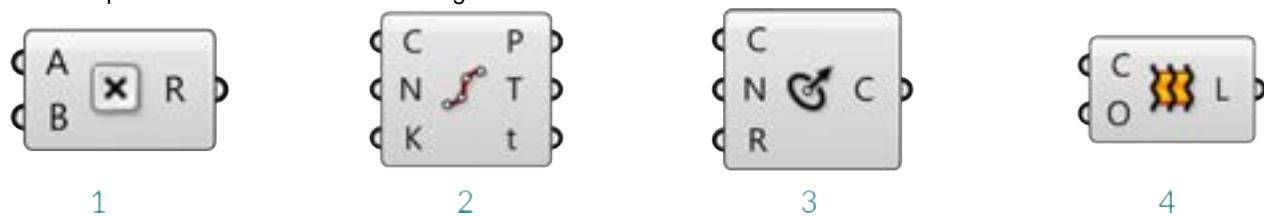


Eingabeparameter sind dynamische Schnittstellenobjekte, die es ermoeglichen direkt mit der Definition zu interagieren. Die Schieberegler und der Graphmapper sind jeweils Eingabeparameter.



### 1.2.1.2. KOMPONENTEN

Komponenten fuehren Aktionen auf Basis der Eingaben aus, welche sie erhalten. Es gibt viele verschiedene Typen von Komponenten fuer verschiedene Aufgaben.

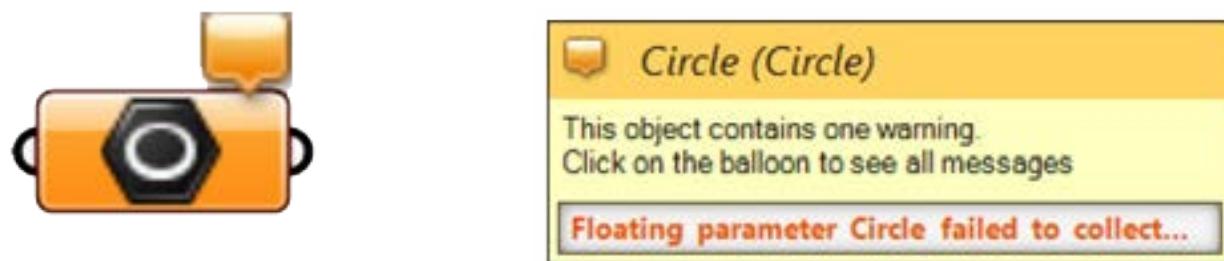


1. Die Multiplikations-Komponente ist ein Operator, der das Produkt zweier Zahlen berechnet.
2. Die Divide Curve-Komponente arbeitet mit Geometrien. Sie teilt eine Kurve in gleich lange Segmente.
1. Die Circle CNR-Komponente konstruiert eine Kreisgeometrie von den Eingaben, dem Mittelpunkt, Normalenvektor und Radius.
2. Die Loft-Komponente konstruiert eine Loftflaeche von zwei Kurven.

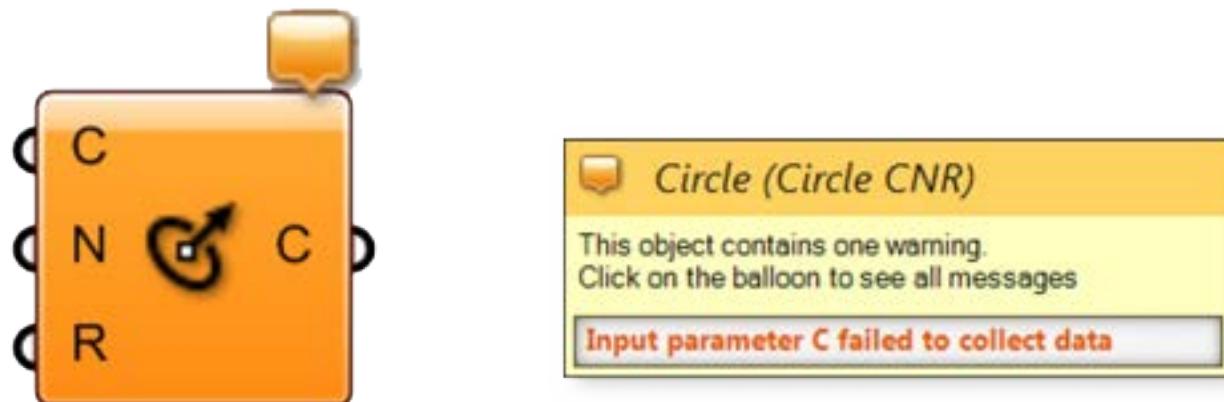
### 1.2.1.3. OBJEKTFARBEN

Wir koennen einige Informationen ueber die Objekte erhalten, wenn wir ihre Farben betrachten. Schauen wir uns die grundsaetzliche Farbcodierung von Grasshopper gemeinsam an. Ein Parameter, der keine Warnungen oder

Fehlermeldungen enthaelt, wird in hellgrau angezeigt. Diese Objektfarbe zeigt an, dass dieser Parameter einwandfrei arbeitet. Ein Parameter, der Warnungen enthaelt, wird als orangenes Kaestchen angezeigt. Objekte, die keine Daten als Eingabe erhalten, sind verdaechtig, weil sie nicht zur Grasshopperdefinition beitragen. Deshalb werden alle Parameter, wenn sie frisch hinzugefuegt werden, erst einmal orange dargestellt, um anzuzeigen, dass sie keine Daten enthalten und keinen Einfluss auf das Ergebnis der Definition haben. Standardmaessig erhalten orange angezeigte Parameter und Komponenten einen kleinen Ballon an der rechten, oberen Ecke des Objekts. Sobald du mit der Maus ueber den Ballon faehrst, wird er Informationen darueber anzeigen, warum die Warnung aufgetreten ist. Sobald ein Parameter Daten enthaelt oder definiert, wird er in grau angezeigt und der Ballon verschwindet.



Eine Komponente ist immer ein staerker eingebundenes Objekt, weshalb wir seine Eingabe und Ausgabe bewusst verstehen und anschliessend koordinieren muessen. Wie bei den Parametern wird eine Komponente, die Warnungen enthaelt, orange dargestellt. Behalte im Hinterkopf, dass Warnungen nicht gezwungenermassen schlecht sind, sondern dass Grasshopper dich mit ihnen lediglich auf ein potenzielles Problem in der Definition hinweisen moechte.



Eine Komponente, die weder Warnungen noch Fehlermeldungen enthaelt, wird in hellgrau angezeigt.

Eine Komponente, deren Vorschau deaktiviert wurde, wird in einem etwas dunkleren Grauton dargestellt. Es gibt zwei Moeglichkeiten, um die Vorschau einer Komponente zu deaktivieren. Zuerst einmal kannst du einfach einen Rechtsklick auf der Komponente ausfuehren und den Vorschauschalter umschalten. Um die Vorschau verschiedener Komponenten gleichzeitig auszuschalten, musst du die gewuenschten Komponenten auswaehlen und dann den entsprechenden Schalter (Mann mit verbundenen Augen) im Dropdownmenue auswaehlen, nachdem du irgendwo auf der Leinwand einen rechten Mausklick ausgefuehrt hast.

Eine deaktivierte Komponente wird in einem stumpfen Grauton dargestellt. Um eine Komponente zu deaktivieren, kannst du einen Rechtsklick auf einer Komponente machen und den Disable-Schalter betaetigen, oder die gewuenschten Komponenten auswaehlen und nach einem Rechtsklick auf der Leinwand die Option Disable auswaehlen. Deaktivierte Komponenten senden keine Daten mehr an nachgelagerte Komponenten.

Eine ausgewaehlte Komponente wird in hellgruen angezeigt. Falls die Komponente Geometrien in der Rhinoszene

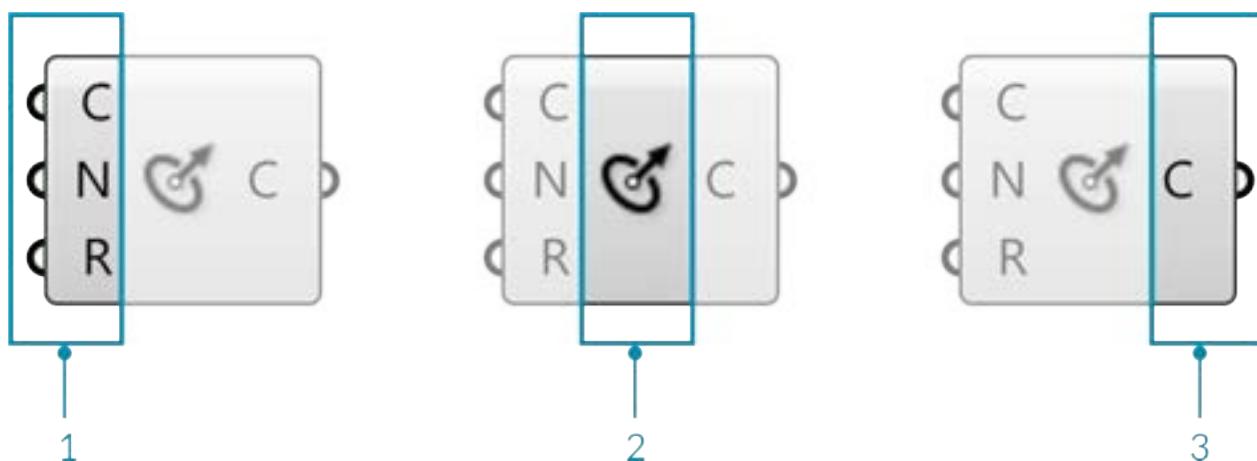
erzeugt hat, werden diese ebenfalls gruen, um dir eine optische Rueckmeldung zu geben. Eine Komponente, die mindestens eine Fehlermeldung enthaelt, wird rot dargestellt. Die Fehlermeldung kann durch die Komponente selbst oder ihre Eingabe und Ausgabe ausgelöst werden.



1. Ein Parameter ohne Warnungen und Fehlermeldungen
2. Ein Parameter mit Warnungen
3. Eine Komponente mit Warnungen
4. Eine Komponente ohne Warnungen und Fehlermeldungen
5. Eine Komponente mit deaktivierter Vorschau
6. Eine deaktivierte Komponente
7. Eine ausgewählte Komponente
8. Eine Komponente mit einer Fehlermeldung

### ### 1.2.2. GRASSHOPPER KOMPONENTEN

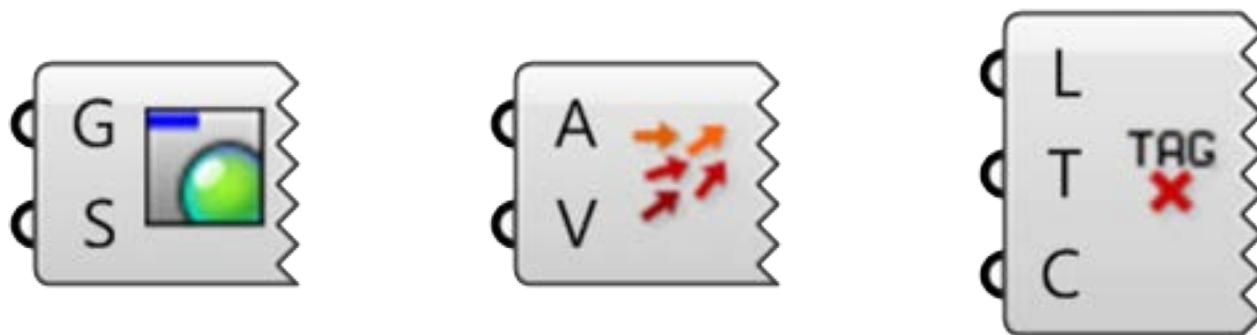
Komponenten sind die Objekte, welche Du auf der Leinwand platzierst und mit Kabeln miteinander verbindest um ein visuelles Programm zu erstellen. Komponenten repräsentieren Rhino Geometrien und verhalten sich wie mathematische Funktionen. Komponenten haben Eingabe- und Ausgabeparameter.



1. Die drei Eingabeparameter einer "Circle CNR" Komponente.
2. Das "Circle CNR" Komponentenlabel.
3. Die Ausgabeparameter der "Circle CNR" Komponente.

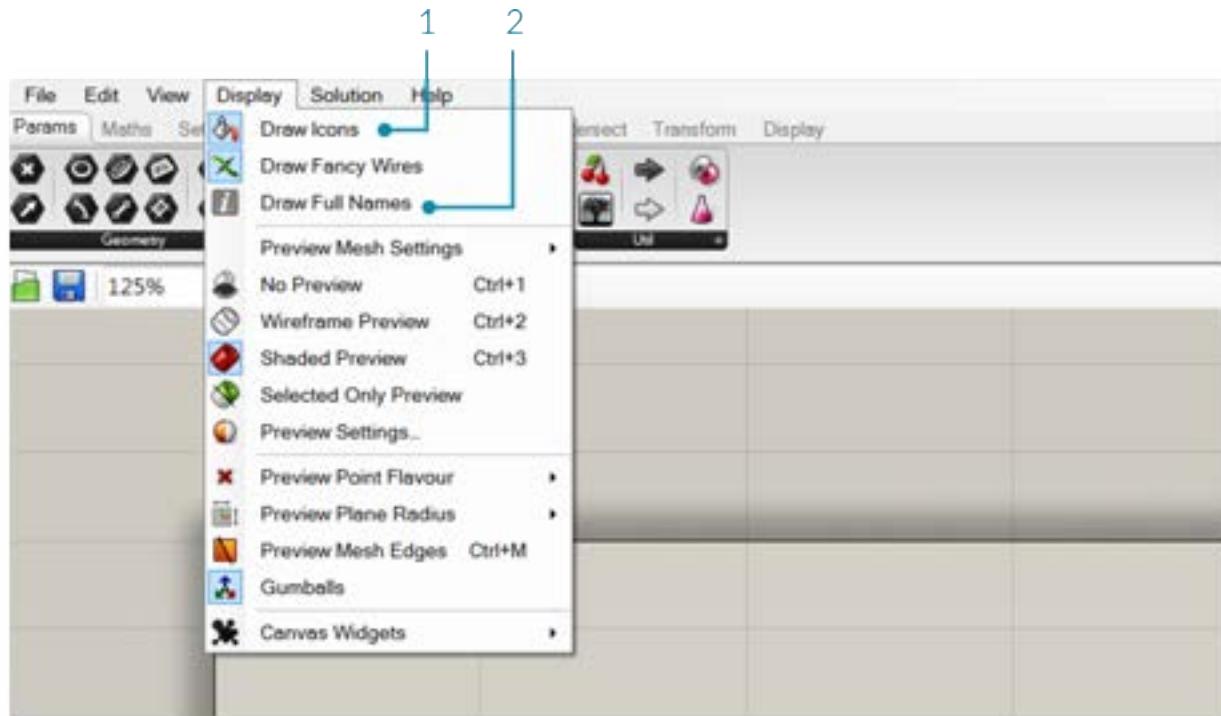
Eine Komponente benoetigt Daten, damit sie ihre Aktionen ausfuehren kann und dann normalerweise ein Ergebnis darstellt. Das ist der Grund warum die meisten Komponenten eine verschachtelte Struktur von Parametern haben, die als Eingabe- und Ausgabeparameter bezeichnet werden. Eingabeparameter sind an der linken Seite angeordnet, Ausgabeparameter an der rechten Seite.

Es gibt einige Grasshopperkomponenten, die nur Eingabe-, aber keine Ausgabeparameter haben, oder andersherum. Wenn eine Komponente keine Eingabe- oder Ausgabeparameter hat, wird es eine gezackte Kante haben.



#### 1.2.2.1. LABEL ODER SYMBOL DARSTELLUNG

Jedes Grasshopperobjekt hat ein einzigartiges Symbol. Diese Symbole werden im Zentrum des entsprechenden Objektes dargestellt und korrespondieren mit den Symbolen, die in der Komponentenpalette dargestellt werden. Objekte koennen auch mit Text beschriftet dargestellt werden. Um zwischen Symbol- und Textdarstellung umzuschalten, waehle "Draw Icons" im Ansichtsmenu. Du kannst auch "Draw Full Names" auswaehlen, um den gesamten Namen eines Objekts mitsamt seinen Eingabe- und Ausgabeparametern darzustellen.



1. Schalte zwischen Symbol- und Textdarstellung um.
2. Zeige den gesamten Namen einer Komponente mitsamt seinen Eingabe- und Ausgabeparametern an.



1



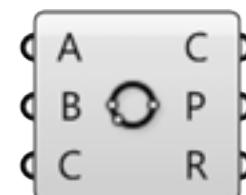
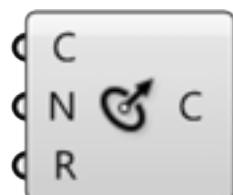
2



3

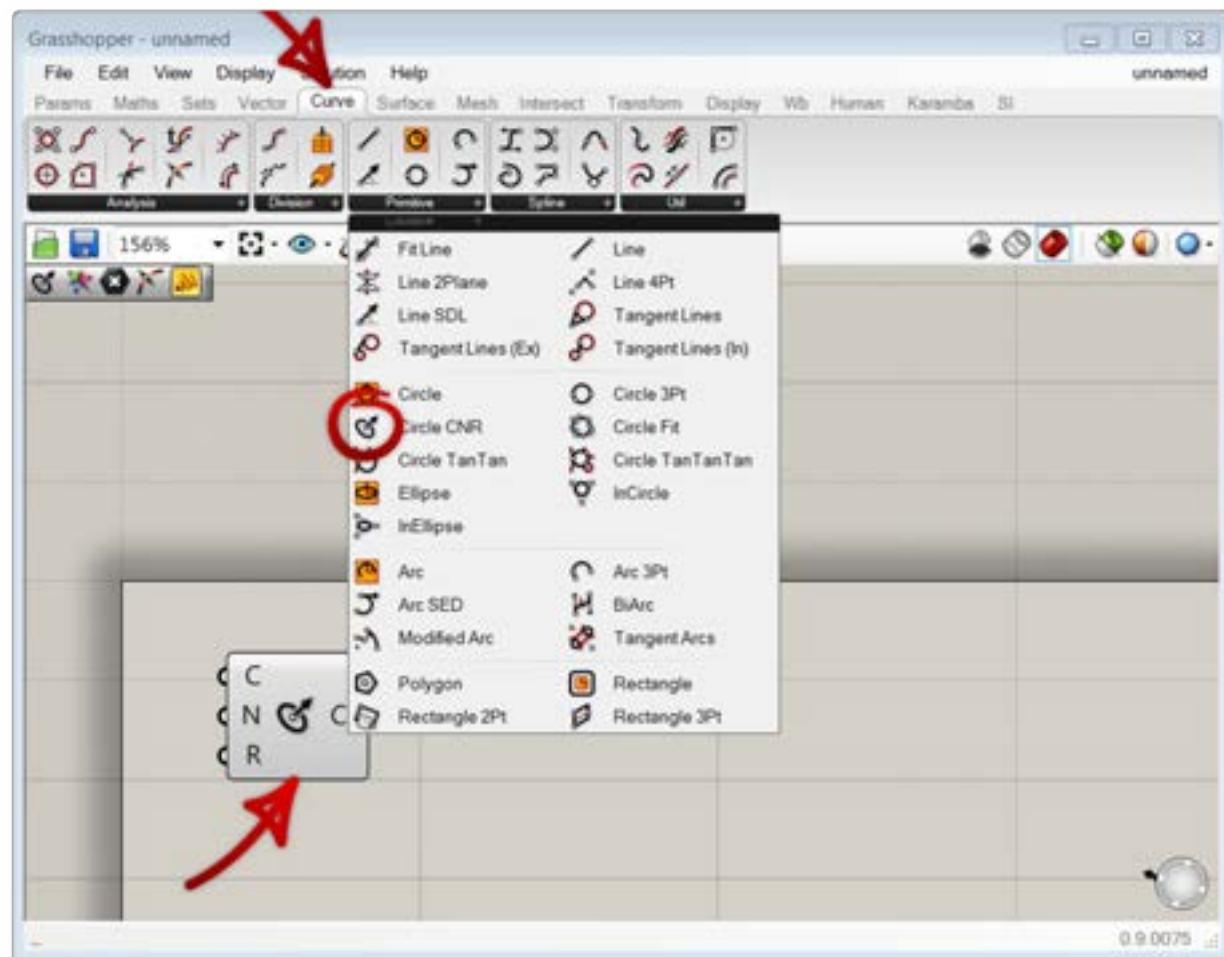
1. Die "Circle CNR" Komponente in Textdarstellung
2. Die "Circle CNR" Komponente in Symboldarstellung
3. Die "Circle CNR" Komponente mit gesamtem Namen angezeigt

Wir empfehlen die Symboldarstellung, um Dich mit den Symbolen in der Komponentenpalette vertraut zu machen. Dies wird Dir auch helfen Definitionen auf einen Blick zu verstehen. Textlabels koennen verwirrend sein, weil verschiedene Komponenten die selbe Bezeichnung teilen koennen.



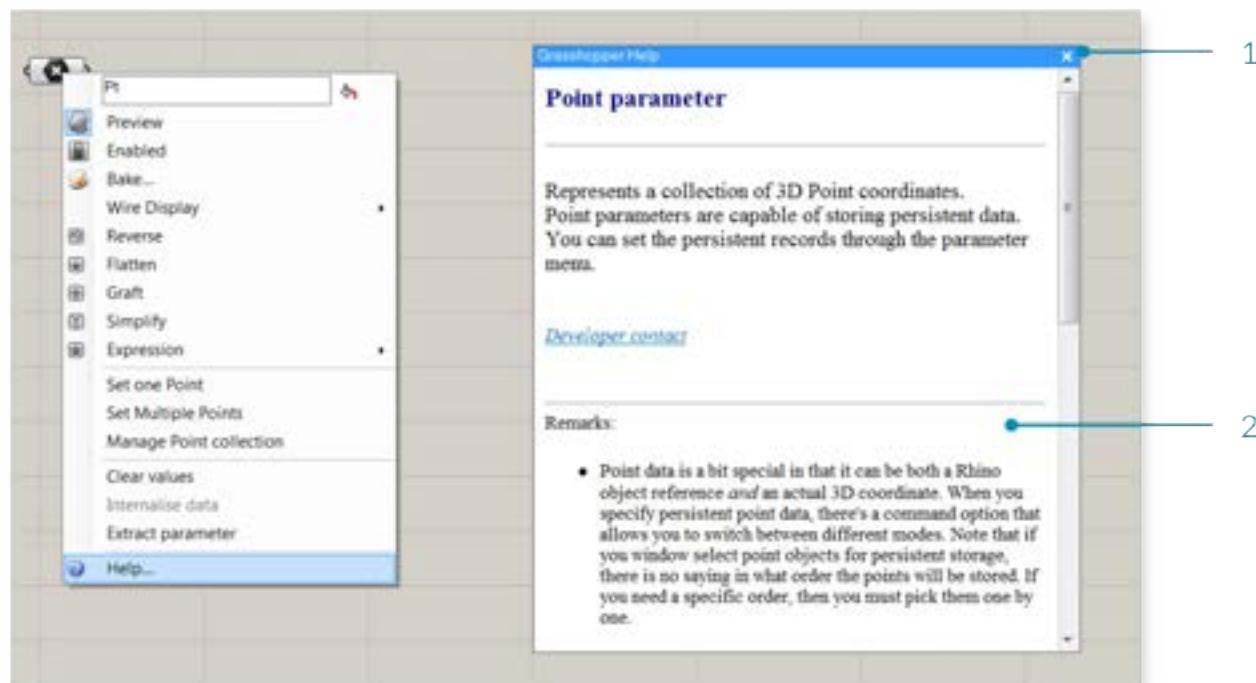
"Circle CNR" und "Circle 3pt" haben das selbe Textlabel, aber unterschiedliche Symbole.

Eine Eigenschaft, die Dir helfen kann, Dich mit der Position von Komponenten in der entsprechenden Palette vertraut zu machen, besteht darin Strg + Alt zu halten, waehrend Du auf eine bestehende Komponente auf der Leinwand klickst. Dies wird die Position der Komponente in der Palette anzeigen.



### 1.2.2.2. KOMPONENTEN HILFE

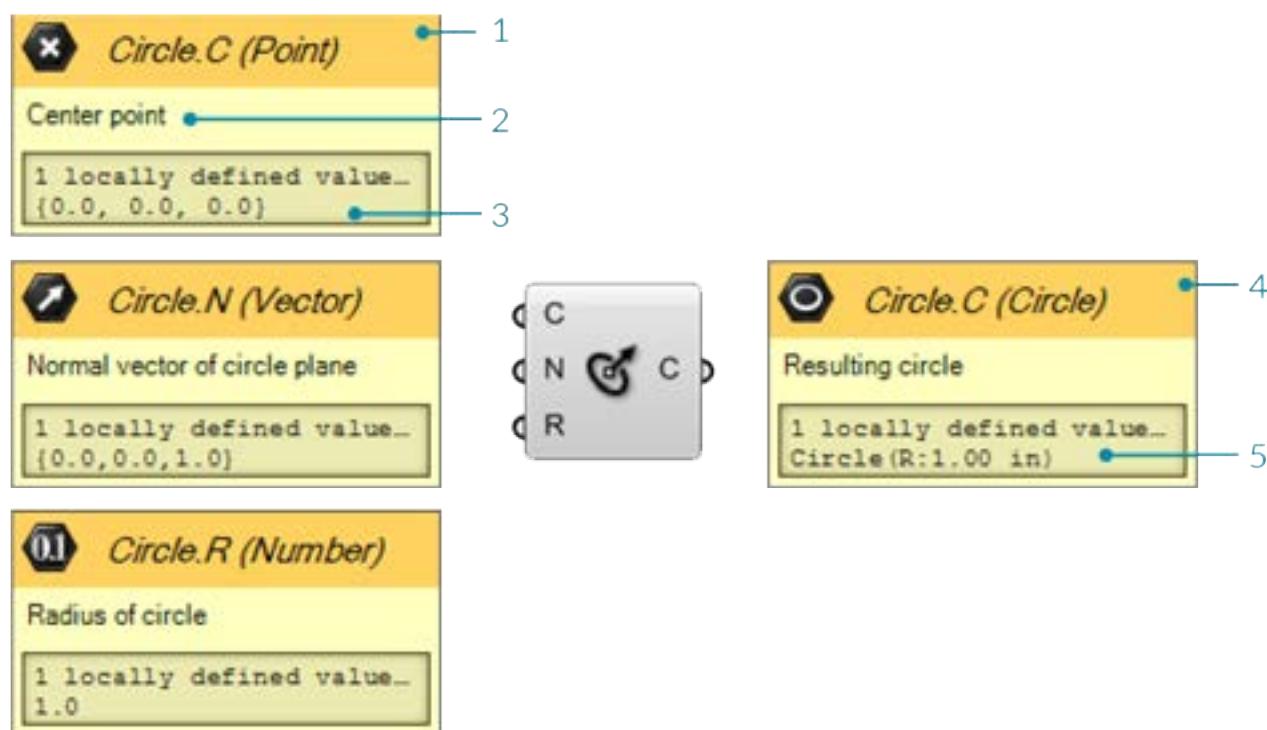
Rechtsklicke auf ein Objekt und wähle "Help" im drop-down menu um das Grasshopper Hilfefenster zu öffnen. Das Hilfefenster enthält eine detailliertere Beschreibung des Objekts, eine Liste der Eingabe- und Ausgabeparameter und einige Anmerkungen.



1. Grasshopper Hilfefenster fuer einen Punktparameter
2. Die Anmerkungen im Hilfefenster geben zusaetliche Hinweise zum Punktparameter.

### 1.2.2.3. WERKZEUGTIPS

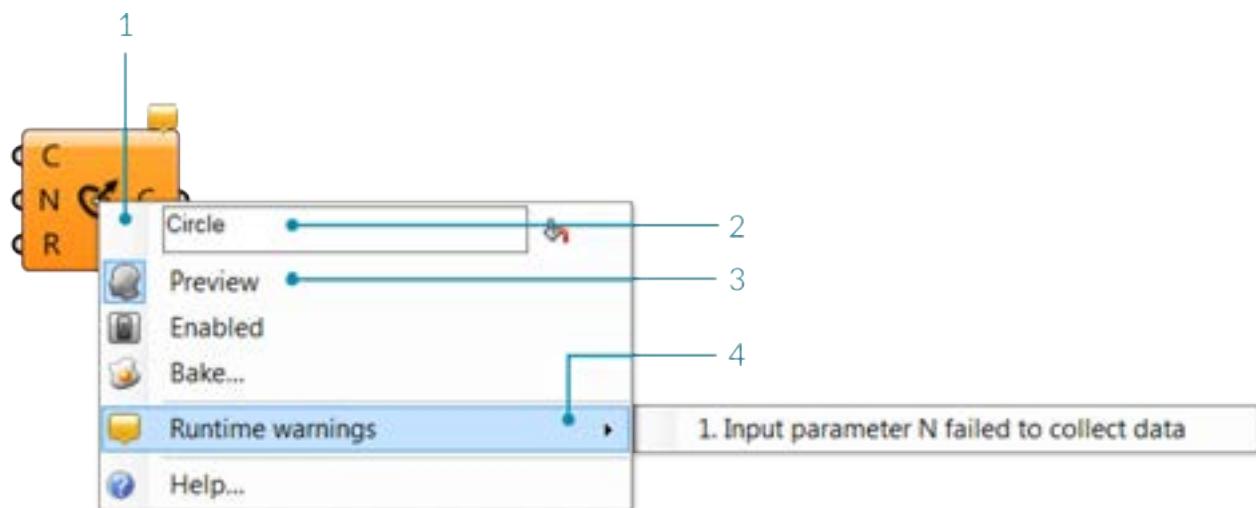
Eingabeparameter von Komponenten erwarten bestimmte Datentypen zu erhalten. Beispielsweise kann eine Komponente anzeigen, dass Du einen Punkt oder eine Ebene als Eingabe mit ihr verbinden sollst. Wenn Du mit Deiner Maus ueber die verschiedenen Teile der Komponente fahrest, wirst Du verschiedene Werkzeugtips sehen, die einen bestimmten Typ anzeigen, den das Unterobjekt, das sich gerade unter der Mauszeige befindet, erfordert. Werkzeugtips sind relativ informativ, da sie Dir von die erforderlichen Typen und Daten der bestimmten Parameter berichten.



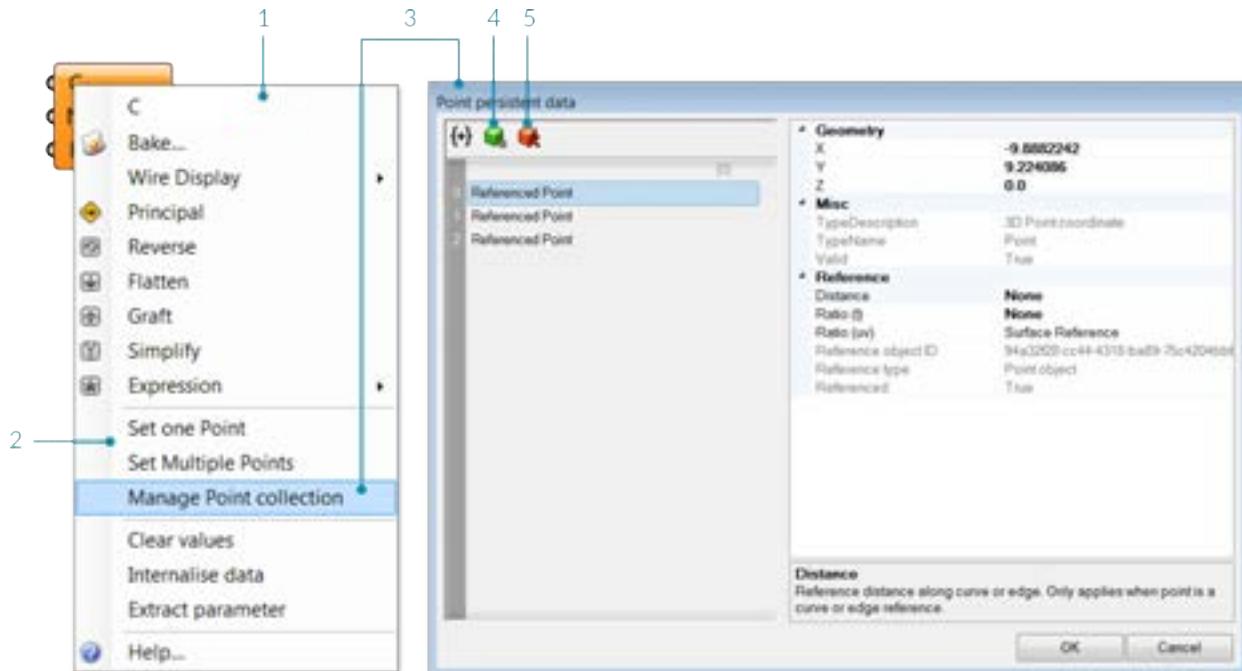
1. Die Überschrift des Werkzeugtips zeigt Dir das Symbol für den Eingabetypen, den Namen der Komponente, das Label für die Eignabe und die Eingabetype nochmals im Textformat an.
2. Die sprachliche Beschreibung des Eingabeparameters für die Komponente.
3. Alle Werte, die für den Eingabeparameter definiert wurden - entweder lokal oder von einem verbundenen Kabel.
4. Die Überschrift des Ausgabewerkzeugtips stellt die selben Details bereit, wie auf der Eingabeseite, jedoch für den entsprechenden Ausgabeparameter.
5. Das Ergebnis der Aktion der Komponente.

#### 1.2.2.4. KONTEXT POPUPMENUS

Alle Objekte auf der Leinwand haben ihre eigenen Kontextmenüs, die ihre Einstellungen und Details beinhalten. Du kannst diese Kontextmenüs öffnen, indem Du im Zentrum der Komponente rechtsklickst. Eingabe- und Ausgabe haben jeweils ihre eigenen Kontextmenüs, welche mit einem entsprechenden Rechts-Klick zugehörig sind.



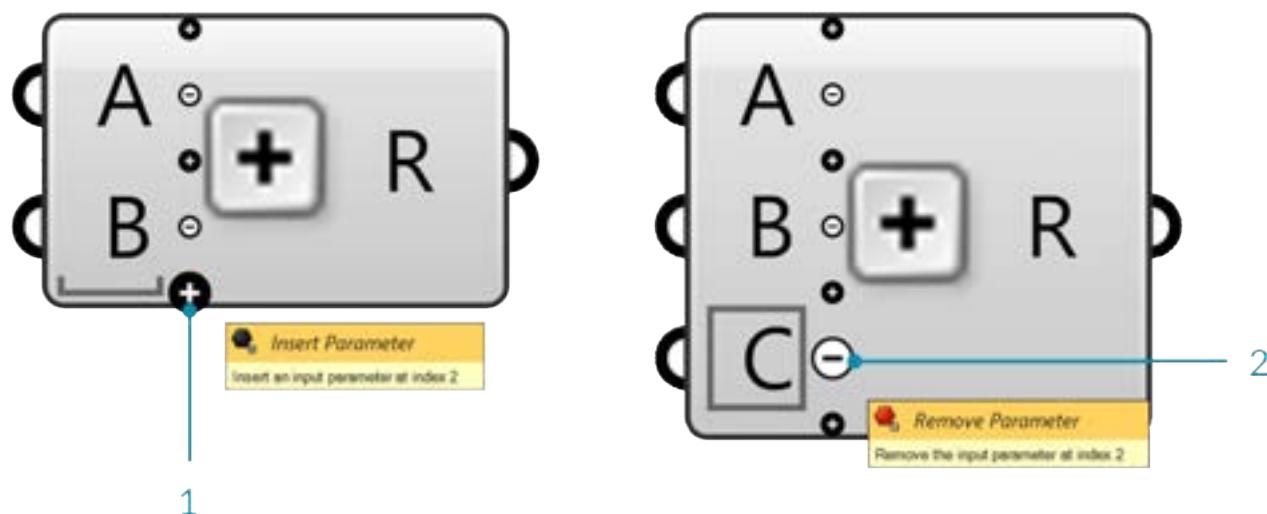
1. Komponenten Kontextmenu.
2. Bearbeitbares Textfeld, das den Namen des Objektes listet.
3. Vorschauanzeige - zeigt an, ob die von diesem Objekt erzeugte Geometrie in den Rhinoansichtsfenstern sichtbar ist. Die Vorschau auszuschalten wird die Aktualisierungsrate der Ansichtsfenster in Rhino, sowie die Zeit zur Berechnung der Lösungen beschleunigen.
4. Laufzeitwarnungen - hier werden Warnungen gelistet, welche die Komponente bei der Ausführung ihrer Funktion behindern.



1. C Eingabe Kontextmenu.
2. Waehle einen oder mehrere Punkte - ermoeglicht es Dir Referenzgeometrien im Rhinoansichtsfenster auszuwaehlen.
3. Verwalte Punktsammlungen - oeffnet ein Dialogfenster, das es Dir erlaubt Punkte zu einer Punktsammlung hinzuzufuegen oder Punkte davon zu entfernen und Informationen zu jedem Punkt einzusehen.
4. Fuege ein Element zur Sammlung hinzu.
5. Loesche die Auswahl.

### 1.2.2.5. VERGROESSERBARE BENUTZEROBERFLAECHE

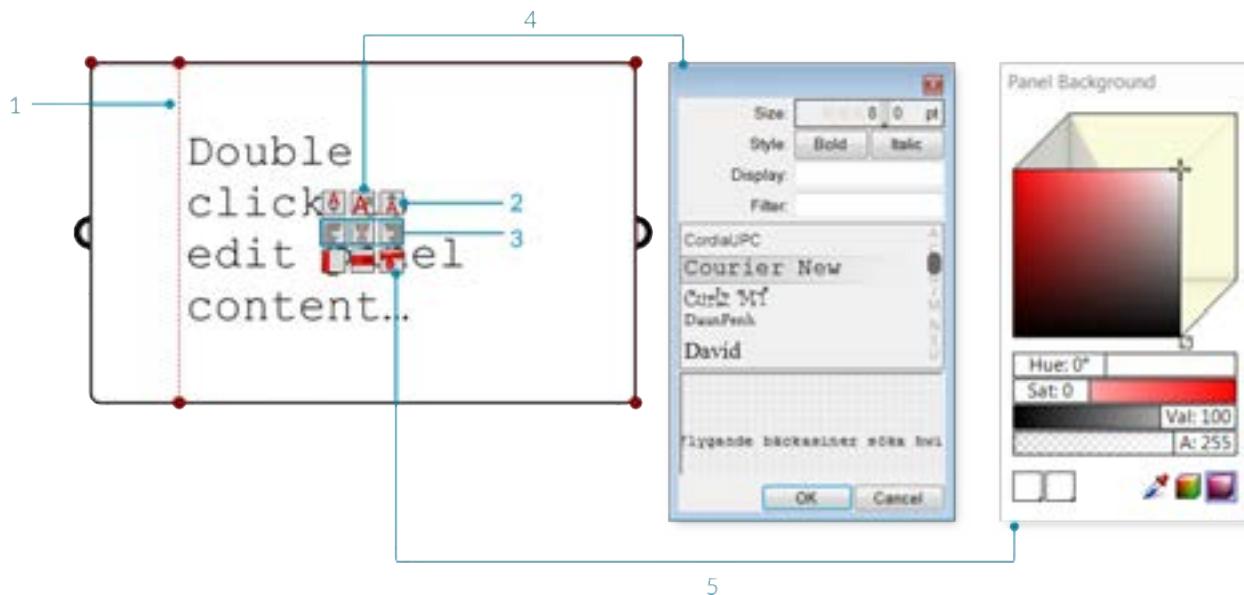
Einige Komponenten koennen durch vergroesserbare Benutzeroberflaechen bearbeitet werden, um die Anzahl von Eingabe- und Ausgabeparametern zu veraendern. Indem Du auf die entsprechende Komponente auf der Leinwand zoomst, kannst Du weitere Optionen einsehen, welche es Dir ermoeglichen Eingabe- und Ausgabeparameter von der Komponente zu entfernen oder u ihr hinzuzufuegen. Die "Addition" Komponente ermoegliche es Dir Eingabeparameter hinzuzufuegen und so weitere Elemente fuer die Additionsoperation darzustellen.



1. Klicke auf das + Zeichen um einen Eingabeparameter hinzuzufuegen.
2. Klicke auf das - Zeichen um einen Eingabeparameter zu entfernen.

2. Klicke auf das - Zeichen um einen Eingabeparameter zu entfernen.

Die Paneelkomponente besitzt ebenso eine vergroesserbare Benutzeroberflaeche. Ein Paneel ist wie ein Post-It™ Sticker. Es erlaubt Dir kleine Anmerkungen und Erklaerungen zu Deinem Dokument hinzuzufuegen. Du kannst den Text des Paneels durch das Menu oder mit einem Doppelklick auf die Oberflaeche veraendert werden. Paneele koennen auch Daten aus anderen Quellen empfangen und anzeigen. Wenn Du einen Ausgabeparameter an ein Paneel anschliesst, kannst Du die Inhalte des entsprechenden Parameters in Echtzeit sehen. Alle Daten in Grasshopper koennen auf diese Weise angezeigt werden. Wenn Du auf ein Paneel zoomst, erscheint ein Menu, welches es Dir erlaubt den Hintergrund, die Schriftart und andere Eigenschaften anzuzeigen. Diese Optionen sind auch ueber einen Rechtsklick auf das Paneel zugaenglich.



1. Ziehe an den Raendern, um die Panelabmessungen zu veraendern.
2. Erhoehe oder reduziere die Schriftgroesse des Panelinhaltes.
3. Veraendere die Ausrichtung des Panelinhaltes.
4. Waehle die Schriftart fuer den Panelinhalt.
5. Waehle die Farbe des Panelhintergrundes. Du kannst einen neuen Standardfarbton fuer Paneele waehlen, indem Du nach einem Rechtsklick auf das Panel "Set Default Color" ausfuehrst.

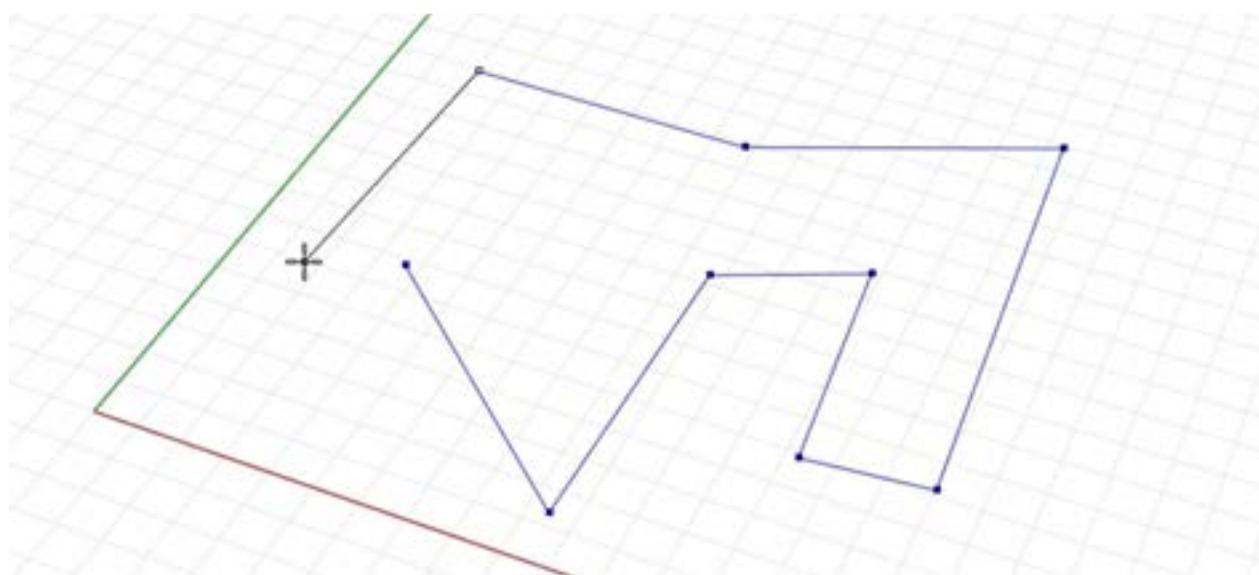
### ### 1.2.3. DATENTYPEN

Die meisten Parameter koennen zwei verschiedene Arten von Daten speichern: **Volatile** und **persistente Daten**. Volatile Daten sind von einer oder mehreren Quellen eingespeist und werden zerstoert (oder neuerhoben) wenn die Loesung erneut startet. Persistente Daten sind Daten, welche spezifisch vom Benutzer eingestellt werden.

#### 1.2.3.1. PERSISTENTE DATEN

Persistente Daten sind durch ein Menu, und abhaengig von der Art des Parameters durch unterschiedliche Manager, zugaenglich. Ein Punktparameter zum Beispiel ermoeigt es Dir einen oder mehrere Punkte durch sein Menu auszuwaehlen. Aber lass uns noch ein paar Schritte zurueckgehen und sehen, wie sich ein Punktparameter so verhaelt.

Wenn Du einen Punktparameter vom "Params/Geometry Panel" auf die Leinwand ziehest, ist der Parameter orange, was anzeigt, dass der Parameter eine Warnung enthaelt. Es ist nichts ernstes, da die Warnung Dich einfach darueber informiert, dass der Parameter leer ist (er enthaelt keine persistenten Eintraege und er konnte keine volatilen Daten empfangen) und deshalb keinen Effekt auf das Ergebnis der Loesung hat. Das Kontextmenu des Parameters bietet zwei Wege der Einstellung von persistenten Daten: einzelne oder mehrere. Rechtsklicke auf den Parameter um mehrere Punkte zu setzen. Sobald Du auf eines dieser Menuelemente klickst, wird da Grasshopperfenster verschwinden und Du wirst aufgefordert werden einen Punkt in einem der Rhinoansichtsfenster auszuwaehlen.



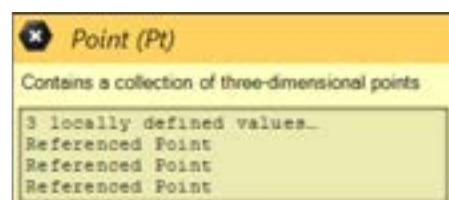
Sobald Du alle Punkte bestimmt hast, kannst Du Enter druecken und sie werden Bestandteil der persistenten Datensammlung des Parameters werden. Das bedeutet, dass der Parameter nicht mehr leer ist und seine Farbe von orange zu grau aendert. (Merke, dass der Informationsbalon in der rechten oberen Ecke der Komponente ebenfalls verschwunden ist und keine Warnungen mehr vorhanden sind). An diesem Punkt kannst Du die gespeicherten Punkte in diesem Parameter fuer jede weitere folgende Eingabe in Deiner Definition verwenden.



1



2



3

1. Der Parameter ist orange und zeigt damit an, dass er keine persistenten Dateneintraege enthaelt (und keine volatilen Daten empfangen hat) und somit keinen Effekt auf das Ergebnis der Loesung hat. Rechtsklicke auf einen beliebigen Parameter um persistente Daten auszuwaehlen.
2. Sobald der Parameter einige persistente Daten enthaelt, wird die Komponente ihre Farbe von orange zu grau aendern.
3. Der Werkzeugtip fuer einen Punktparameter zeigt die persistenten Daten (eine Sammlung von Referenzpunkten), die gespeichert ist.

### 1.2.3.2. VOLATILE DATEN

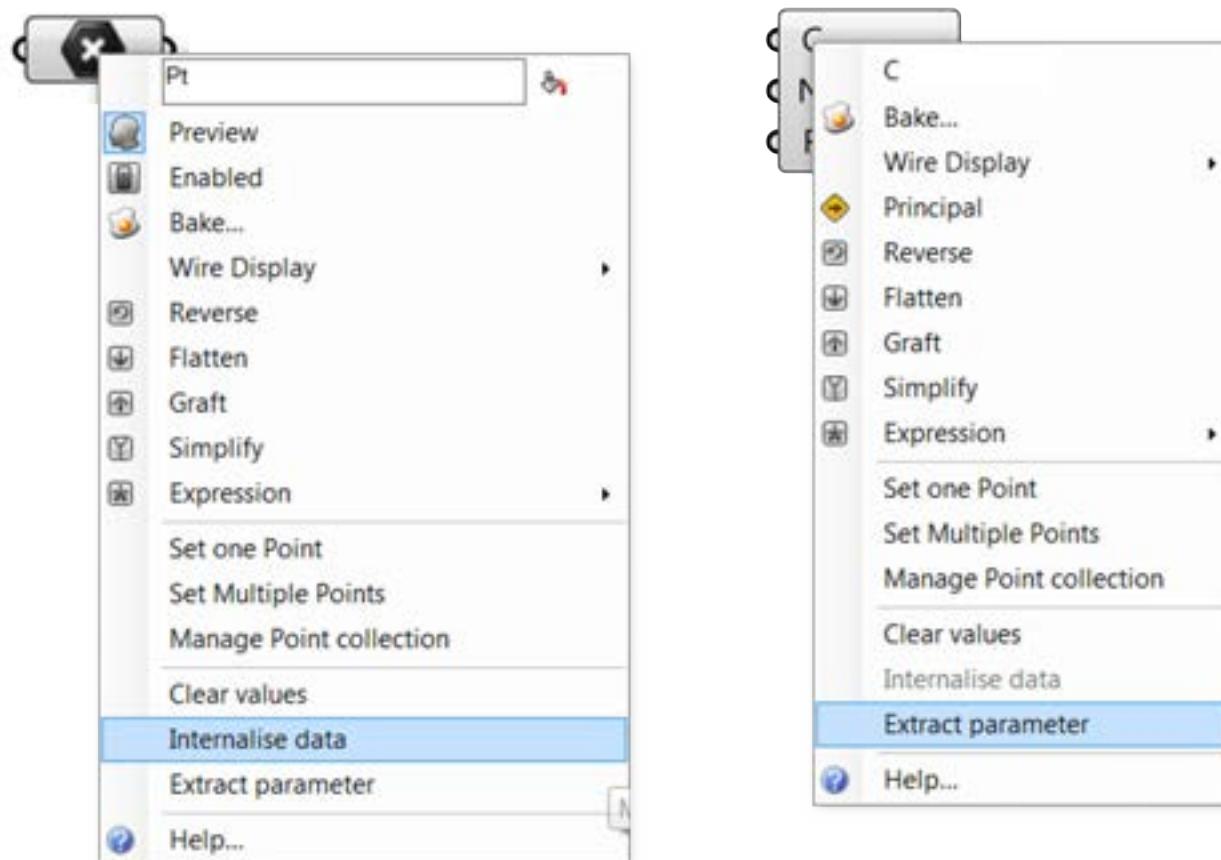
Volatile Daten, wie der Name schon nahelegt, sind nicht permanent und werden jedes Mal wenn die Loesung abgeschlossen ist geloescht. Jedoch kann eine Begebenheit ausloesen, dass die Loesung wiederaufgebaut und die Szene aktualisiert wird. Generell werden die meisten Daten, die waehrend der Erzeugung der Loesung entstehen als volatile bezeichnet.

Wie bereits genannt, werden Grasshopperdaten in Parametern gespeichert (entweder in volatiler oder in persistenter Form) und in verschiedenen Komponenten genutzt. Wenn Daten nicht als permanente Daten in Parametern gespeichert werden, muss sie implizit aus einer anderen Quelle kommen. Jeder Parameter (ausser Ausgabeparameter) definiert woher er seine Daten bezieht und die meisten Parameter sind dabei nicht besonders spezifisch. Du kannst einen "Number"-Parameter (welche nur besagt, dass es sich um eine Dezimalzahl handelt) mit einem Integereingabeparameter verbinden und er wird sich um die Umwandlung kümmern.

Du kannst den Weg in dem Daten bezogen und gespeichert werden im Kontextmenu eines Parameters oder eines Komponenteneingabeparameters veraendern. Um gespeicherte Referenzgeometrie aus Rhino in der Grasshopperdefinition selbst zu veraendern, rechtsklicke auf den Parameter und wähle "Internalise Data" aus dem Menu. Dies ist hilfreich, wenn Deine Grasshopperdatei unabhaengig von der referenzierten Rhinodatei werden soll.

Du kannst auch Daten in Komponenteneingabeparametern internalisieren. Sobald Du "Internalise Data" im Menu ausgewählt hast, werden alle Kable von dem entsprechenden Eingabeparameter entfernt. Die Daten wurden von volatile zu persistent umgewandelt und werden im weiteren Verlauf nicht mehr aktualisiert.

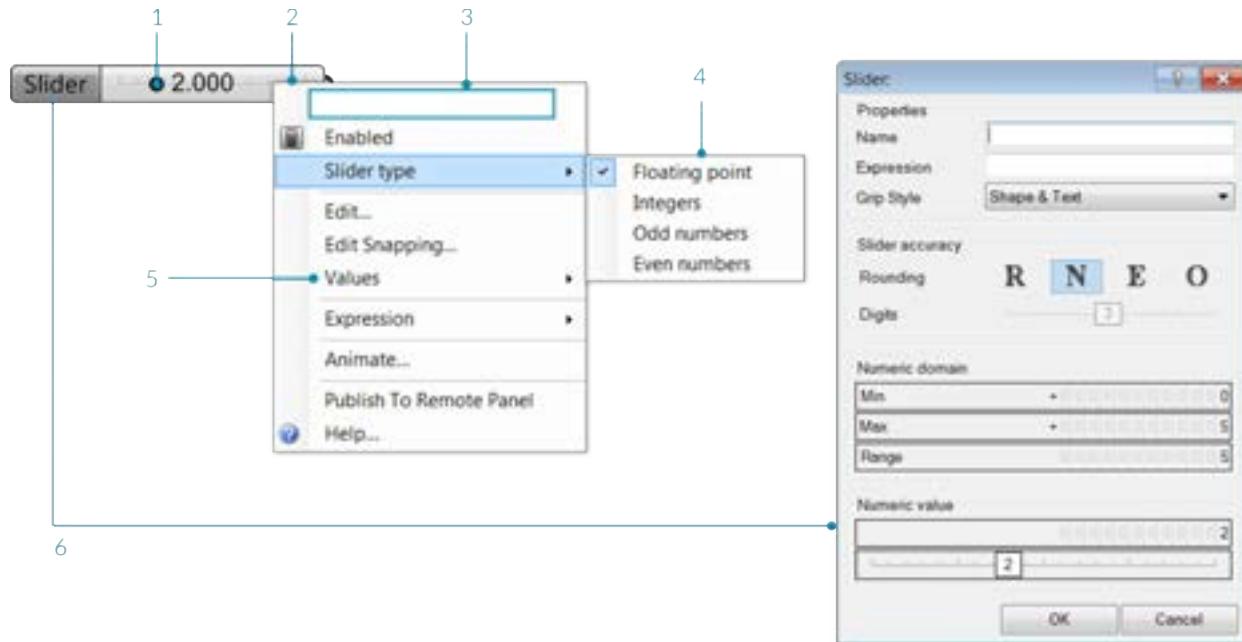
Wenn Du willst, dass die Daten wieder volatile werden, kannst Du einfach wieder Kabel in den Eingabeparameter einstecken und die Werte werden automatisch ersetzt. Du kannst auch auf den Eingabeparameter rechtsklicken und "Extract parameter" wählen. Grasshopper wird einen Parameter erzeugen und ihn mit dem Eingabeparameter verbinden, der die Daten enthaelt.



### 1.2.3.3. EINGABEPARAMETER

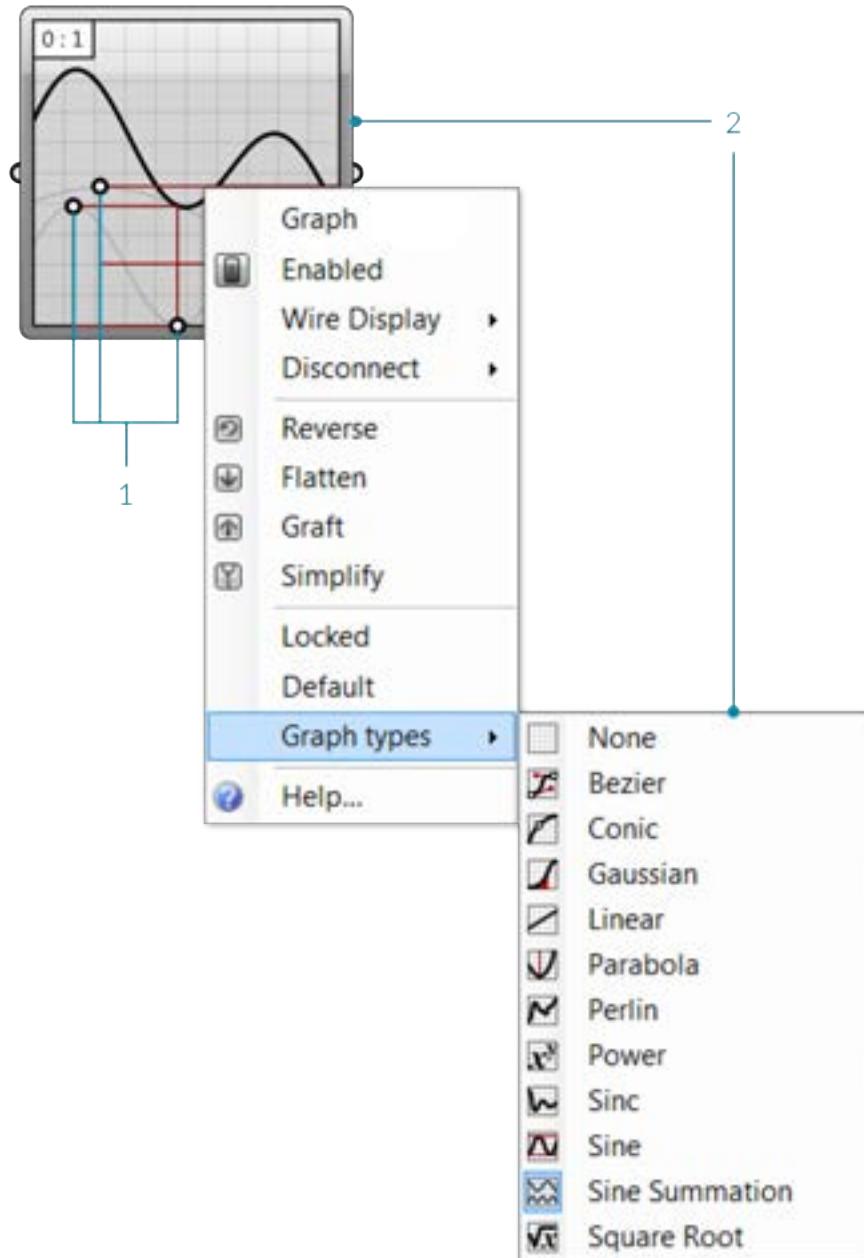
Grasshopper hat eine Bandbreite an Parametern, die Dir die Moeglichkeit eroeffnen die Komponenteingabeparameter mit den Daten zu verbinden und dadurch Kontrolle ueber die veraenderlichen Ergebnisse in der Definition auszuueben. Diese Parameter veraendern sich mit der entsprechenden Eingabe und erzeugen volatile Daten.

**Schieberegler** Die Schieberegler ("number slider") sind die wichtigsten und am haeufigsten genutzten Eingabeparameter. Sie erlauben es uns Werte zwischen zwei Extremen auszugeben, indem wir mit einem Mausgriff interagieren. Schieberegler koennen benutzt werden um einen bestimmten Wert zu spezifizieren und die Veraenderung der Definition zu beobachten, die durch den Mausgriff erzeugt werden. Ein Schieberegler aollte jedoch auch als Mittel gesehen werden, um erfolgreich Wertegrenzen unserer Definition festzulegen.

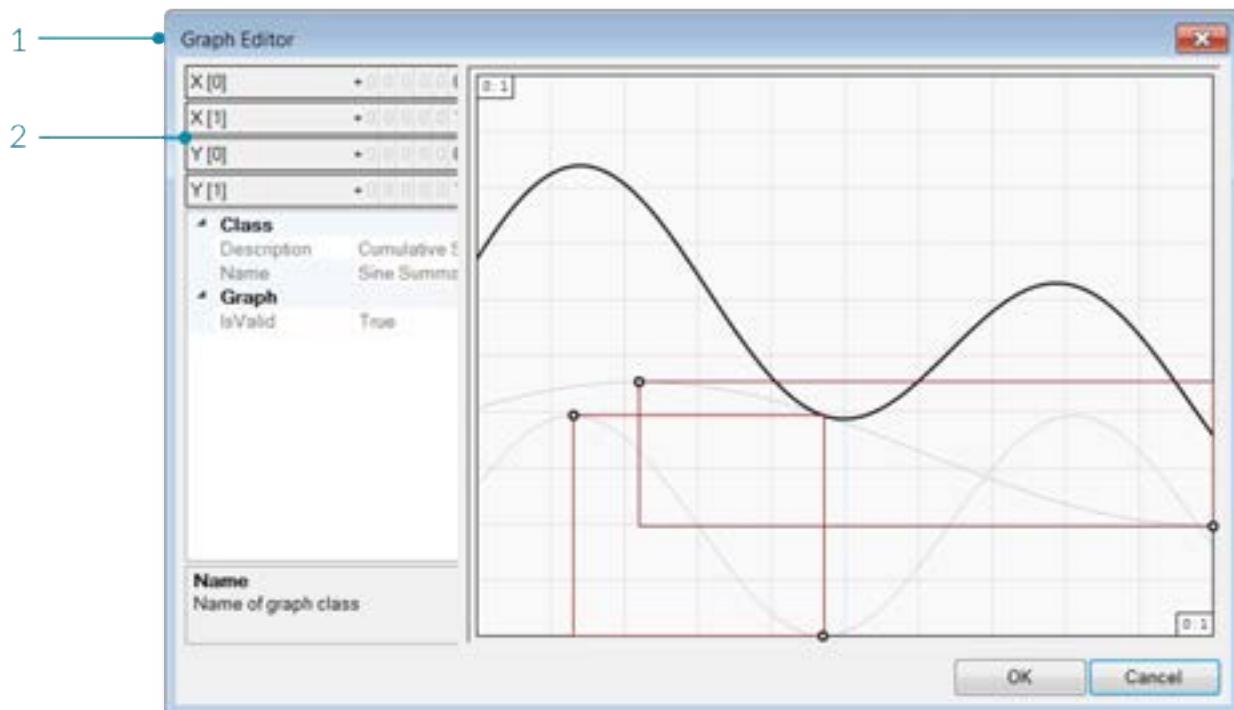


1. Ziehe den Schieberegler um seinen Wert zu ändern - jedes Mal, wenn Du dies tust wird Grasshopper die Lösung erneut berechnen.
2. Rechtsklick auf eine Schiebereglerkomponente um ihren Namen, Typ und Wert zu ändern.
3. Editierbares Textfeld für den Namen des Schiebereglers.
4. Wähle den Typ an Zahlen den der Schieberegler verwenden soll.
5. Veränder die Wertegrenzen des Schiebereglers.
6. Doppelklick auf den Namen der Schiebereglerkomponente um den Schiebereglereditor zu öffnen.

**Graph mapper** Der "Graph Mapper" ist eine zwei-dimensionale Benutzeroberfläche mit der wir die numerischen Werte modifizieren können, indem wir die Eingabewerte an der x-Achse des Graphs antragen und die entsprechenden Funktionswerte entlang der y-Achse ausgeben. Er ist sehr nützlich um einen Satz Werte innerhalb einer intuitiven, Mausgriff-basierten Benutzeroberfläche modulieren zu können.

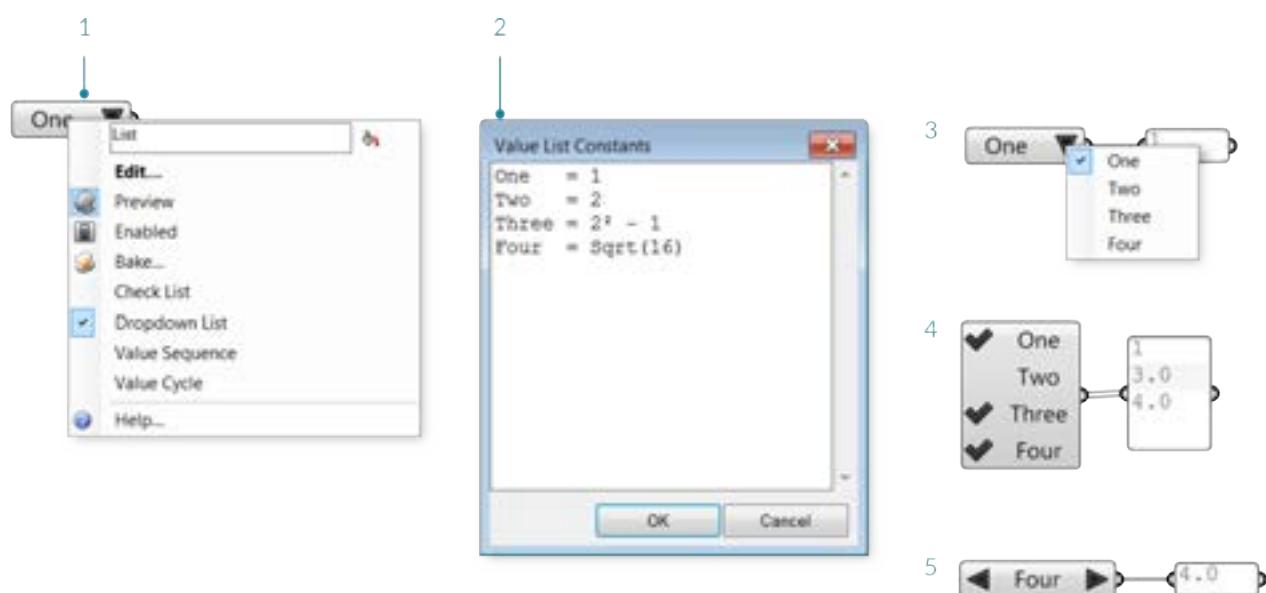


1. Bewege die Griffe um den Graph zu bearbeiten - jedes Mal, wenn Du dies tust, wird Grasshopper die Lösung neu berechnen.
2. Rechtsklicke die "Graph mapper" Komponente um den Graphtypen zu wählen.



1. Doppelklicke den "Graph mapper" um den Grapheditor zu oeffnen.
2. Aendere die x und y Domäne.

**Value List** Die Werteliste speichert eine Sammlung von Werten entsprechend einer Liste entlang einer entsprechenden Liste von Labels, zugewiesen, durch ein Gleichheitssymbol. Sie ist im Besonderen hilfreich, wenn Du ein paar Moeglichkeiten zur Auswahl haben willst, die entsprechend bedeutungsvoll benannt wurde um bestimmte Ausgabewerte zur Verfuegung zu haben.



1. Rechtsklicke auf die Werteliste Komponente und waehle eine Option aus dem Menu.
2. Doppelklicke die Werteliste Komponente um den Editor zu oeffnen und Werte hinzuzufuegen oder zu aendern.
3. Im Dropdownmodus, klicke auf den Pfeil um einen der Werte auszuwaehlen. Die Loesung wird jedes Mal neu berechnet, wenn Du den Wert aenderst.
4. Im Checklistenmodus, klicke neben die Werte, um sie auszuwaehlen. Die Komponente wird alle ausgewaehlten Werte ausgeben.

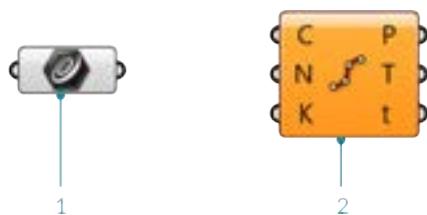
5. Im Wertesequenz- und Wertzyklusmodus, klicke die Pfeile nach links und rechts, um Dich durch die Werte zu bewegen.

## 1.2.4. KOMPONENTEN VERKABELN

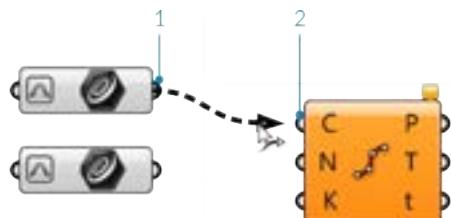
Wenn Daten nicht in einem permanenten Satz eines Parameters gespeichert sind, muessen sie von einer anderen Quellen uebernommen werden. Daten werden von einer Komponente zur anderen durch Kabel uebertragen. Du kannst Dir diese buchstaeblich als elektrische Kabel vorstellen, die Datenimpluse von einem Objekt zum anderen uebertragen.

### 1.2.4.1. VERBINDUNGSMANAGEMENT

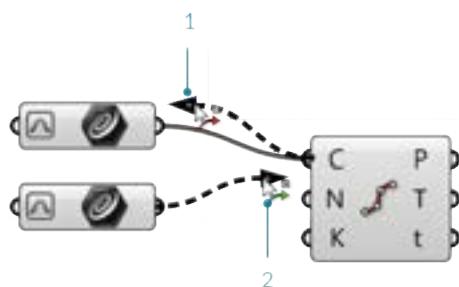
Um Komponenten zu verbinden, klicke und ziehe nahe dem Kreis an der Ausgabeseite eines Objektes. Ein Verbindungskabel wird an Deiner Maus angeheftet. Sobald die Maus ueber einem moeglichen Zieleingabeparameter schwebt, wird sich das Kabel verbinden und verfestigen. Dies ist keineswegs eine permanente Verbindung bis Du die Maustaste loslaesst. Es macht keinen Unterschied, ob wir die Verbindungen von "links nach rechts" oder von "rechts nach links" erstellen.



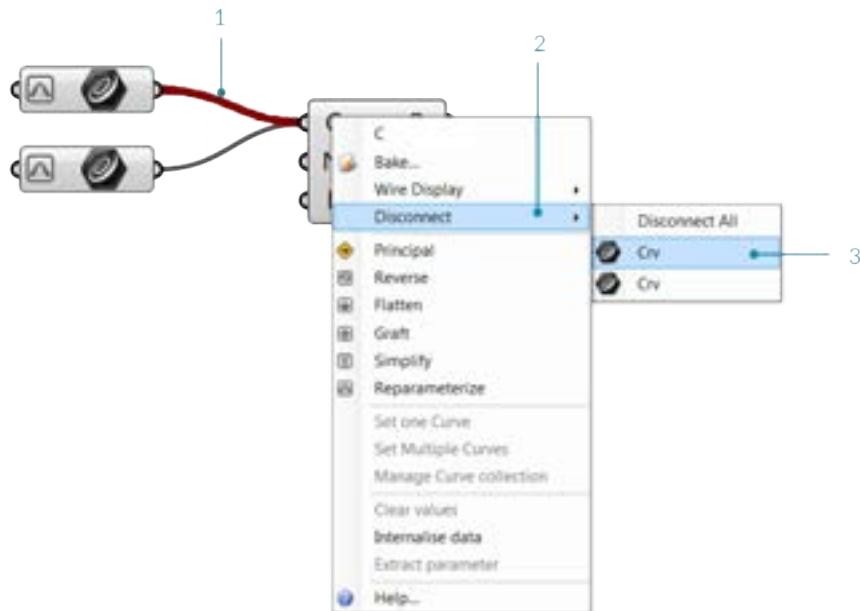
1. Die "Divide Curve" Komponente teilt eine Kurve in gleich lange Segmente.
2. "Curve" Parameter - rechtsklicke und waehle "Set One Curve" um eine Rhinogeometrie zu referenzieren.



Linksklicke und ziehe ein Kabel vom Ausgabeparameter (1.) eines Objekts zu einem Eingabeparameter (2.) eines anderen.



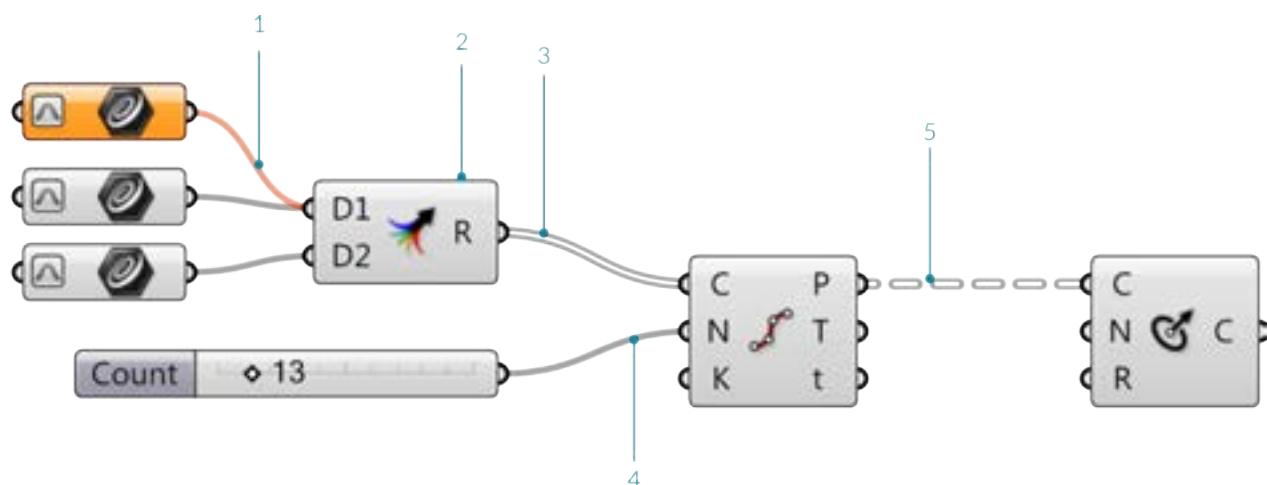
1. Wenn Du STRG haeltst, wird der cursor rot werden und die Zielquelle wird aus der Quellenliste entfernt werden.
2. Als Standard werden neue Verbindungen bestehende Verbindungen loeschen. Halte SHIFT gedrueckt, waehrend Du das Verbindungskabel ziehst um mehrere Quellen zu definieren. Der Cursor wird gruen um das additive Verhalten darzustellen.



1. Du kannst Kabel ebenfalls durch das Kontextpopupmenu entfernen - rechtsklicke auf den Griff am Eingabe- oder Ausgabeparameter und waehle "disconnect".
2. Wenn mehrere Verbindungen bestehen, kannst Du in einer Liste aussuchen, welche Du loesen moechtest.
3. Wenn Du Deine Maus ueber ein Element bewegst, werden die entsprechenden Verbindungen in rot hervorgehoben.

### 1.2.4.2. FANCY WIRES

Kabel repräsentieren die Verbindungen, sowie den Fluss der Daten in einem Graph einer Definition. Grasshopper kann nun auch visuellen Hinweise darauf geben, was in den Kabel so vor sich geht. Die Standardeinstellung fuer diese sogenannten "fancy wires" ist ausgeschalten, weshalb Du sie einschalten musst, bevor Du die verschiedenen Typen von Linien fuer die Verbindungskabel sehen kannst. Um dies zu tun, klicke einfach auf den "View Tab" in der Hauptmenuleiste und waehle den Knopf genannt "Draw Fancy Wires". Fancy wires kann Dir viele Informationen darueber welche Typen von Information von einer Komponente zur anderen fliesen geben.



1. Leeres Element - Ein orangener Kabeltyp zeigt an, dass keine Information darin uebertragen wird. Dieser Parameter hat eine Warnung erzeugt weil er keine Daten enthaelt und damit keine Information ueber das Kabel gesendet wird.
2. Die "Merge" Komponente ist eine alternative um mehrere Quellen mit einem einzelnen

Eingabeparameter zu verbinden.

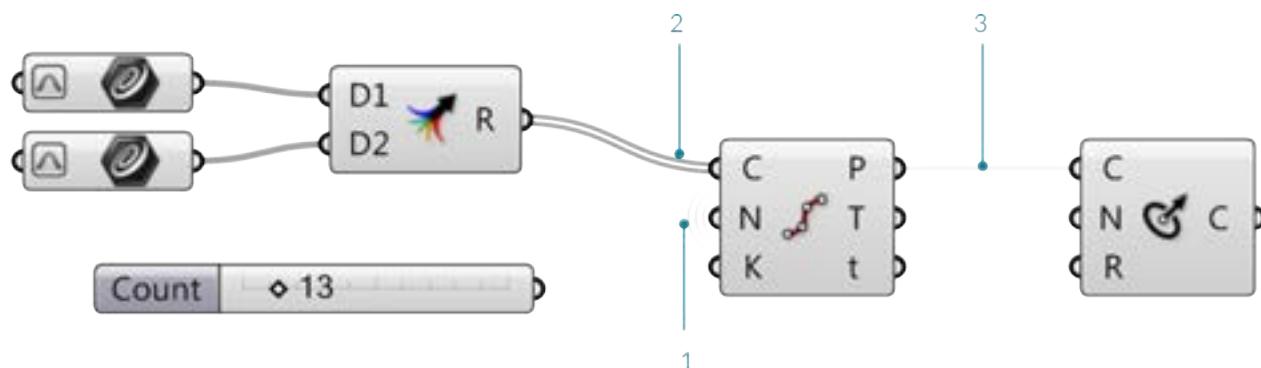
3. Liste – Wenn die Information, welche von einer Komponente ausfließt eine Liste von Informationen enthaelt, wird der Kabeltyp als graue Doppellinie dargestellt.
4. Einzelnes Element – Die Daten aus einem beliebigen Parameter enthalten ein einzelnes Element und das Kabel wird als durchgehende graue Linie dargestellt.
5. Baum – Informationen, die zwischen Komponenten uebertragen werden und eine Datenstruktur enthalten, werden als graue, gestrichelte Doppellinie dargestellt.

### 1.2.4.3. KABELDARSTELLUNG

Wenn Du eine lange Zeit damit zugebracht hast an einer einzelnen Grasshopper definition zu arbeiten, hast Du eventuell gemerkt, dass die Leinwand mit relativ schnell mit einer Menge kabel zugekleistert wird.

Gluecklicherweise haben wir die Faeigkeit die Darstellung der Kabel fuer jeden einzelnen Eingabeparameter einer Komponente zu bestimmen.

Dabei gibt es drei Kabeldarstellungen: "Default Display", "Faint Display", and "Hidden Display". Um die Kabeldarstellung zu bearbeiten, rechtsklicke einfach auf den Eingabeparameter einer Komponente und waehle eine der Darstellungen im "Wire Display" Popupmenu aus.



1. Versteckte Darstellung – Wenn "hidden display" ausgewaehlt wurde, wird das Kabel komplett unsichtbar. Die Daten werden dann kabellos von der Quelle zum Eingabeparameter uebertragen. Wenn Du die Quelle oder die Zielkomponente auswaehlst, wird ein grunes Kabel erscheinen und Dir zeigen welche Komponenten untereinander verbunden sind. Sobald Du die Komponente nicht mehr ausgewaehlt hast, wird das Kabel wieder verschwinden.
2. Standarddarstellung – Die Option "Default Wire Display" wird alle Verbindungen zeigen (wenn "Fancy Wires" eingeschalten wurde).
3. Matte Darstellung – "Faint Wire Display" zeichnet eine sehr duenne, semitransparente Verbindung. Matte und versteckte Darstellung koennen sehr hilfreich sein, wenn viele Quellkabel an einem einzelnen Eingabeparameter ankommen.

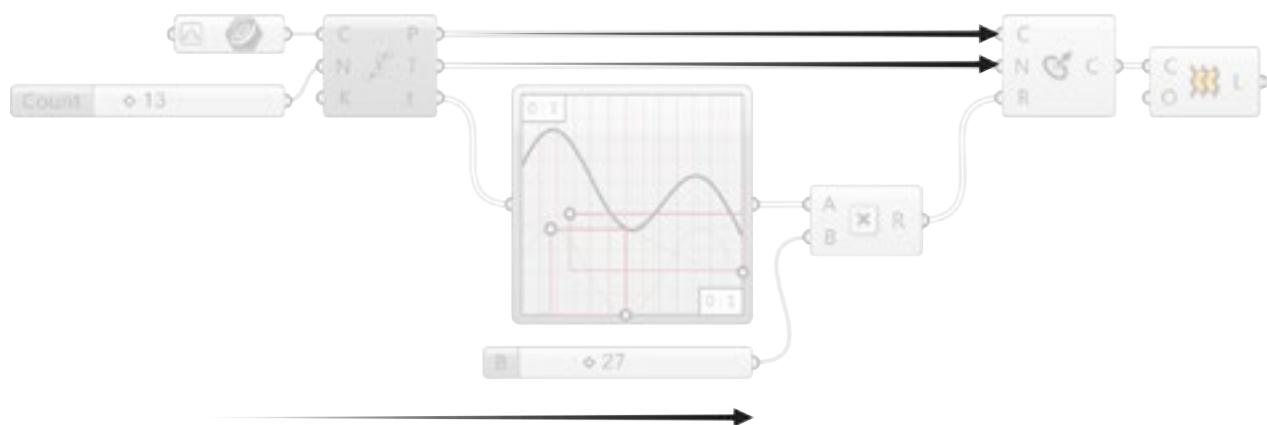
### ### 1.2.5. DIE GRASSHOPPERDEFINITION

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

**Grasshopperdefinitionen haben einen Programmfluss, der darstellt, an welcher Stelle das Programm die Ausfuehrung beginnt, was in der Mitte passiert und wie man sehen kann, dass die Ausfuehrung abgeschlossen ist.**

#### 1.2.5.1. PROGRAMMFLUSS

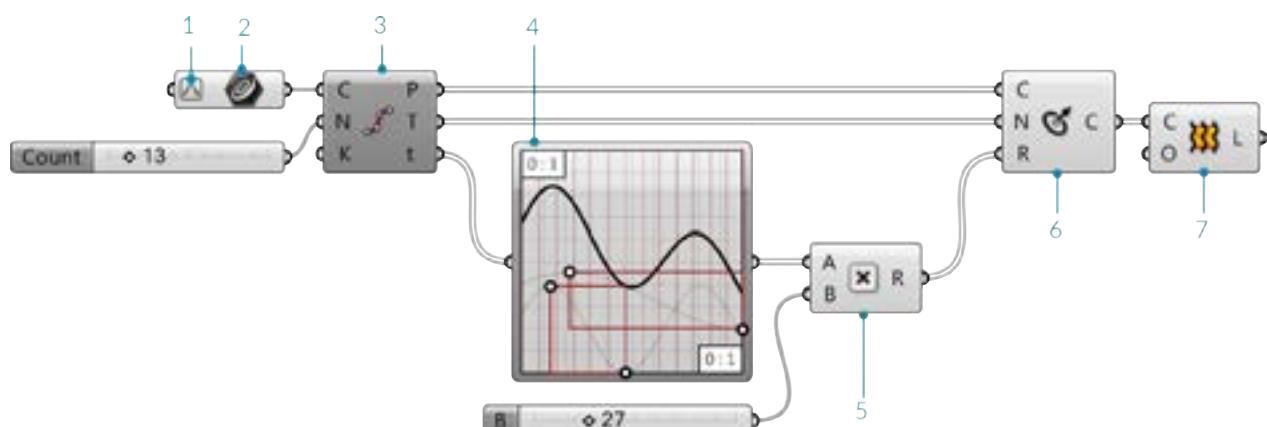
Visuelle Grasshopperprogramme werden von links nach rechts ausgefuehrt. Das Lesen des Graphen in Abhaengigkeit von den verkabelten Verbindungen von stromaufwaerts nach stromabwaerts ermoeigt das Verstaendnis des Programmflusses.



Richtung des Datenflusses von links nach rechts.

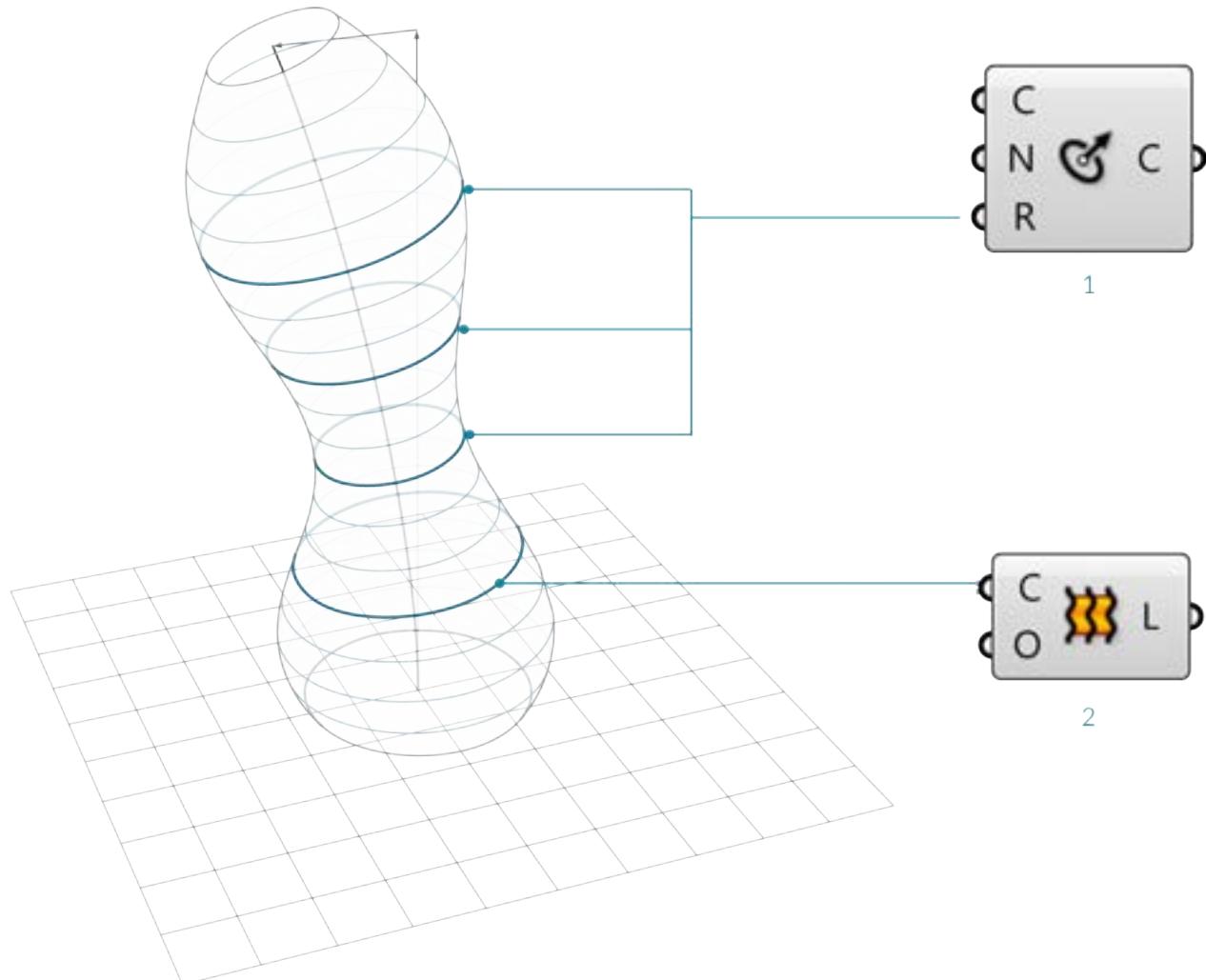
#### 1.2.5.2. DER LOGISCHE PFAD

Alle Objekte und Kabel, welche Objekte verbinden stellen den logischen Graphen unseres Programmes dar. Dieser Graph offenbart den Datenfluss, die Abhaengigkeiten eines jeden Eingabeparameters von dem mit ihm verbundenen Ausgabeparameter. Zu jeder Zeit, an der sich unser Graph veraendert, sozusagen "verunreinigt" wird, wird jede Komponente flussabwaerts und jede Verbindung in dieser Richtung aktualisiert.



1. Reparametrisiere die Domaene einer Kurve auf die Werte von 0.0 bis 1.0.
2. Referenziere eine Kurve von Rhino.
3. Teile eine Kurve in 13 gleiche Teile.
4. Sende die Parameterwerte eines jeden Teilpunktes der Kurventeilung durch den Graphmapper.
5. Multipliziere jeden Wert mit 27.

6. Zeichne einen Kreis und jedem Teilpunkt entlang der Kurve normal zum Tangentialvektor an jedem Punkt, mit dem Radius definiert durch den Parameterwert ( $t$ ), der durch den Graphmapper und die Multiplikation mit 27 modifiziert wurde.
7. Lofte eine Fläche zwischen den Kreisen.



1. Variabler Kreisradius.
2. Loft zwischen Kreisen.

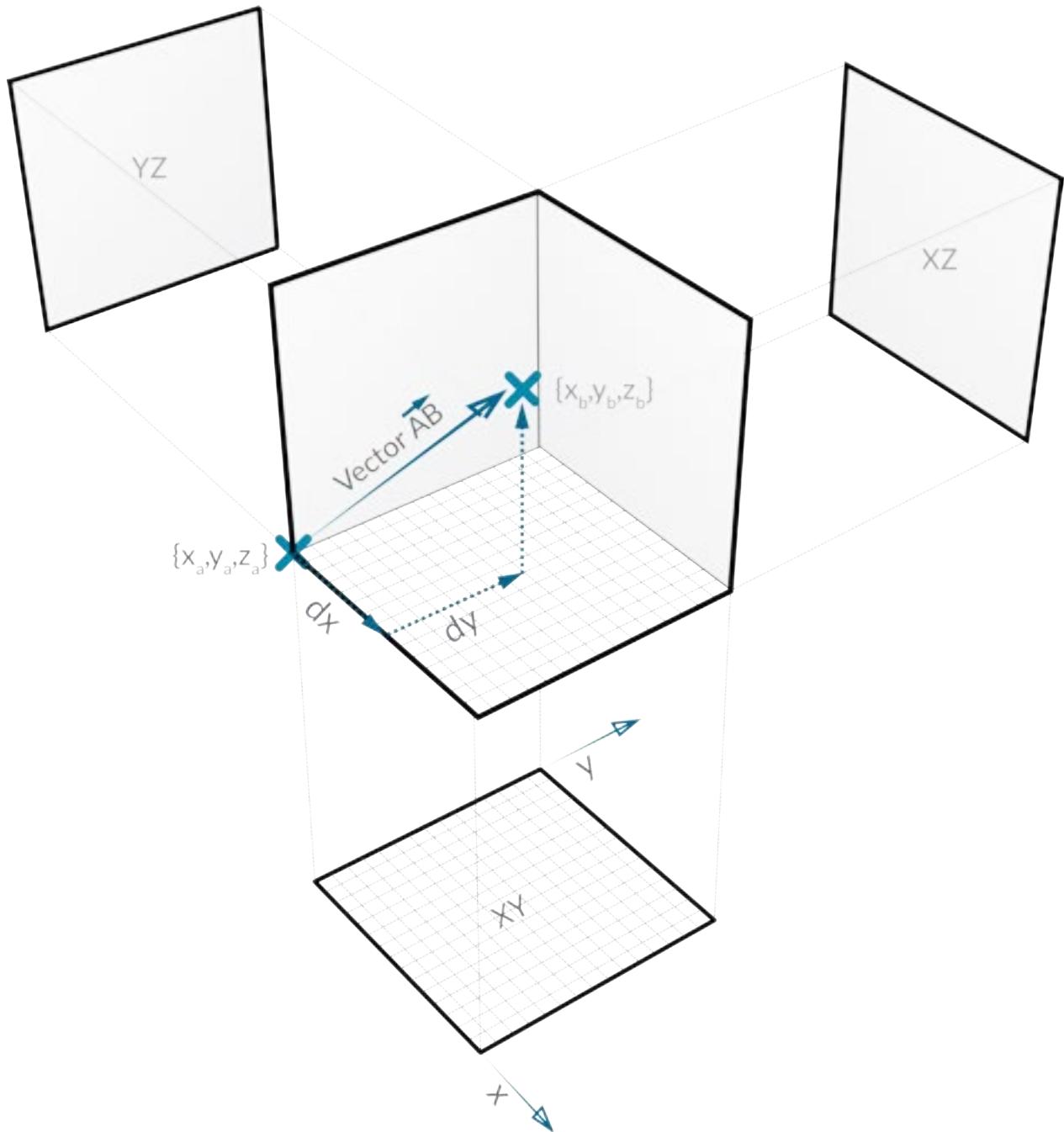
## 1.3. BAUSTEINE FUER ALGORITHMEN

Dieses Kapitel wird die geometrischen und mathematischen Grundkonzepte einfuehren und darstellen wir diese in Grasshopper implementiert und manipuliert werden.



### ###1.3.1. Punkte, Ebenen & Vektoren

**Alles beginnt mit Punkten. Ein Punkt ist nicht mehr als ein oder mehrere Werte, die Koordinaten genannt werden. Die Anzahl der Koordinatenwerte korrespondieren mit der Anzahl der Dimensionen des Raums in welchem sie dargestellt werden. Punkte, Ebenen und Vektoren sind die Basis der Erstellung und Transformation von Geometrie in Grasshopper.**

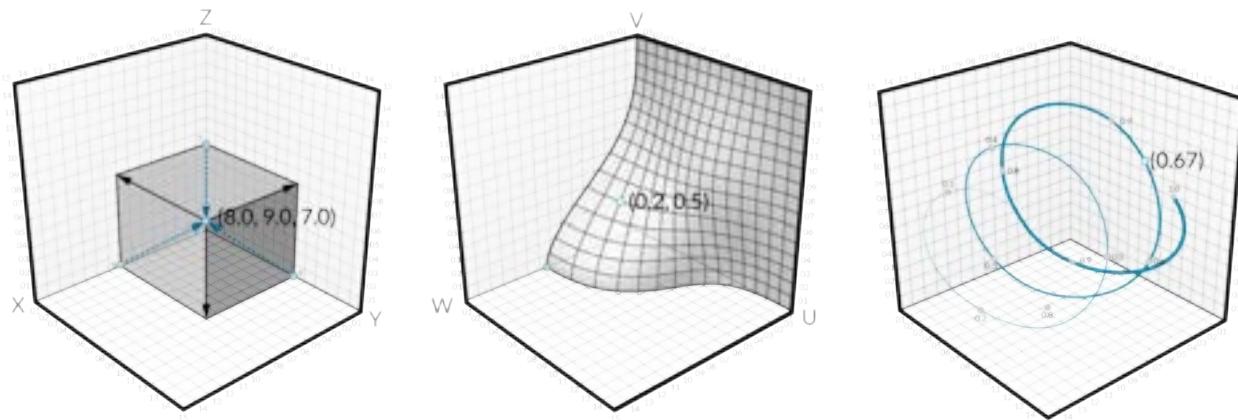


#### 1.3.1.1 PUNKTE

Punkte im dreidimensionalen Raum haben drei Koordinaten, normalerweise bezeichnet als  $[x,y,z]$ . Punkte in einem zweidimensionalen Raum haben nur zwei Koordinaten, die entweder  $[x,y]$  oder  $[u,v]$  genannt werden, abhaengig davon, von welcher Art zweidimensionalem Raum wir sprechen. 2D Parameterraum ist an eine begrenzte Flaeche gebunden. Er ist immer noch kontionuierlich und kann hypothetisch gesprochen immer noch unendlich viele Punkte auf der Flaeche darstellen, aber die maximale Distanz zwischen beliebigen Punkten dieser Menege ist begrenzt. 2D Parameterkoordinaten sind nur gueltig, wenn sie eine bestimmte Spanne nicht uebersteigen. In der Beispielzeichnung, wurde diese Spanne zwischen 0.0 und 1.0 fuer die Richtungen  $[u]$  und  $[v]$  definiert, aber sue

koennte jede beliebige begrenzte Domaene annehmen. Ein Punkt mit den Koordinaten [1.5, 0.6] wuerde irgendwo ausserhalb der Flaeche liegen und ist deshalb ungultig.

Since the surface which defines this particular parameter space resides in regular 3D world space, we can always translate a parametric coordinate into a 3D world coordinate. The point [0.2, 0.5] on the surface for example is the same as point [1.8, 2.0, 4.1] in world coordinates. Once we transform or deform the surface, the 3D coordinates which correspond with [0.2, 0.5] will change.

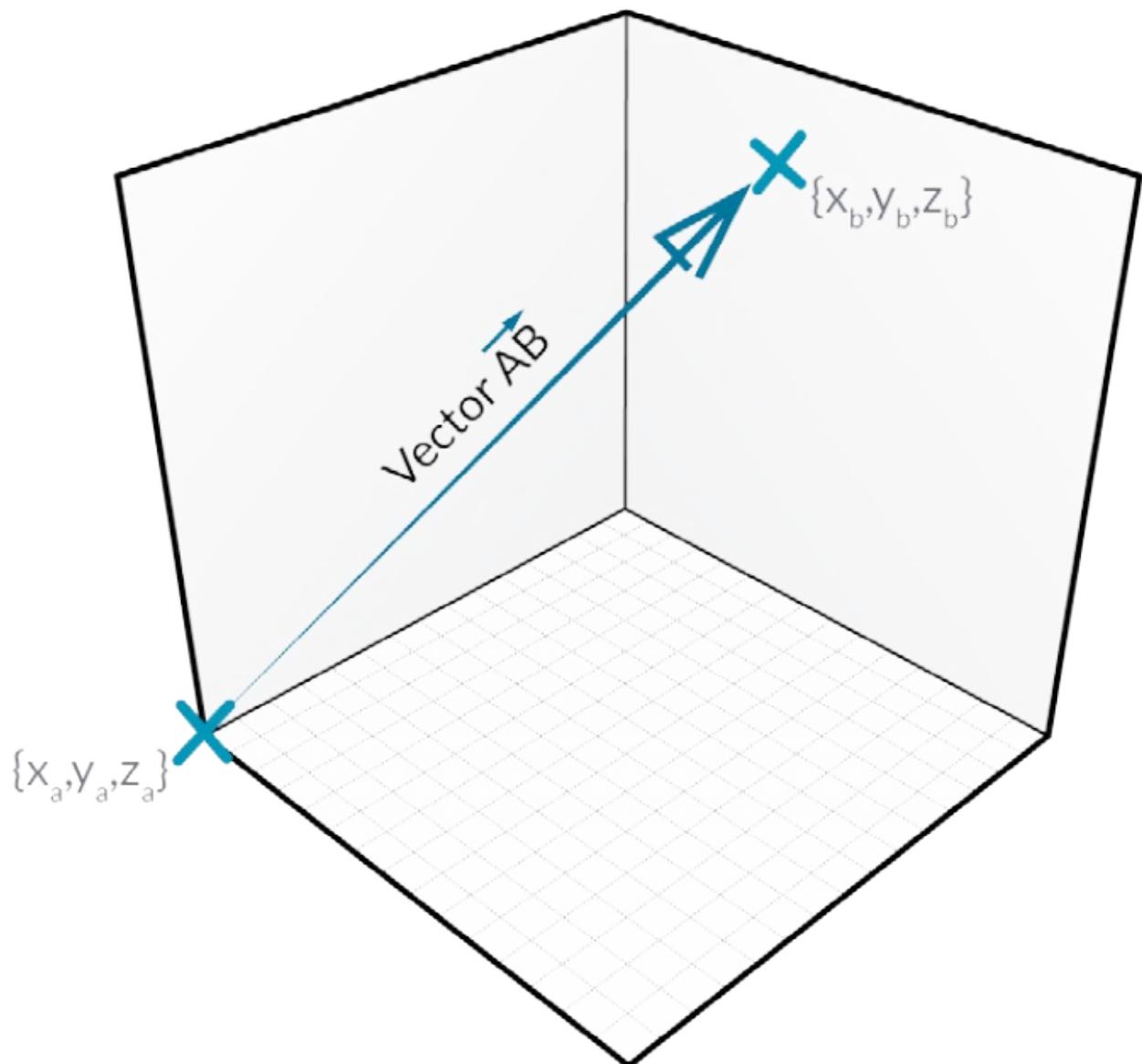


Wenn es schwer ist, dieses Konzept zu verstehen, kannst Du Dir eventuell damit behelfen, Dir Deine eigene Position im Raum vorzustellen. Wir neigen dazu unsere lokalen Koordinatensysteme zu benutzen um zu beschreiben, wo wir uns befinden; "Ich sitze im dritten Sitz in der siebten Reihe des Kinos", "Ich bin auf dem Ruecksitz". Wenn das Auto in dem Du Dich befindest auf der Strasse unterwegs ist, veraendern sich Deine globalen Koordinaten staendig, auch wenn Du Dich immer noch auf der selben Ruecksitz-"Koordinate" befindest.

### 1.3.1.2. VEKTOREN

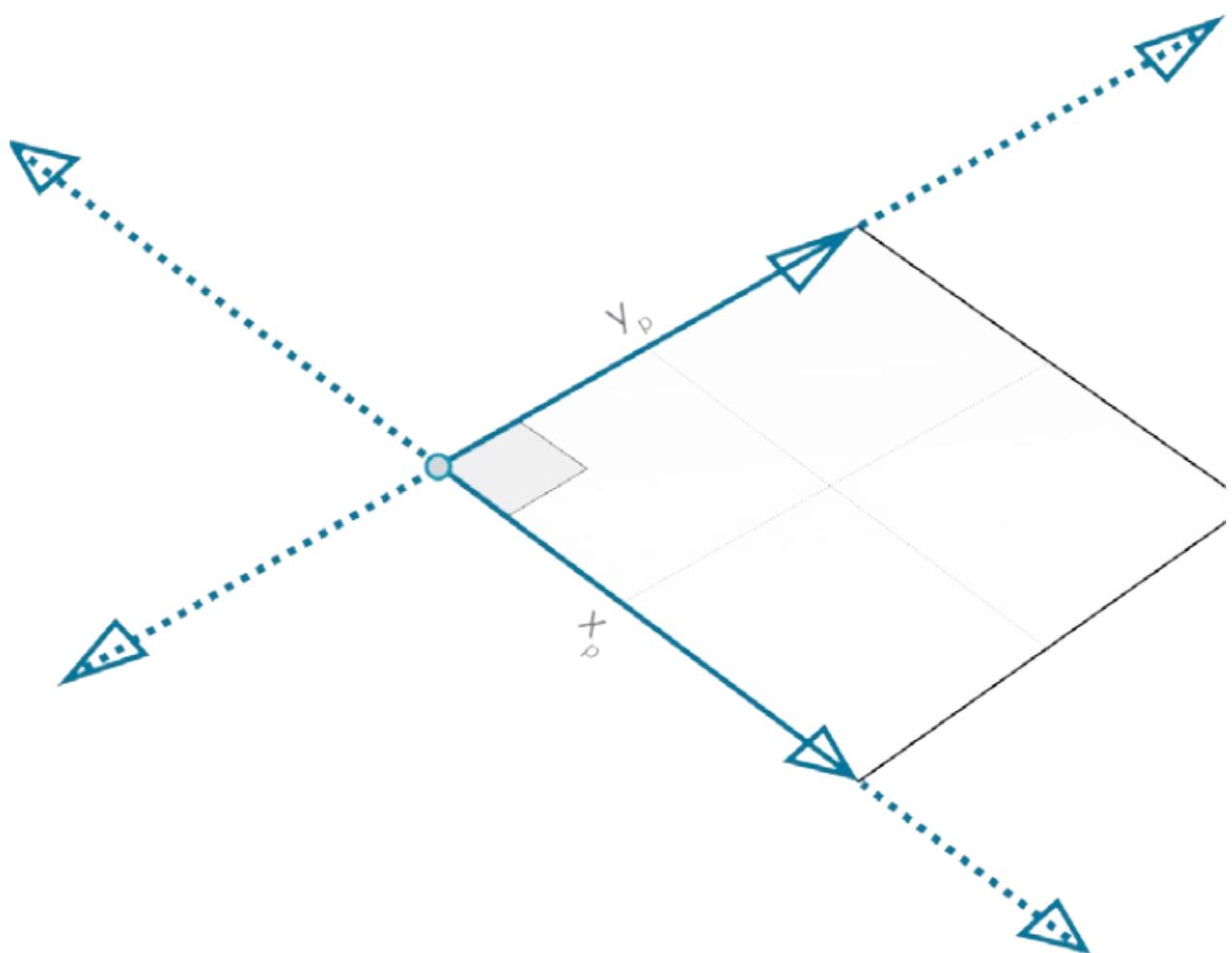
Ein Vektor ist eine geometrische Groesse, die Richtung und Staerke beschreibt. Vektoren sind abstrakt; das heisst, dass sie eine Groesse beschreiben und nicht ein geometrisches Element.

Vektoren sind von Punkten nicht zu unterscheiden. Das bedeutet, dass die beiden Listen von je drei Zahlen sich so wenig unterscheiden, dass man nicht mit Sicherheit sagen kann, ob sie Punkte oder Vektoren darstellen. Es gibt jedoch einen praktischen Unterschied; Punkte sind absolut, Vektoren sind relativ. Wenn wir eine Liste von drei Dezimalzahlen als einen Punkt behandeln, stellt sie eine bestimmte Koordinate dar im Raum dar. Wenn wir sie als Vektor behandeln, stellt sie eine bestimmt Richtung dar. Ein Vektor ist ein Pfeil in einem Raum, der immer vom Weltursprung (0.0, 0.0, 0.0) beginnt und an einer bestimmten Koordinate endet.



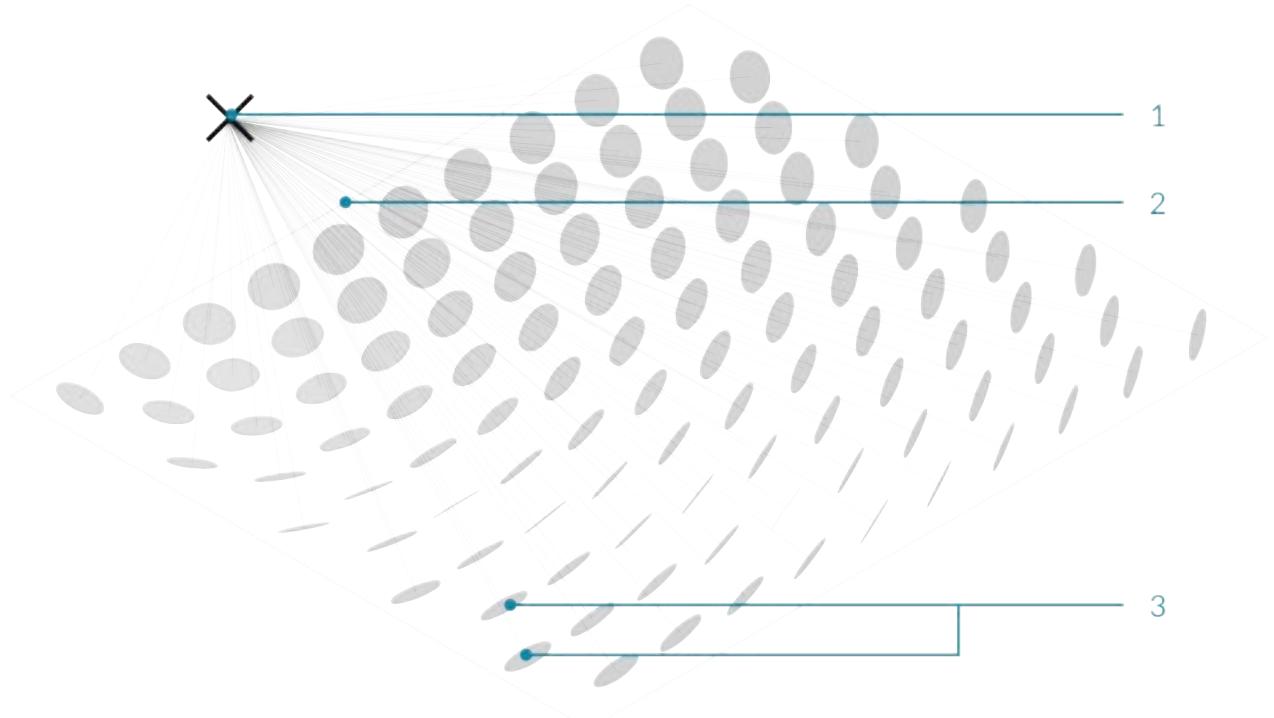
### 1.3.1.3. EBENEN

Ebenen sind "flach" und dehnen sich unendlich in zwei Dimensionen aus, während sie ein lokales Koordinatensystem definieren. Ebenen sind keine expliziten Objekte in Rhino, da sie lediglich dazu verwendet werden um Koordinatensysteme im 3D Weltraum zu bestimmen. Faktisch ist es am Besten sich die Ebenen als Vektoren vorzustellen, da sie lediglich mathematische Konstrukte sind.



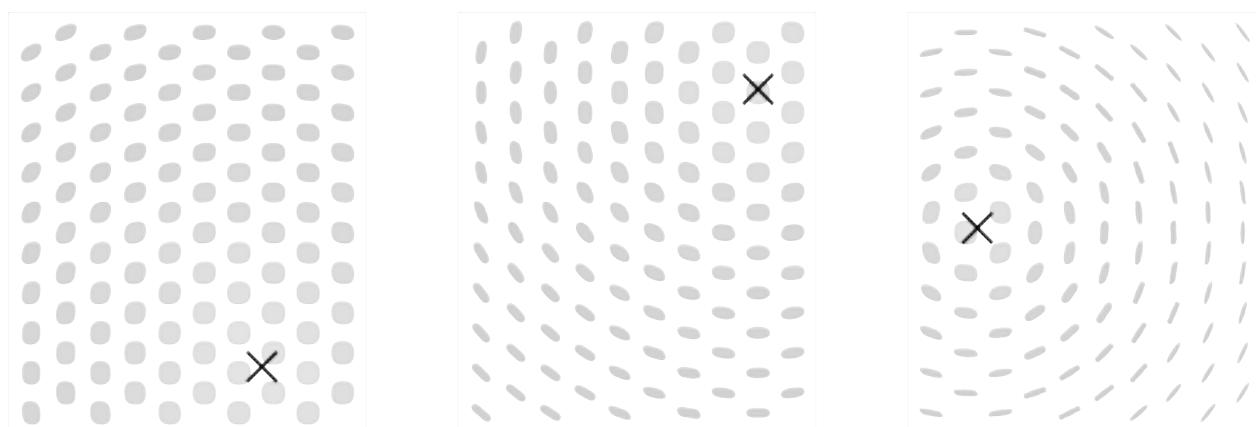
## 1.3.2. Arbeiten mit Attraktoren

Attraktoren sind Punkte, die wie virtuelle Magneten wirken - jeder zieht andere Objekte an oder stoet diese ab. In Grasshopper, kann jede Geometrie von Rhino referenzierte oder in Grasshopper erstellte Geometrie als Attraktor verwendet werden. Attraktoren koennen eine Anzahl von Parametern der sie umgebenden Objekte beeinflussen, inkl. Skalierung, Rotation, Farbe und Position. Diese Parameter werden im Verhaeltnis zur Attraktorgeometrie veraendert.



- 1. Attraktorpunkt
- 2. Vektoren
- 3. Kreise orientieren sich in Richtung des Attraktors entsprechend ihrer Normalen

Im Bild oberhalb werden Vektoren zwischen dem Attraktorpunkt und dem Zentrum eines jeden Kreises gezeichnet. Diese Vektoren werden verwendet um die Ausrichtung der Kreise zu definieren, damit sie immer zum Attraktorpunkt zeigen. Der selbe Attraktor koennte benutzt werden um andere Parameter der Kreise zu veraendern. Zum Beispiel koennte der Kreis am naehesten zum Attraktorpunkt groesser skaliert werden, indem man die Laenge jedes Vektors verwendet um den Radius eines jeden Kreises zu skalieren.



### 1.3.2.1. ATTRAKTORDEFINITION

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

In diesem Beispiel werden wir einen Attraktorpunkt nutzen um ein Raster aus Kreisen, basierend auf den Vektoren zwischen ihren Zentren und dem Attraktorpunkt, auszurichten. Jeder Kreis wird ausgerichtet, so dass sein Normalenvektor auf den Attraktorpunkt zeigt.

01.	Druecke Strg+N in Grasshopper um eine neue Definition zu beginnen	
02.	<b>Vector/Grid/Hexagonal</b> - Ziehe eine <b>Hexagonal Grid</b> Komponente auf die Leinwand	
03.	<b>Parameter/Input/Slider</b> - Ziehe zwei <b>Numeric Sliders</b> auf die Leinwand	
04.	Doppelklicke auf den ersten <b>Numeric Slider</b> und setze die folgenden Werte:	

Double-click on the first **Numeric Sliders** and set the following:

Name: Cell Radius

Rounding: Floating Point

Lower Limit: 0.000

Upper Limit: 1.000

Value: 0.500

|| |05.| Doppelklicke auf den zweiten \*\*Numeric Slider\*\* und setze die folgenden Werte:

Name: # of Cells

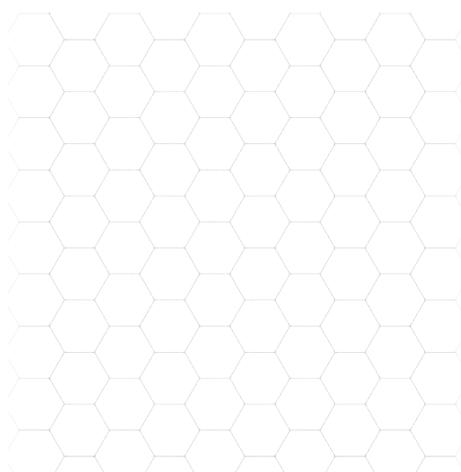
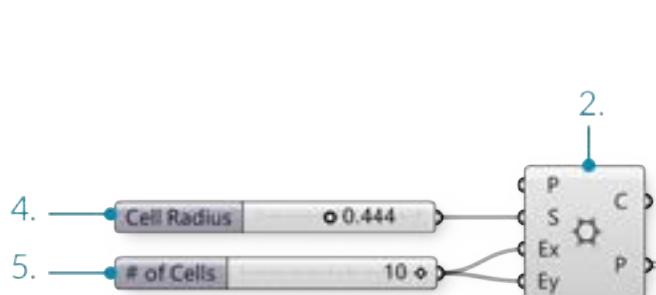
Rounding: Integers

Lower Limit: 0

Upper Limit: 10

Value: 10

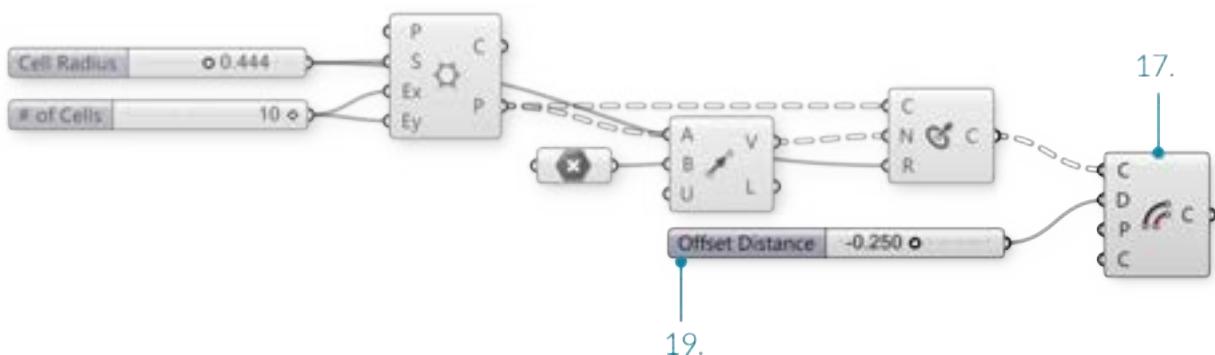
|| |06.| Verbinde den **Numeric Slider** (Cell Radius) mit dem Eingabeparameter fuer die Groesse (S) der **Hexagon Grid** Komponente|| |07.| Verbinde den **Numeric Slider** (# of Cells) mit dem Abmessung X Eingabeparameter (Ex) und dem mit dem Abmessung Y Eingabeparameter (Ey) der **Hexagon Grid** Komponente|||

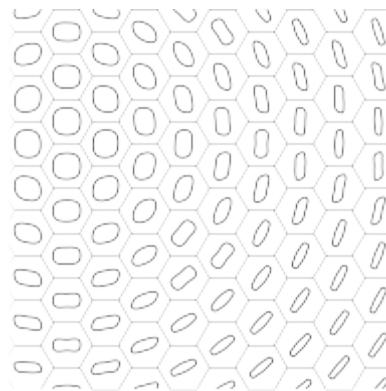


--	--	--

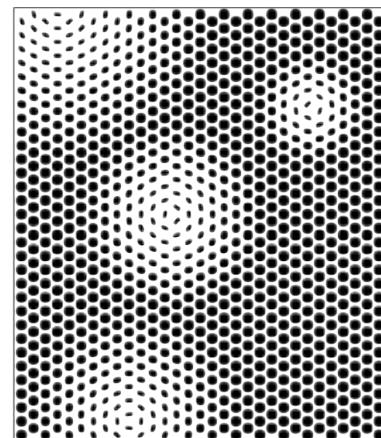
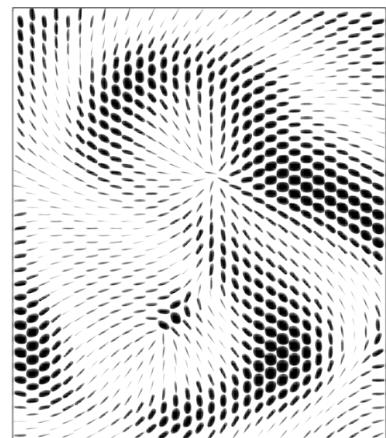
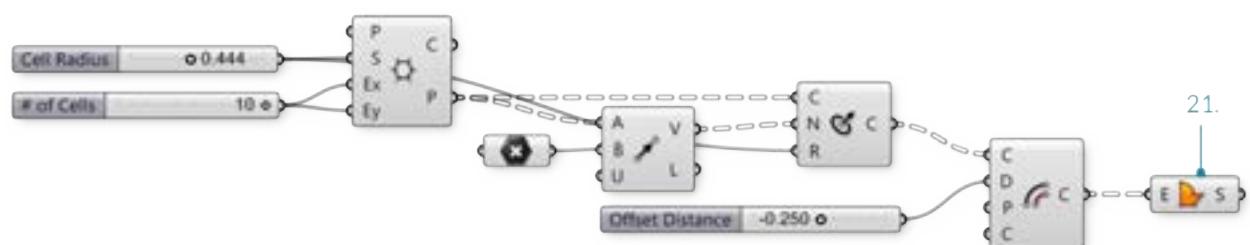
08.	<b>Curve/Primitive/Circle CNR</b> - Ziehe eine <b>Circle CNR</b> Komponente auf die Leinwand	
09.	Verbinde den Ausgabeparameter fuer Punkte (P) von <b>Hexagon Grid</b> mit dem Zentrumeingabeparameter (C) input der <b>Circle CNR</b> Komponente	
10.	Verbinde den <b>Numeric Slider</b> (Cell Radius) mit dem Radius Eingabeparameter (R) der <b>Circle CNR</b> Komponente.	
11.	<b>Vector/Vector/Vector 2Pt</b> - Ziehe die <b>Vector 2Pt</b> Komponente auf die Leinwand	
12.	Verbinde den Punkt ausgabeparameter (P) der <b>Hexagonal Grid</b> Komponente zum Basispunkt Eingabeparameter (A) der <b>Vector 2Pt</b> Komponente.	
13.	<b>Params/Geometry/Point</b> – Ziehe die <b>Point</b> Komponente auf die Leinwand	
14.	Rechtsklicke die <b>Point</b> Komponente und waehle einen Punkt. Im Modellraum kannst Du waehlen an welcher Stelle Du den Attraktorpunkt haben moechtest	
15.	Verbinde die <b>Point</b> Komponente mit dem Endpunkteingabeparameter (B) der <b>Vector 2Pt</b> Komponente	
16.	Verbinde den Vektor Ausgabeparameter (V) von <b>Vector 2Pt</b> mit dem Normalenvektoreingabeparameter(N) der <b>Circle CNR</b> Komponente.	

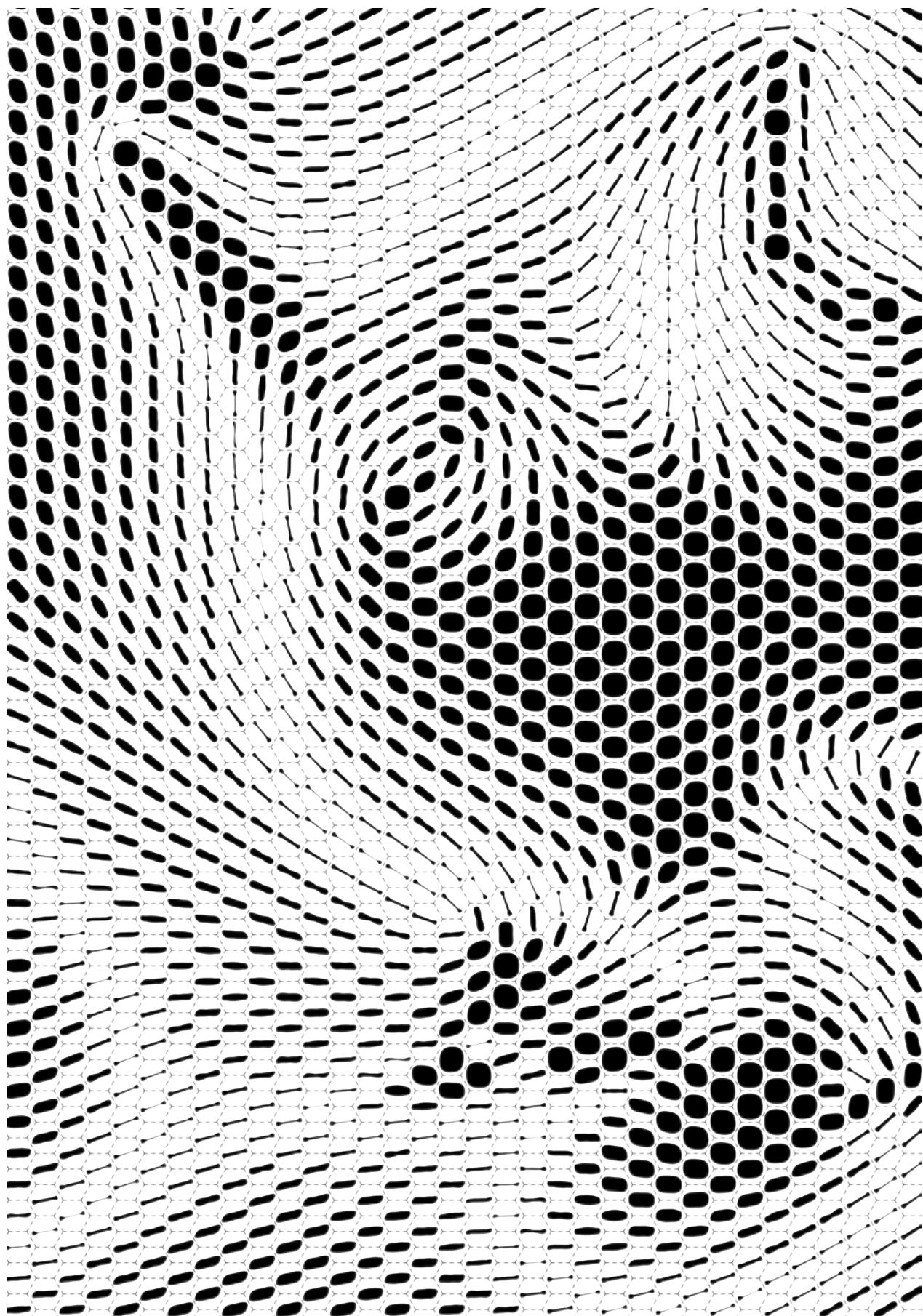
17.	<b>Curve/Util/Offset</b> – Ziehe die <b>Offset</b> Komponente auf die Leinwand.	
18.	<b>Params/Input/Slider</b> - Ziehe einen <b>Numeric Slider</b> auf die Leinwand	
19.	Doppelklicke auf den Schieberegler und setze die folgenden Werte: Name: Offset Distance Rounding: Floating Point Lower Limit: - 0.500 Upper Limit: 0.500 Value: -0.250	
20.	Verbinde den <b>Numeric Slider</b> (Offset Distance) mit dem Eingabeparameter Distanz(D) der <b>Offset</b> Komponente	





21.	Surface/Freeform/Boundary Surfaces – Ziehe <b>Boundary Surfaces</b> auf die Leinwand	
22.	Verbinde den Ausgabeparameter Kurven (C) der <b>Offset</b> Komponente mit dem Kanten Eingabeparameter (E) der <b>Boundary Surfaces</b> Komponente	





### 1.3.3. Mathematik, Funktionen & Konditionale

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

**Zu wissen wie man mit nummerischen Informationen umgeht, ist eine grundlegende Fertigkeit, die Du lernen musst um Grasshopper zu nutzen. Grasshopper enthaelt viele Komponenten um mathematische Operationen auszufuehren, Konditionale auszuwerten und Mengen von Zahlen zu manipulieren.**

In der Mathematik werden Zahlen in Mengen organisiert und es gibt zwei mit denen Du wahrscheinlich bekannt bist:

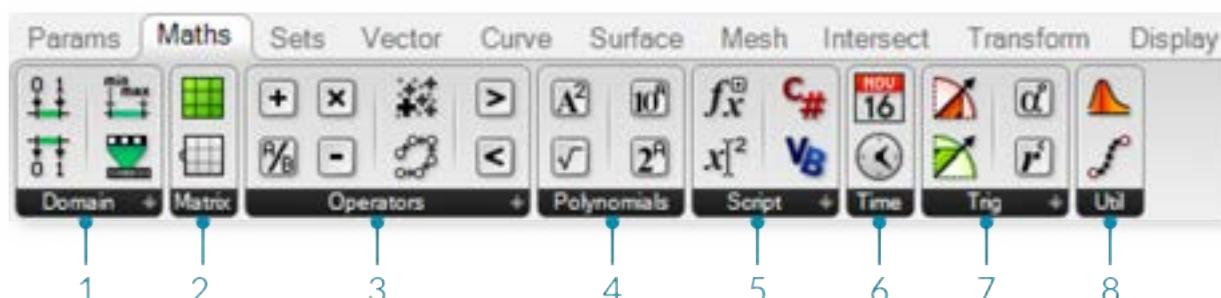
Integer Zahlen: [..., -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, ...]

Reelle Zahlen: [8, ..., -4.8, -3.6, -2.4, -1.2, 0.0, 1.234, e, 3.0, 4.0, ..., 8]

Neben anderen Zahlenmengen die es gibt, interessieren uns diese beiden am meisten, da Grasshopper diese bevorzugt verwendet. Obwohl es Grenzen in der Darstellung dieser Mengen gibt und diese in einer digitalen Umgebung genau definiert sind, koennen wir sie mit einem hohen Grad an Praezision annaehern. Zusaetlich sollten wir die Unterscheidung zwischen integralen Zahlenarten (integers) und Gleitkommazahlen (real numbers) als Unterschied zwischen einer diskreten und einer kontinuierlichen Domaene verstehen. In diesem Kapitel werden wir verschiedene Methoden fuer die Arbeit mit und die Auswertung von verschiedenen Zahlenmengen erkunden.

#### 1.3.3.1. DER MATH REITER

Die meisten Komponenten die mit mathematischen Operationen und Funktionen zu tun haben, koennen unter den folgenden Unterkategorien des "Math" Reiters gefunden werden:



1. Domaenen werden benutzt um die Bandbreite von Werten (frueher als Intervalle bekannt) zwischen zwei Werten zu definieren. Die Komponenten unter dem "Domain"Reiter erlauben es Dir verschiedene Domaenen zu schaffen oder zu zerlegen.
2. In der Mathematik organisiert eine Matrix eine Reihe von Zahlen in Zeilen und Spalten. Diese Unterkategorie enthaelt eine Reihe von nuetzlichen Werkzeugen um Matrizen zu schaffen und zu veraendern.
3. Operatoren werden genutzt um mathematische Operationen, wie Addition, Subtraktion, Multiplikation, u.a. auszufuehren. Konditionale Operatoren erlauben es festzulegen, ob eine Menge von Zahlen groesser als, kleiner als, oder gleich gross im Vergleich zu einer anderen Menge ist.
4. Polynome sind eines der wichtigsten Konzepte in Algebra, Mathematik und Wissenschaft. Du kannst die Komponenten aus dieser Unterkategorie nutzen um Fakultaeten, Logarithmen oder Exponentiale zu berechnen.
5. Die "Script" Unterkategorie enthaelt Einzel- und Multivariabel Ausdruecke, sowie VB.NET and C# Komponenten zur Entwicklung von Skripten.
6. Diese Komponeten ermoeglichen es Dir trigonometrische Funktionen, wie Sinus, Kosinus und Tangent

zu berechnen.

7. Die "Time"Unterkategorie hat eine Anzahl von Komponenten, die er Dir erlauben Instanzen von Datum und Zeit zu erstellen.
8. Die Unterkategorie "Utility" ist ein Sack von Komponenten, die in einer Bandbreite von mathematischen Gleichungen angewendet werden koennen. Suche hier, wenn Du versucht Minima oder Maxima ueber zwei Listen von Zahlen zu finden oder den Durchschnitt einer Menge von Zahlen zu berechnen.

### 1.3.3.2. OPERATOREN

Wie bereits genannt, sind Operatoren eine Menge an Komponenten, die algebraische Funktionen mit zwei numerischen EingabevARIABLEn nutzen, welche in der Ausgabe eines einzelnen Wertes resultieren.

Die meiste Zeit wirst Du die mathematische Operatoren in arithmetischen Prozessen im Bezug auf eine Zahlenmenge finden. Jedoch koennen diese Operatoren auch auf verschiedene Datentypen, inkl. Punkte und Vektoren angewendet werden.

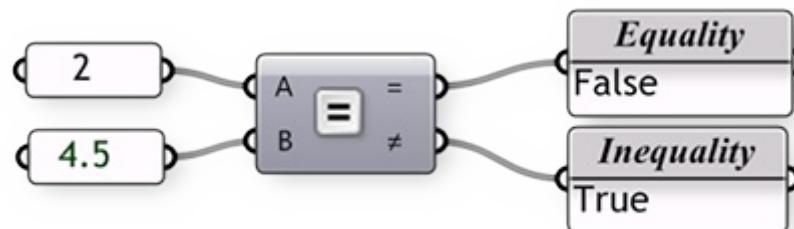


### 1.3.3.3. KONDITIONALE OPERATOREN

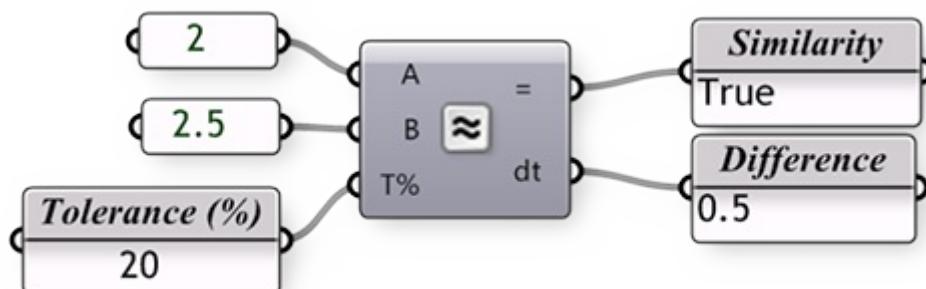
Fast jede Programmiersprache hat eine Methode um Konditionale Ausdruecke auszuwerten. In den meisten

Faelen wird der Programmierer ein Stueck Code entwerden, welches eine einfache Frage wie "Was waere wenn?" stellt. Was waere wenn die Flaeche einer Stockwerksumgrenzung einen realistischen Betrag ueberschreitet? Diese wichtigen Fragen stellen eine hoehere Ebene der Abstraktion da. Computerprogramme haben die Moeglichkeit auszuwerten "was waere wenn" und entsprechende Aktionen auf die Antwort der Frage folgen zu lassen. Las suns einen Blick auf ein einfaches Konditional werden, das ein Programm interpretieren wuerde: Wenn dass Objekt eine Kurve ist, loesche es. Das Stueck Code schaut zuerst auf das Objekt und stellt in einer boolschen Variable fest, ob es sich dabei um eine Kurve handelt. Der boolsche Wert ist "wahr" wenn das Objekt eineKurve ist oder "falsch" wenn das Objekt keine Kurve ist. Der zweite Teil des Ausdruckes fuehrt eine Aktion entsprechend dem Ergebnis des konditionalen Ausdrucks aus; in diesem Fall wird das Objekt geloescht, wenn es keine Kurve sein sollte. Dieser konditionale Ausdruck wird "If Statement"genannt. In diesem Kontext gibt es vier konditionale Operatoren (in der "Math/Operators" Unterkategorie), die Konditionale auswerten und boolsche Werte ausgeben.

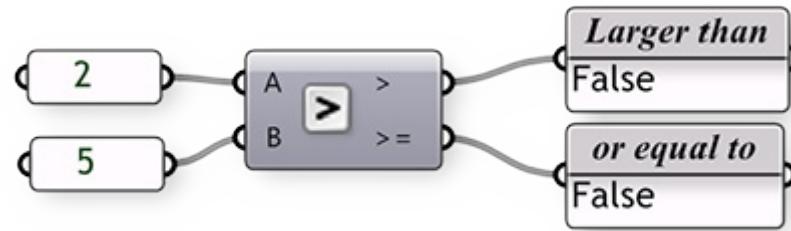
Die "Equality" Komponente nimmt zwei Listen und vergleicht die ersten Elemente der Liste Amit dem ersten element der Liste B. Wenn die beiden Werte die selben sind, wird "True" als boolscher Wert ausgegeben; auf der anderen Seite, wenn die beiden Werte ungleich sind, wird "Falsch" ausgegeben. Die Komponente rotiert durch die Liste in Uebereinstimmung mit dem "Data Matching" Algorithmus (als Standard ist die Einstellung "Longest List"). Es gibt zwei Ausgabeparameter fuer diese Komponente. Der erste gibt eine Liste mit boolschen Werten aus, die angibt welcher der Werte in der einen Liste mit einem in der anderen Liste uebereinstimmt. Der zweite Ausgabeparameter gibt eine Liste aus, die zeigt welche Werte nicht gleich den entsprechenden Werten in der zweiten Liste sind - oder eine invertierte Liste im Vergleich zum ersten Ausgabeparameter.



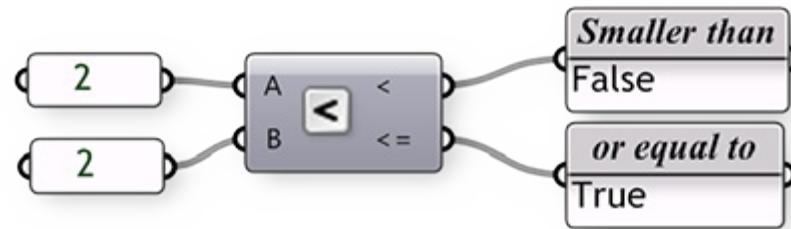
Die "Similarity" Komponente wertet die Daten von zwei Listen aus und prueft diese auf Aehnlichkeit der beiden Zahlen. Sie ist beinahe identisch mit dem Verhalten der "Equality" Komponente, jedoch mit einer Ausnahme: sie hat einen prozentualen Eingabeparameter, der es erlaubt ein Verhaeltnis der zulaessigen Abweichung zwischen Werten der Liste A und der Liste B anzugeben, ausserhalb derer Ungleichheit angenommen wird. Die "Similarity" Komponente hat außerdem einen Ausgabeparameter der in absoluten Werten angibt, welche Distanz zwischen den Werten der beiden Listen liegt.



Die "Larger Than" Komponente wird die Daten aus zwei Listen vergleichen und bestimmen, ob der erste Wert von Liste Agroesser sind als der erste Wert von Liste B. Die beiden Ausgabeparameter erlauben es Dir festzulegen ob Du die beiden Listen entsprechend der groesser als (>) oder groesser als oder gleich (>=) Funktion auswerten moechtest.



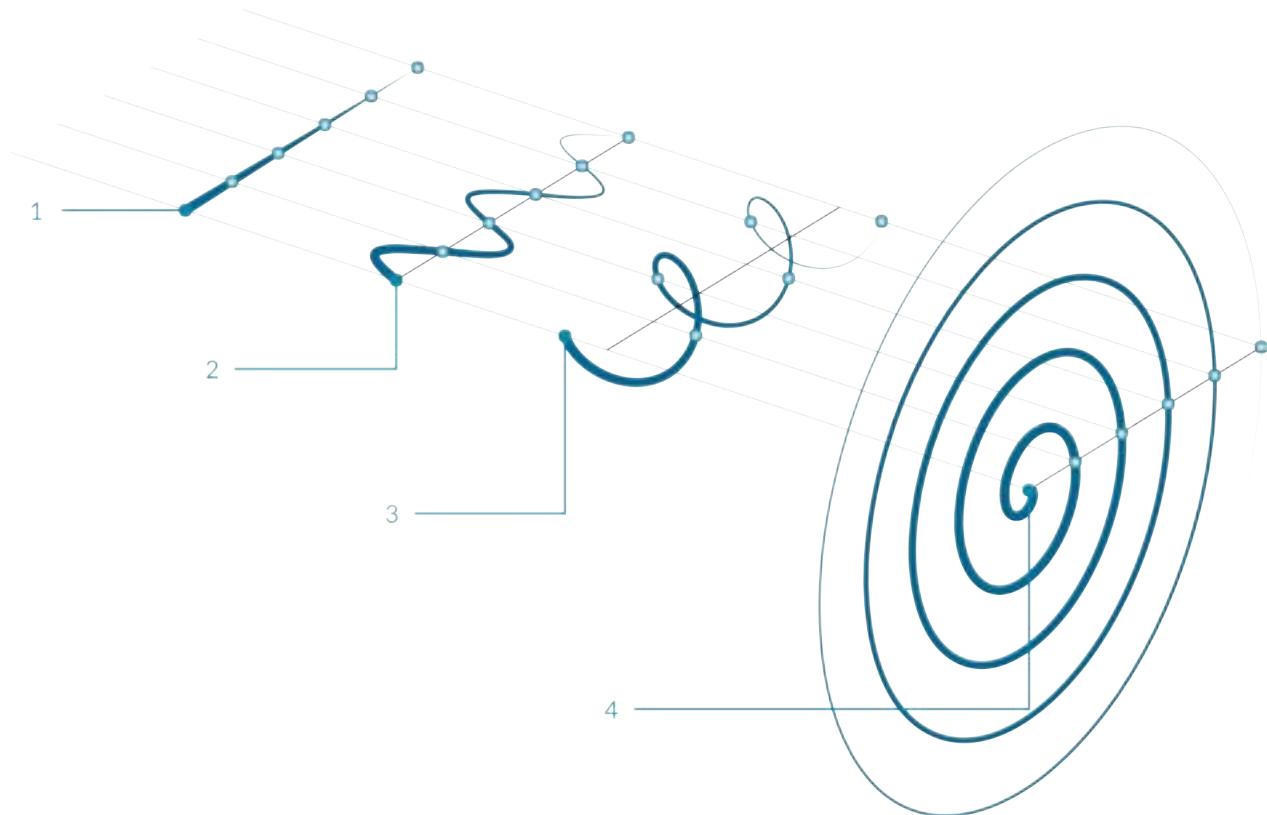
Die "Smaller Than" Komponente verhält sich entgegengesetzt zur "Larger Than" Komponente. Die "Smaller Than" Komponente bestimmt, ob die Werte einer Liste kleiner sind als die Werte einer Liste V und gibt die entsprechenden boolschen Werte aus. Ähnlich wie zuvor, kannst Du mit den beiden Ausgabeparametern wählen, ob Du die Listen entsprechend der kleiner als oder kleiner als ( $<$ ) oder gleich ( $\leq$ ) Funktion auswerten willst.



#### 1.3.3.4. TRIGONOMETRIE KOMPONENTEN

Beispieldateien für diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

Wir haben bereits gezeigt, dass wir eine "Expression" (oder "Evaluate") Komponente nutzen können um konditionale Ausdrücke, sowie algebraische Gleichungen auswerten zu können. Jedoch gibt es auch andere Wege um einfache Ausdrücke mit ein paar eingebauten trigonometrischen Funktionen zu berechnen. Wir können diese Funktionen nutzen, um periodische Phänomene zu beschreiben, wie beispielsweise sinusartige Wellenformen, wie Wellen im Ozean, Schallwellen und Lichtwellen.



## 1. Line

 $y(t) = 0$ 

## 2. Sine Curve

 $y(t) = \sin(t)$ 

## 3. Helix

 $x(t) = \cos(t)$  $y(t) = \sin(t)$  $z(t) = b(t)$ 

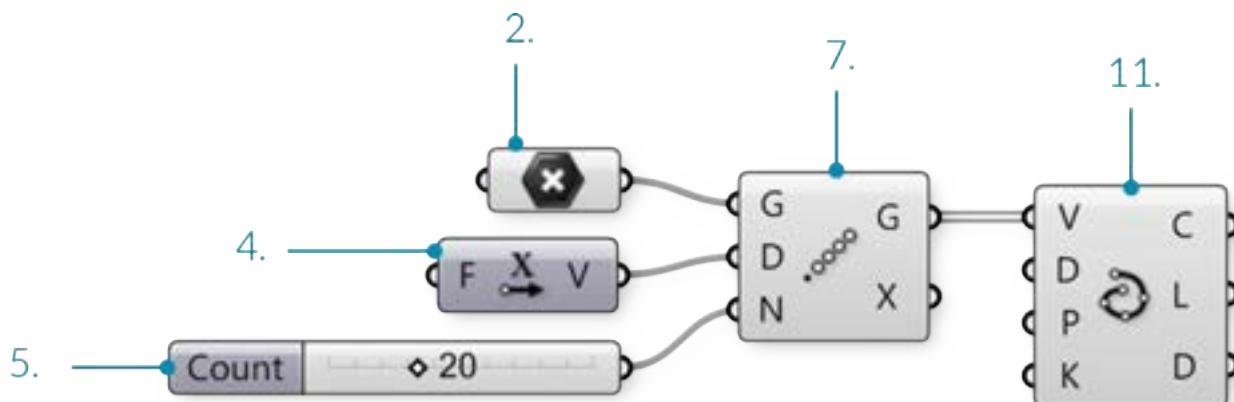
## 4. Spiral

 $x(t) = t * \cos(t)$  $y(t) = t * \sin(t)$ 

In diesem Beispiel werden wir Grasshopper benutzen um verschiedene trigonometrische Kurven mit den trigonomischen Funktionskomponenten aus dem "Math"-Reiter zu beschreiben:

01.	Tippe Strg+N (in Grasshopper) um eine neue Definition zu beginnen	
02.	<b>Params/Geometry/Point</b> – Ziehe einen <b>Point</b> Parameter auf die Leinwand	
03.	Rechtsklicke den <b>Point</b> Parameter und klicke "Set One Point" - waehle einen Punkt aus dem Rhinoansichtsfenster	
04.	<b>Vector/Vector/Unit X</b> – Ziehe eine <b>Unit X</b> Komponente auf die Leinwand	
05.	<b>Params/Input/Number Slider</b> – Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand	
06.	Doppelklicke auf den <b>Number Slider</b> und setze folgende Werte: Rounding: Integer Lower Limit: 10 Upper Limit: 40	

	Value: 20	
07.	Transform/Array/Linear Array – Ziehe eine <b>Linear Array</b> Komponente auf die Leinwand	
08.	Verbinde den Ausgabeparameter des <b>Point</b> Parameters mit dem Geometrie (G) Eingabeparameter der <b>Linear Array</b> Komponente	
09.	Verbinde den Einheitsvektor (V) Ausgabeparameter der <b>Unit X</b> Komponente mit dem Richtung (D) Eingabeparameter der <b>Linear Array</b> Komponente Du solltest eine Linie mit 20 Punkten entlang der x-Achse in Rhino sehen. Passe den Schieberegler an, um die Anzahl der Punkte in der Reihe zu veraendern.	
10.	Verbinde den <b>Number Slider</b> Ausgabeparameter mit dem Anzahl (N) Eingabeparameter der <b>Linear Array</b> Komponente	
11.	Curve/Spline/Interpolate – Ziehe eine <b>Interpolate Curve</b> Komponente auf die Leinwand	
12.	Verbinde den Geometrie (G) Ausgabeparameter der <b>Linear Array</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Interpolate Curve</b> Komponente	



Wir habe gerade eine Linie erzeugt, indem wir eine Reihe von Punkten zu einer Kurve verbunden haben.  
Lass uns nun versuchen die Trigonometriekomponenten in Grasshopper zu nutzen, um die Kurve zu veraendern:

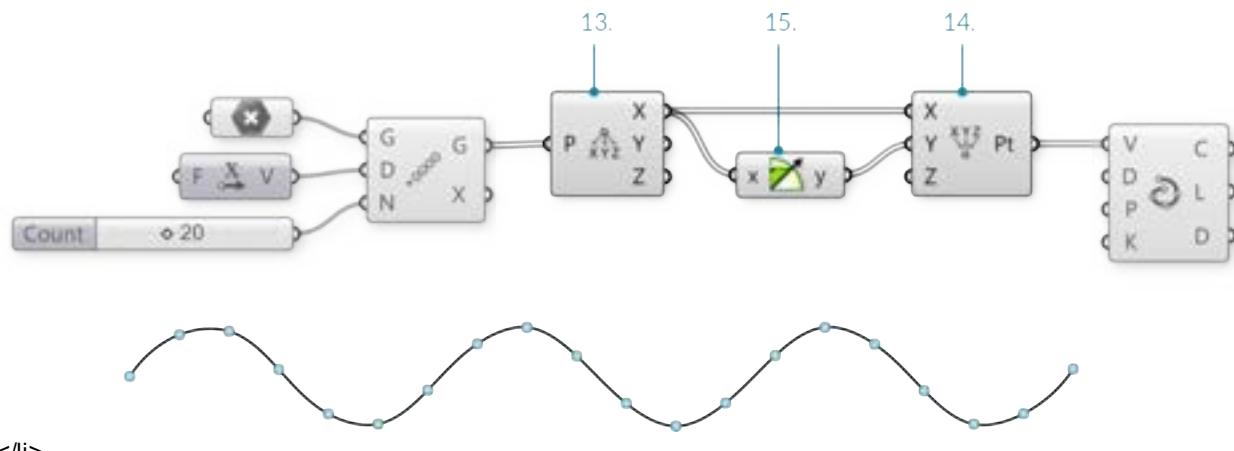
13.	Vector/Point/Deconstruct – Ziehe eine <b>Deconstruct</b> Komponente auf die Leinwand	
14.	Vector/Point/Construct Point - Ziehe eine <b>Construct Point</b> Komponente auf die Leinwand	
15.	Maths/Trig/Sine - Ziehe eine <b>Sine</b> Komponente auf die Leinwand	
	Entferne das Kabel vom Eckpunkte (V) Eingabeparameter der <b>Interpolate Curve</b> Komponente. Du kannst Kabel entfernen, indem Du mit gedrueckter Strg-Taste entlangziehest	

	oder indem Du auf den Eingabeparameter rechtsklickst und "Disconnect" auswaehlst.	
17.	Verbinde den Geometrie (G) Ausgabeparameter der <b>Linear Array</b> Komponente mit dem Punkt (P) Eingabeparameter der <b>Deconstruct</b> Komponente	
18.	Verbinde den Punkt X (X) Ausgabeparameter der <b>Deconstruct</b> Komponente mit dem X Koordinate (X)	

Eingabeparameter der **Construct Point** Komponente || |19.| Verbinde ein zweites Kabel vom Punkt X (X) Ausgabeparameter der **Deconstruct** Komponente mit dem Wert (x) Eingabeparameter der **Sine** Komponente || |20.| Verbinde den Ergebnis (y) Ausgabeparameter der **Sine** Komponente mit dem Y Koordinate (Y) Eingabeparameter der **Construct Point** Komponente

Wir haben nun unsere Punkte mit den gleichen x-Werten rekonstruiert und dabei die y-Werte mit der Sinuskurve modifiziert.

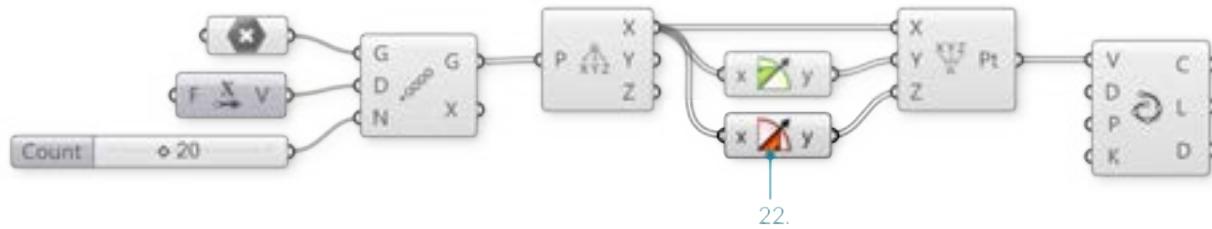
|| |21.| Verbinde den Punkt (Pt) Ausgabeparameter der **Construct Point** Komponente mit dem Eckpunkte (V) Eingabeparameter der **Interpolate** Komponente |||



</li>

Du solltest nun eine Sinuskurve entlang der x-Achse in Rhino sehen

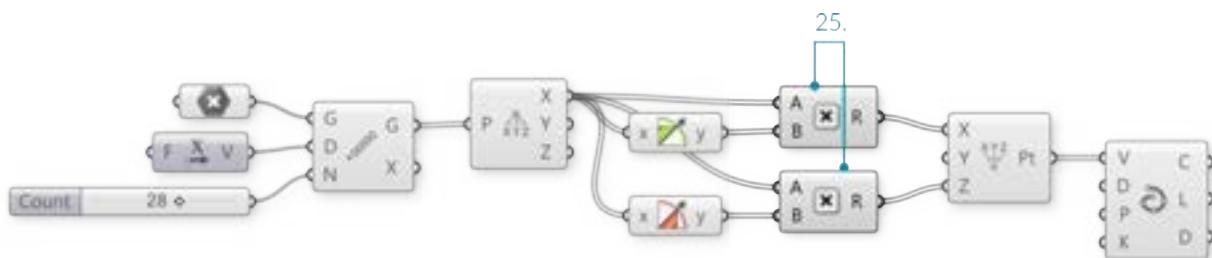
22.	Maths/Trig/Cosine – Ziehe eine <b>Cosine</b> Komponente auf die Leinwand	
23.	Verbinde ein drittes Kabel mit dem Punkt X (X) Ausgabeparameter der <b>Deconstruct</b> Komponente mit dem Wert (x) Eingabeparameter der <b>Cosine</b> Komponente	
24.	Verbinde den Ergebnis (y) Ausgabeparameter der <b>Cosine</b> Komponente mit dem Z Koordinate (Z) Eingabeparameter der <b>Construct Point</b> Komponente	



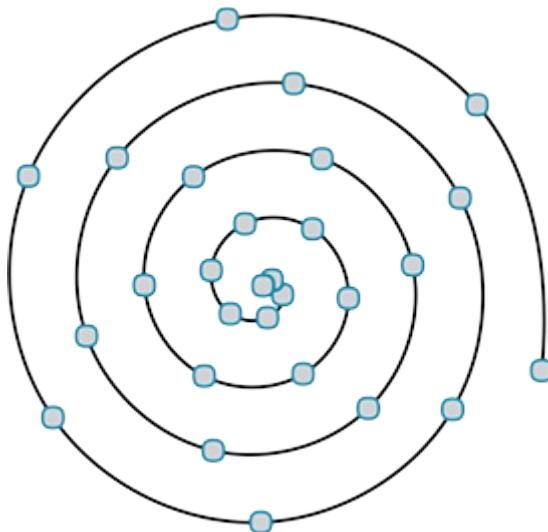
&lt;/li&gt;

Wir haben nun eine dreidimensionale Helix

25.	<b>Maths/Operators/Multiplication</b> – Ziehe zwei <b>Multiplication</b> Komponenten auf die Leinwand	
26.	Verbinde Kabel von den Punkt X (X) Ausgabeparameter der <b>Deconstruct</b> Komponente mit den (A) Eingabeparametern jeder <b>Multiplication</b> Komponente	
27.	Verbinde den Ergebnis (y) Ausgabeparameter der <b>Sine</b> Komponente mit dem (B) Eingabeparameter der ersten <b>Multiplication</b> Komponente	
28.	Verbinde den Ergebnis (y) Ausgabeparameter der <b>Cosine</b> Komponente mit dem (B) Eingabeparameter der zweiten <b>Multiplication</b> Komponente	
29.	Verbinde die Kabel des Y Koordinate (Y) Eingabeparameter mit dem Eingabeparameter der <b>Construct Point</b> Komponente	
30.	Verbinde den Ergebnis (R) Ausgabeparameter der ersten <b>Multiplication</b> Komponente mit dem X Koordinate (X) Eingabeparameter der <b>Construct Point</b> Komponente	
31.	Verbinde den Ergebnis (R) Ausgabeparameter der zweiten <b>Multiplication</b> Komponente mit dem Z Koordinate (Z) Eingabeparameter der <b>Construct Point</b> Komponente	



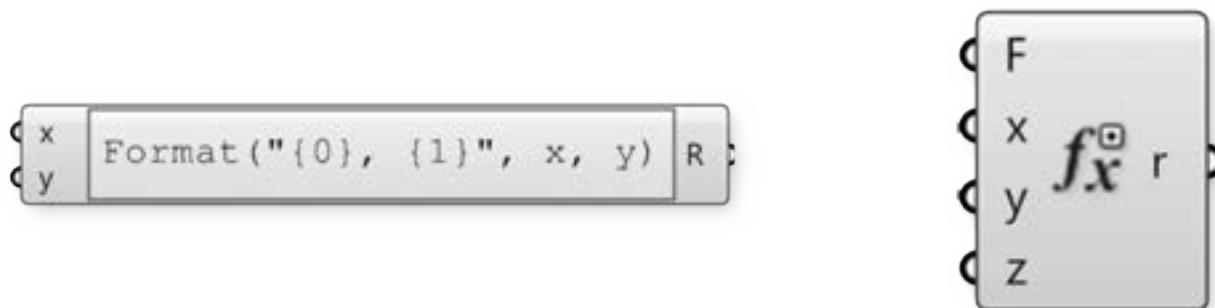
Du solltest nun eine spiralförmige Kurve sehen



### 1.3.3.5. EXPRESSIONS

Beispieldateien fuer diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

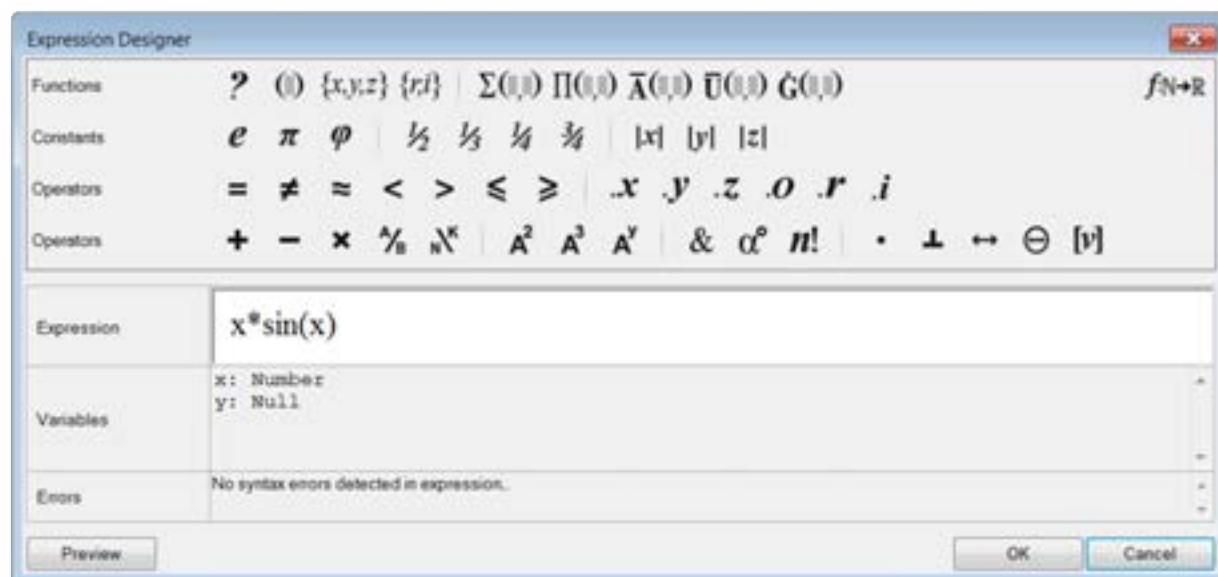
Die "Expression" Komponente (und ihr Bruder die "Evaluate" Komponente) sind sehr flexible Werkzeuge; dazu ist zu sagen, dass die fuer eine Vielzahl von verschiedenen Anwendungen genutzt werden koennen. Wir koennen die "Expression" (oder die "Evaluate" Komponente) heranziehen um mathematische Algorithmen auszuwerten und numerische Daten als Ausgabeparameter zurueckzugeben.



In den folgenden Beispielen werden wir uns mathematische Spiralen ansehen, die in der Natur gefunden werden koennen und wie wir mit wenigen Funktionskomponenten aehnliche Muster in Grasshopper erzeugen koennen. Wir werden in unsere trigonometrische Kurven Definition als Ausgangspunkt waehlen.

01.	Oeffne die trigonometrische Kurven Definition aus dem vorherigen Beispiel	
02.	Loesche die <b>Sine</b> , <b>Cosine</b> , <b>Multiplication</b> , und <b>Interpolate</b> Komponenten	
03.	<b>Params/Input/Number Slider</b> – Ziehe einen Schieberegler auf die Leinwand	
04.	Doppelklicke auf den <b>Number Slider</b> und setze folgende Werte: Rounding: Float Lower Limit: 0.000 Upper Limit: 1.000 Value: 1.000	
05.	Verbinde den <b>Number Slider</b> mit dem Faktor (F) Eingabeparameter der "Unit X" Komponente. Die Schieberegler erlauben es Dir die Distanz zwischen den Punkten der Reihe zu veraendern.	

06.	<b>Maths/Script/Expression</b> – Ziehe zwei <b>Expression</b> Komponenten auf die Leinwand
07.	Doppelklicke die erste <b>Expression</b> Komponente um den "Expression Editor" zu öffnen und ändere den Ausdruck in: $x*\sin(x)$
08.	Doppelklicke die zweite <b>Expression</b> Komponente um den "Expression Editor" zu öffnen und ändere den Ausdruck in: $x*\cos(x)$

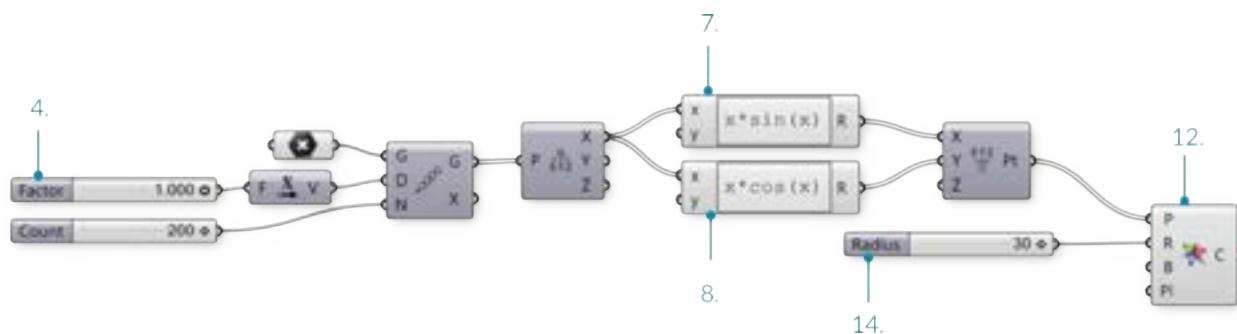


Doppelklicke die "Expression" Komponente um den Grasshopper "Expression Editor" zu oeffnen

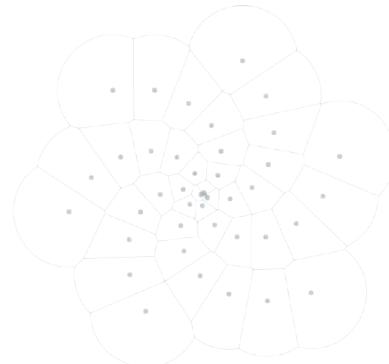
09.	Verbinde die beiden Kabel des Punkt X (X) Ausgabeparameters der <b>Deconstruct</b> Komponente mit dem Variable x (x) Eingabeparameter jeder <b>Expression</b> Komponente
10.	Verbinde den Ergebnis (R) Ausgabeparameter mit der ersten <b>Expression</b> Komponente mit dem X Koordinate (X) Eingabeparameter der <b>Construct Point</b> Komponente
11.	Verbinde den Ergebnis (R) Ausgabeparameter der zweiten <b>Expression</b> Komponente mit dem Y Koordinate (Y) Eingabeparameter der <b>Construct Point</b> Komponente Wir haben die trigonometrischen Funktionen und Multiplikationsoperatoren mit den "Expression" Komponenten ersetzt und dadurch eine effizientere Definition erzeugt.
12.	<b>Mesh/Triangulation/Voronoi</b> – Ziehe eine <b>Voronoi</b> Komponente auf die Leinwand
13.	<b>Params/Input/Number Slider</b> – Ziehe einen <b>Number Slider</b> auf die Leinwand
14.	Doppelklicke auf den <b>Number Slider</b> und setze folgende Werte: Rounding: Integer Lower Limit: 1 Upper Limit: 30 Value: 30
15.	Verbinde den <b>Number Slider</b> mit dem Radius (R) Eingabeparameter der <b>Voronoi</b> Komponente

16.

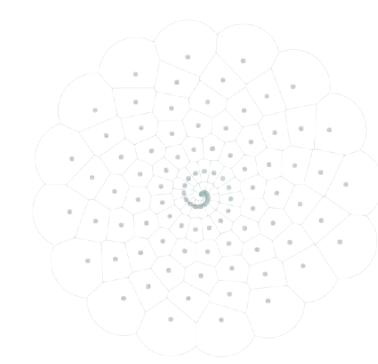
Verbinde den Punkt (Pt) Ausgabeparameter der **Construct Point** Komponente mit dem Punkte (P) Eingabeparameter der **Voronoi** Komponente



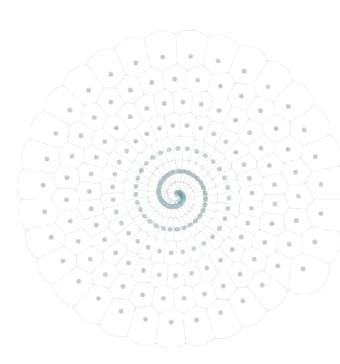
Du kannst verschiedene Vronoimuster durch die Manipulation der Werte fuer "Factor", "Count" und "Radius" mit den Schiebereglern erzeugen. Unterhalb sind drei Beispiele:



1

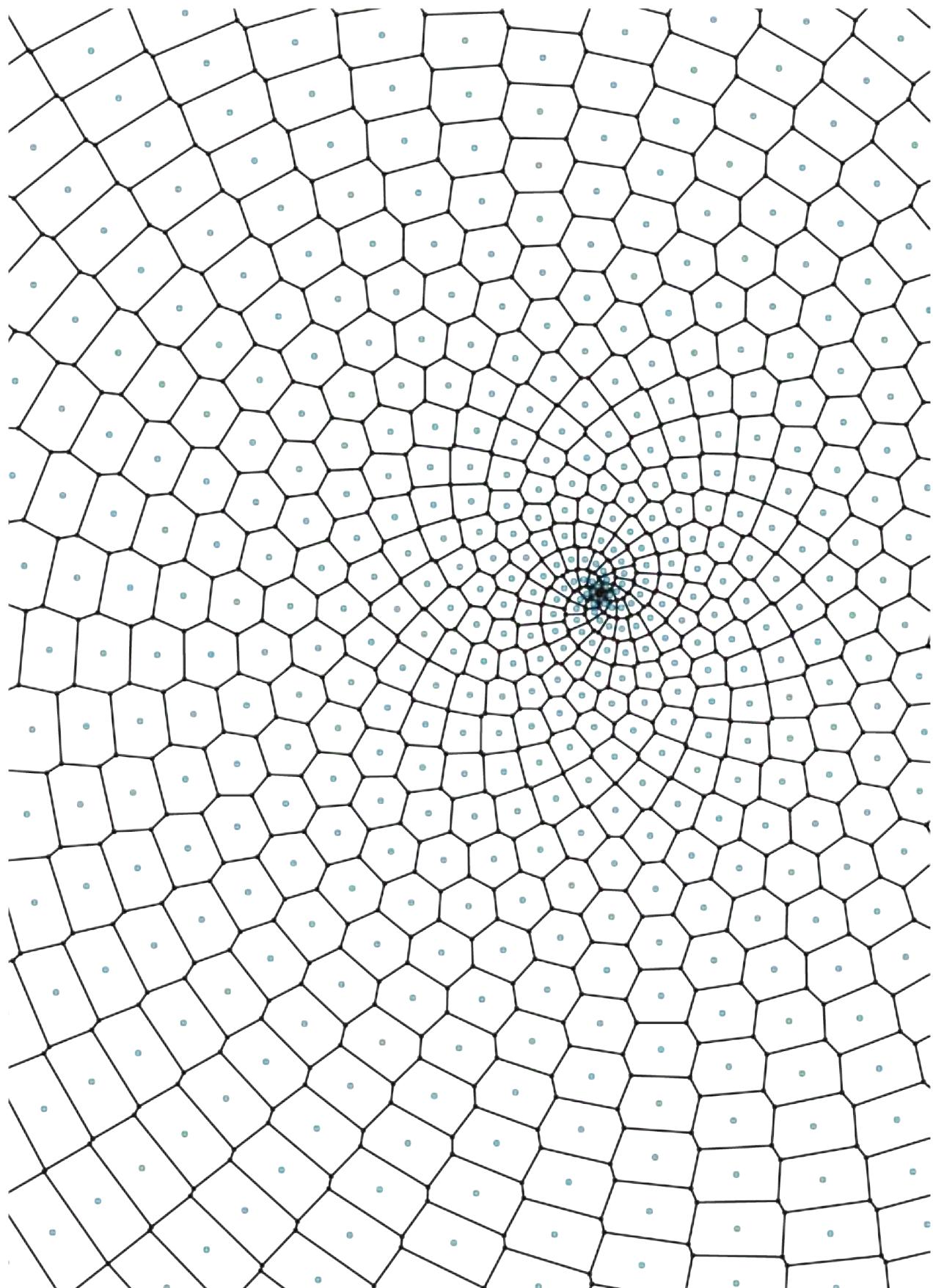


2



3

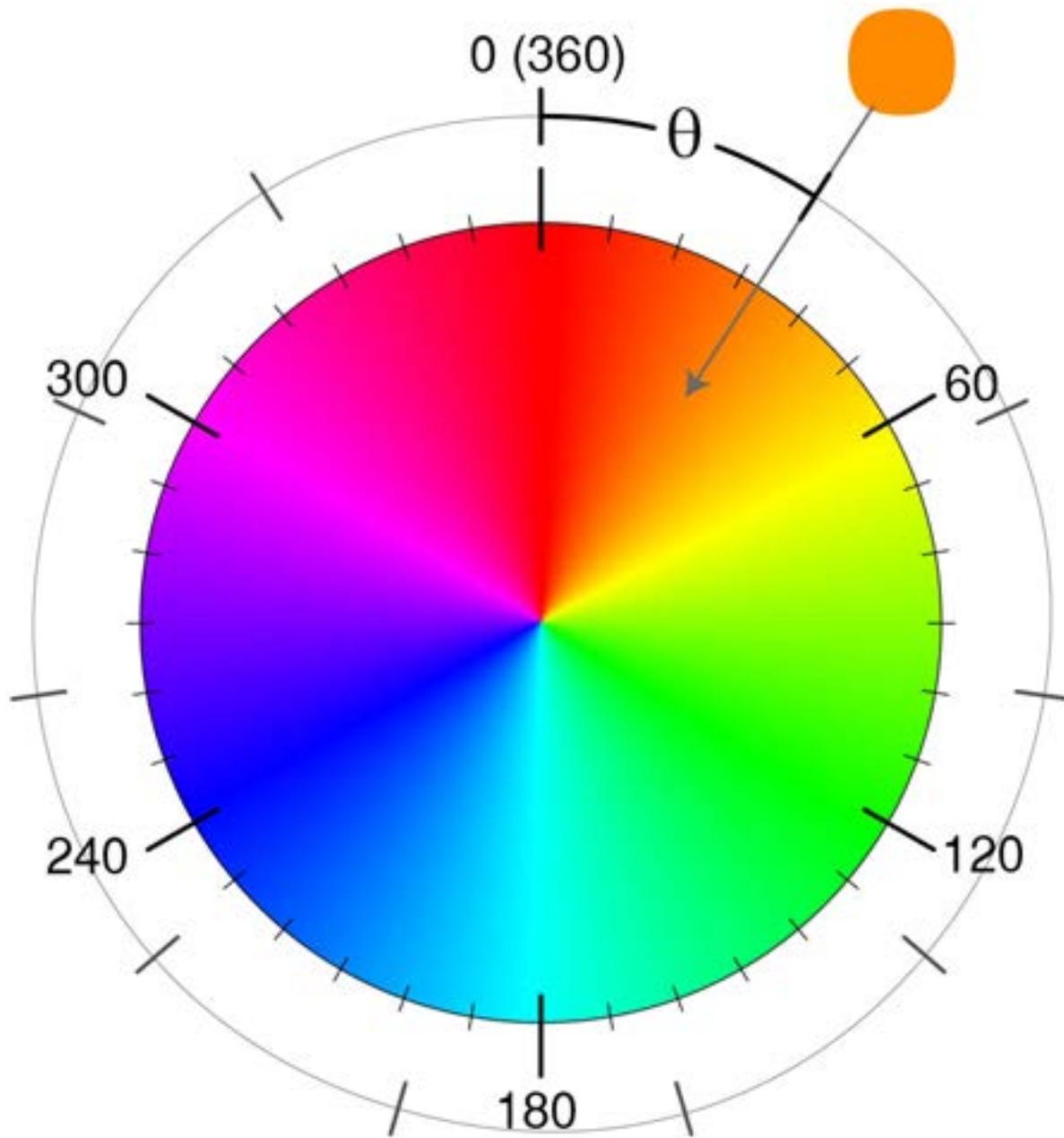
1. Factor = 1.000, Radius = 15
2. Factor = 0.400, Radius = 10
3. Factor = 0.200, Radius = 7



### 1.3.4. Domaenen & Farben

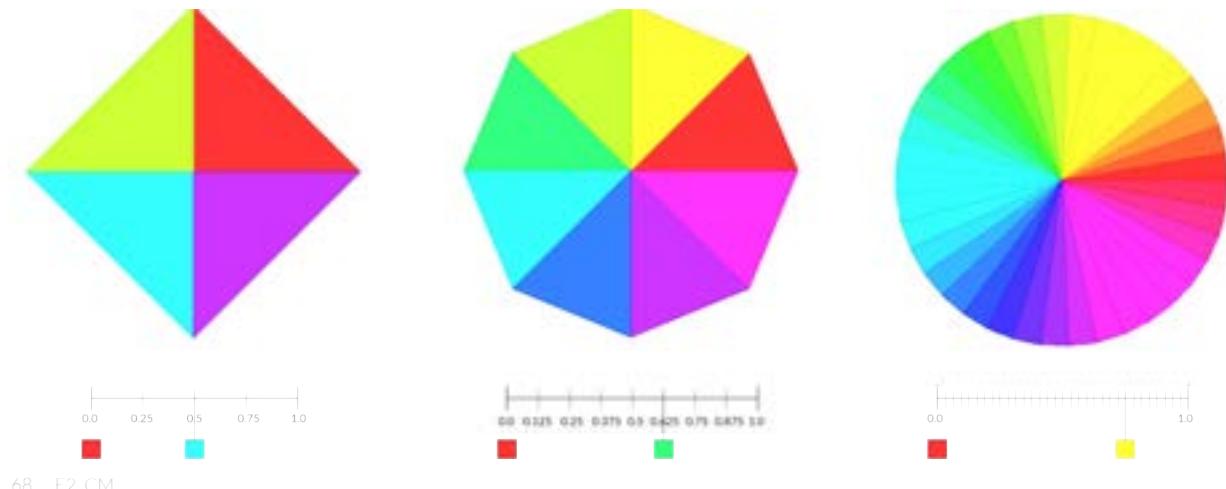
Beispieldateien fuer diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

**Das Farbrad ist ein Modell um Farben basierend auf ihrem Farbton zu organisieren. In Grasshopper werden Farben durch einen Farbton zwischen 0.0 to 1.0 beschrieben. Domaenen werden verwendet um ein Spektrum an moeglichen Werten zwischen einer Untergrenze (A) und einer Obergrenze (B) anzugeben.**



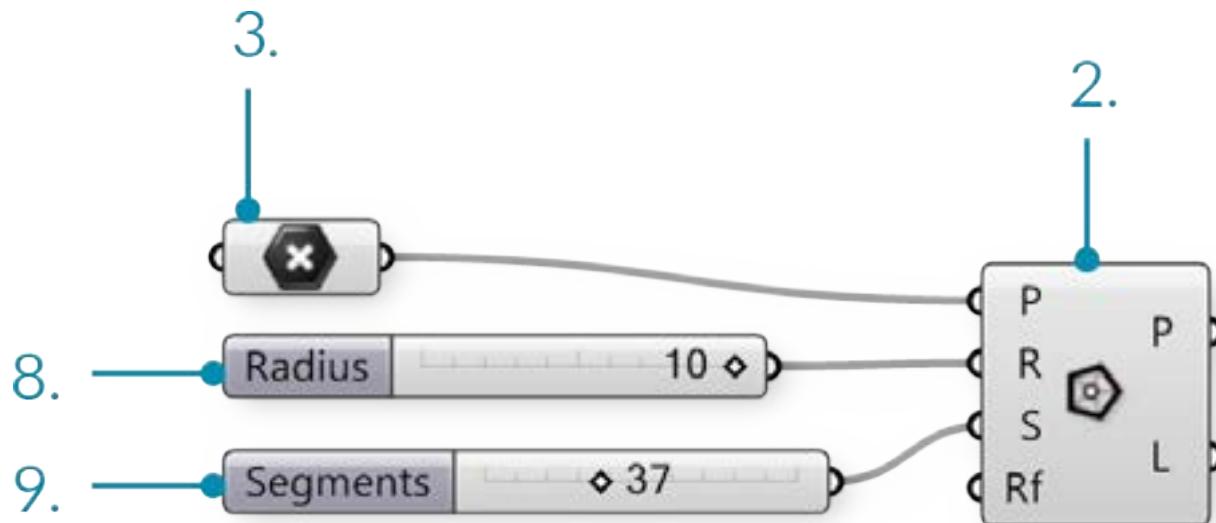
Auf dem Farbrad entspricht der Farbton dem Winkel. Grasshopper hat diese 0-360 Domaene genommen und auf eine zwischen null und eins uebertragen.

Indem die Farbdomäne (0.0 to 1.0) mit der Anzahl an gewünschten Segmenten unterteilt wird, kann man wor dem Wert des Farbtons dem entsprechenden Segment zuordnen um ein Farbrad zu erzeugen.

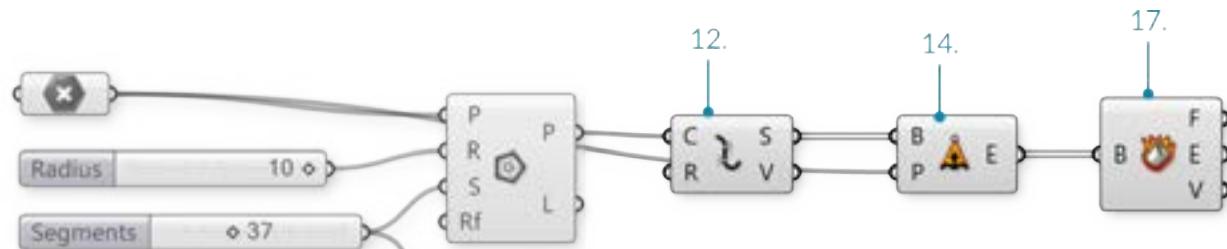


In diesem Beispiel werden wir Grasshopper's Domaenen und Farbkomponenten heranziehen um ein Farbrad mit variabler Anzahl von Segmenten zu erzeugen.

01.	Tippe Ctrl+N (in Grasshopper) um eine neue Definition zu erzeugen	
02.	<b>Curve/Primitive/Polygon</b> – Ziehe eine <b>Polygon</b> Komponente auf die Leinwand	
03.	<b>Params/Geometry/Point</b> – Ziehe einen <b>Point</b> Parameter auf die Leinwand	
04.	Rechtsklicke auf die <b>Point</b> Komponente und waehle "Set one point"	
05.	Waehle einen Punkt aus dem Modellraum.	
06.	Verbinde den <b>Point</b> Parameter (Basispunkt) mit dem Ebene (P) Eingabeparameter der <b>Polygon</b> Komponente	
07.	<b>Params/Input/Number Sliders</b> – Ziehe zwei <b>Number Sliders</b> auf die Leinwand	
08.	Doppelklicke den ersten <b>Number Sliders</b> und setze folgende Werte: Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10	
09.	Doppelklicke den zweiten <b>Number Sliders</b> und setze folgende Werte: Rounding: Integers Lower Limit: 0 Upper Limit: 100 Value: 37	
10.	Verbinde den <b>Number Slider</b> (Radius) mit dem Radius (R) Eingabeparameter der <b>Polygon</b> Komponente Wenn Du den Schieberegler mit einer Komponente verbindest, aendert er automatisch seinen Namen in den des Eingabeparameters mit dem er verbunden wird.	
11.	Verbinde den <b>Number Slider</b> (Segmente) mit dem segmente (S) Eingabeparameter der <b>Polygon</b> Komponente	

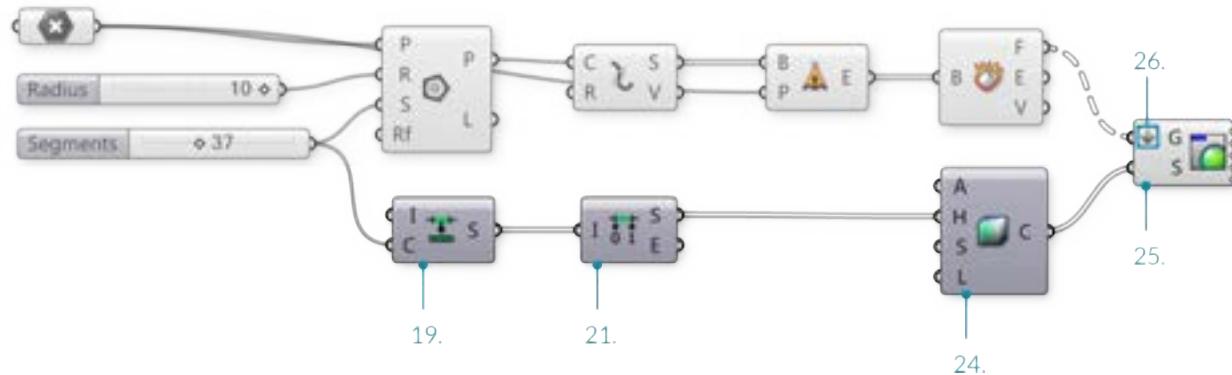


12.	<b>Curve/Util/Explode</b> – Ziehe eine <b>Explode</b> Komponente auf die Leinwand.	
13.	Verbinde den Polygon (P) Ausgabeparameter der <b>Polygon</b> Komponente mit dem Kurve (C) Eingabeparameter der <b>Explode</b> Komponente	
14.	<b>Surface/Freeform/Extrude Point</b> – Ziehe eine <b>Extrude Point</b> Komponente auf die Leinwand	
15.	Verbinde den Segmente (S) Ausgabeparameter der <b>Explode</b> Komponente mit dem Basis (B) Eingabeparameter von <b>Extrude Point</b>	
16.	Verbinde den <b>Point</b> Parameter (Basispunkt) mit dem Extrusionsspitze (P) Eingabeparameter der <b>Extrude Point</b> Komponente	
17.	<b>Surface/Analysis/Deconstruct Brep</b> – Ziehe eine <b>Deconstruct Brep</b> Komponente auf die Leinwand	
18.	Verbinde den Extrusion (E) Ausgabeparameter der <b>Extrude Point</b> Komponente mit der <b>Deconstruct Brep</b> (B) Komponente	



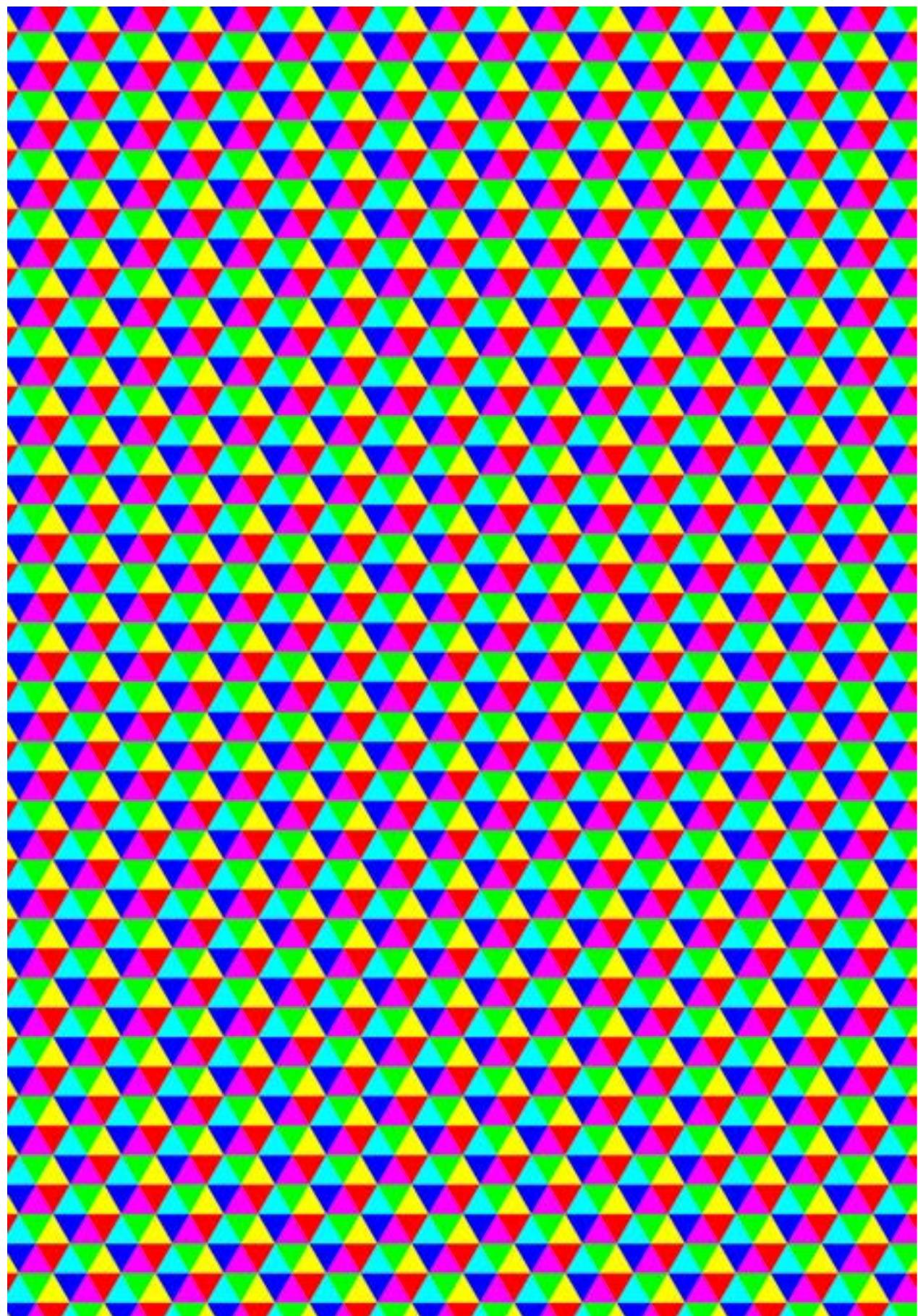
19.	<b>Maths/Domain/Divide Domain</b> – Ziehe eine <b>Divide Domain</b> Komponente auf die Leinwand Die Basisdomäne (I) ist automatisch zwischen 0.0-1.0, was wir fuer diese Uebung benoetigen	
20.	Verbinde den <b>Number Slider</b> (Segmente) mit dem Anzahl (C) Eingabeparameter der <b>Divide Domain</b> Komponente	
21.	<b>Math/Domain/Deconstruct Domain</b> – Ziehe eine <b>Deconstruct Domain</b> Komponente auf die Leinwand	

22.	Verbinde den Segmente (S) Ausgabeparameter der <b>Divide Domain</b> Komponente mit dem Domaene (I) Eingabeparameter der <b>Deconstruct Domain</b> Komponente	
23.	<b>Display/Colour/Colour HSL</b> – Ziehe eine <b>Colour HSL</b> Komponente auf die Leinwand	
24.	Verbinde den Start (S) Ausgabeparameter der <b>Deconstruct Domain</b> Komponente mit dem Farbton (H) Eingabeparameter der <b>Colour HSL</b> Komponente	
25.	<b>Display/Preview/Custom Preview</b> – Ziehe eine <b>Custom Preview</b> Komponente auf die Leinwand	
26.	Rechtsklicke auf den Geometrie (G) Eingabeparameter der <b>Custom Preview</b> Komponente und waehle "Flatten" Siehe 1-4 gestalten mit Datenbaeumen fuer Details	
27.	Verbinde den Seitenflaechen (F) Ausgabeparameter der <b>Deconstruct Brep</b> Komponente mit dem Geometrie(G) Eingabeparameter der <b>Custom Preview</b> Komponente	
28.	Verbinde den Farb (C) Ausgabeparameter der <b>Colour HSL</b> Komponente mit dem Schattierung (S) Eingabeparameter der <b>Custom Preview</b> Komponente	



Fuer verschiedene Farbeffekte, versueche die "Deconstruct Domain" Komponente mit den Saettigungs (S) oder Leuchtdichte (L) Eingabeparametern der "Colour HSL" Komponente zu verbinden.





## 1.3.5. Boolscche & Logische Operatoren

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

### 1.3.5.1. BOOLSCHEN OPERATOREN

Nummerische Variablen können eine ganze Bandbreite verschiedener Zahlen speichern. Boolscche Variablen können lediglich zwei Werte speichern auf die wir uns mit Ja und Nein, Wahr und Falsch oder 1 und 0 beziehen. Offensichtlich werden wir niemals boolscche Werte heranziehen um Kalkulationen durchzuführen, da ihre Werte so begrenzt sind. Wir nutzen boolscche Werte um Konditionale auszuwerten.



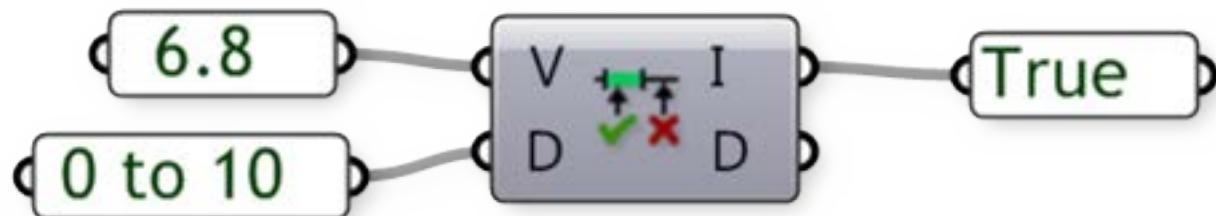
Boolscche Parameter

In Grasshopper werden boolscche Werte auf verschiedene Arten genutzt. Der boolscche Parameter ist ein Container für einzelne oder mehrere boolscche Werte, während der "Boolean Toggle" ermöglicht es schnell zwischen einzelnen Werten für wahr und falsch umzuschalten.



Boolean Toggle - doppelklicke um den boolscchen Wert des Schalters zwischen wahr und falsch umzuschalten

Grasshopper hat auch Objekte mit denen Konditionale getestet werden können und deren Ausgabe boolscche Werte sind. Beispielsweise die "Includes" Komponente erlaubt es einen nummerischen Wert auf seine Präsenz in einer Domäne zu testen.



Die "Includes" Komponente testet ob die Zahl 6.8 in einer Domäne von 0 bis 10 enthalten ist. Sie gibt einen boolscchen Wert für wahr zurück.

### 1.3.5.2. LOGISCHE OPERATOREN

Logische Operatoren arbeiten meistens mit boolscchen Werten und sie sind wirklich ziemlich logisch. Wie Du Dich erinnern kannst, können boolscche Werte nur zwei verschiedene Werte annehmen. Die boolscche Mathematik wurde von George Boole (1815-1864) entwickelt und ist heute Kernbestandteil der gesamten digitalen Industrie. Boolscche Algebra liefert uns die Werkzeuge mit welchen wir Datensätze analysieren, vergleichen und beschreiben können. Obwohl Boole ursprünglich sechs boolscche Operatoren beschrieb, werden wir nur drei

davon genauer unter die Lupe nehmen:

1. Not
2. And
3. Or

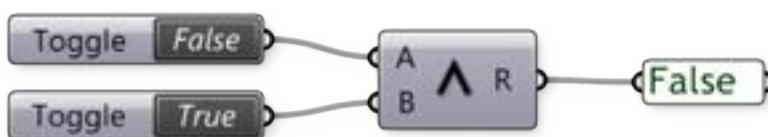
Der Not Operator ist eine Seltsamkeit unter den Operatoren, da er keine zwei Eingabewerte benoetigt. Stattdessen invertiert er einfach den Wert, den er bekommt. Stell Dir vor wir haben ein Skript, welches die Existenz von einer Menge Blockdefinitionen in Rhino ueberprueft. Wenn eine Blockdefinition nicht existiert, wollen wir den Nutzer informieren und das Skript abbrechen.



Der Grasshopper Not Operator (Gate)

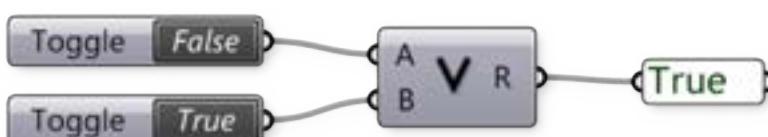
"And" und "or" nehmen jeweils zwei Argumente. Der "and" Operator wertet als "wahr" aus, wenn beide eingegebenen Argumente "wahr" sind. Der "or" Operator ist schon damit gluecklich, wenn ein einzelner Wert "wahr" ist.

Wie Du sehen kannst, ist das Problem mit den logischen Operatoren nicht die Theorie, sondern was passiert wenn Du eine Menge davon benoetigst um etwas auszuwerten. Sobald man sie zusammenstrickt, hat man schnell einen unuebersichtlichen Code; nicht davon zu sprechen, wenn man festlegen muss, welcher Operator Vorrang hat.



A	B	Result
True	True	True
True	False	False
False	True	False
False	False	False

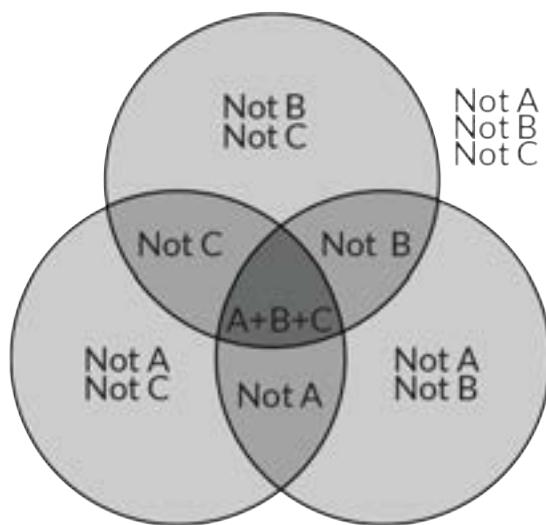
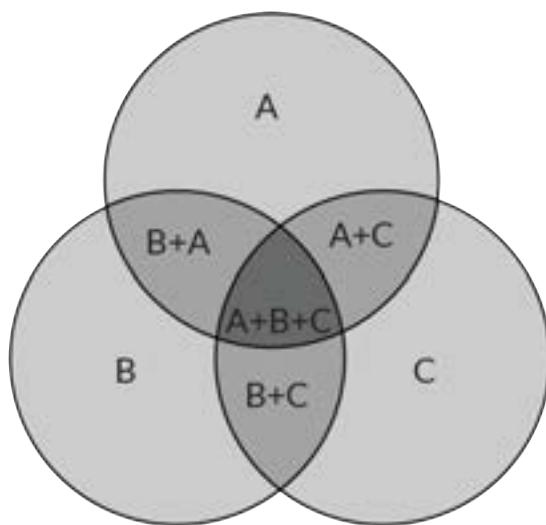
Der Grasshopper And Operator (gate)



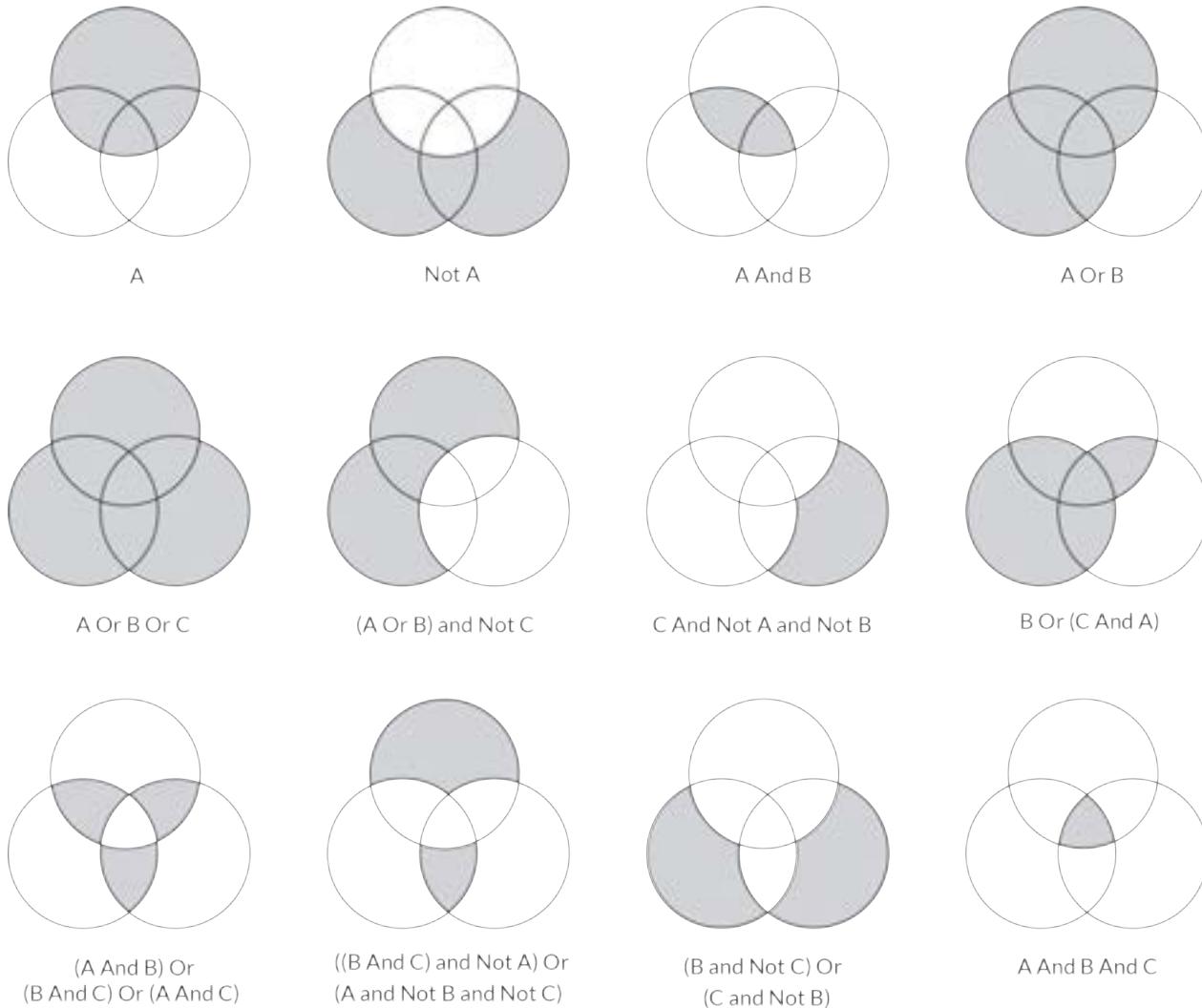
A	B	Result
True	True	True
True	False	True
False	True	True
False	False	False

Der Grasshopper Or Operator (gate)

Ein guter Weg um Deine eigene boolsche Logik zu trainieren ist die Verwendung von Venn Diagramme. Ein Venn Diagramm ist eine graphische Repraesentation einer boolschen Menge, in der jede Region eine Untermenge von Werten enthaelt, die eine gemeinsame Eigenschaft miteinander teilen. das beruehteste ist das Drei-Kreis-Diagramm:

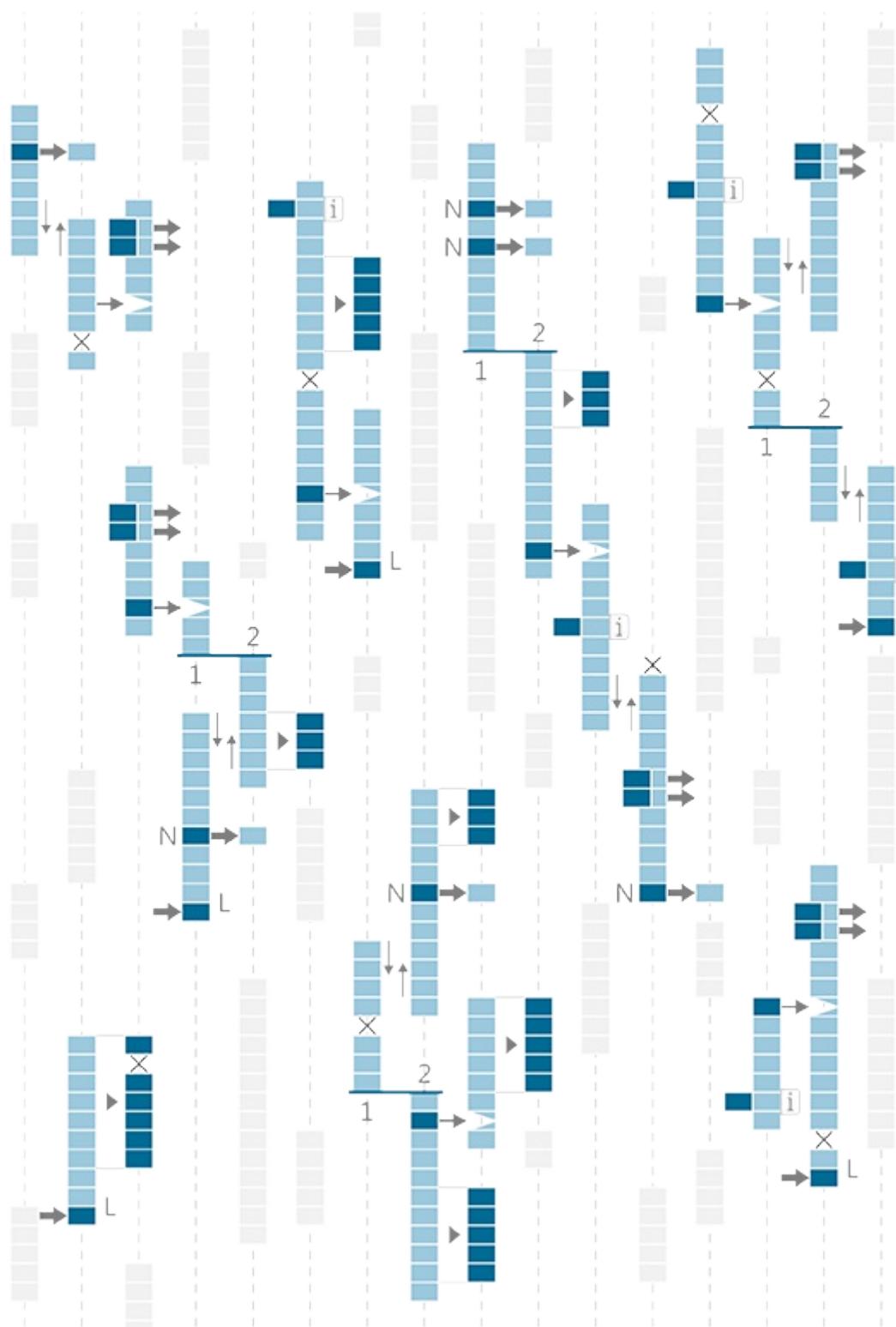


Jede Kreisregion enthaelt alle Werte, die zu einer Menge gehoeren; der obere Kreis zum Beispiel beschreibt eine Menge  $\{A\}$ . Jeder Wert innerhalb des Kreises wird fuer  $\{A\}$  als "wahr" ausgewertet und jeder Wert ausserhalb wird fuer  $\{A\}$  als "falsch"ausgewertet. Indem die Regionen eingefärbt werden, koennen wir die boolsche Auswertung in programmiertem Code nachahmen:



## 1.4. Gestaltung mit Listen

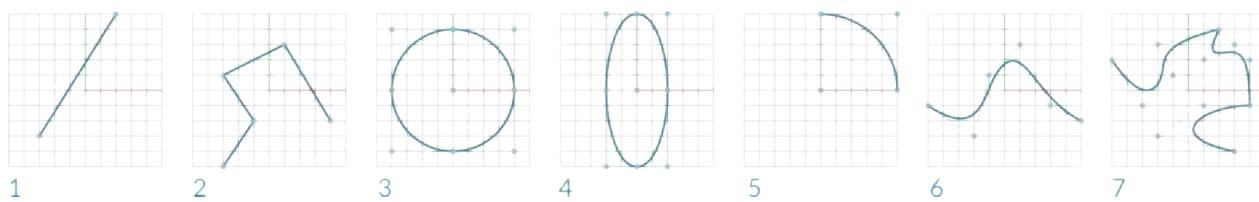
Eine der maechtigsten Eigenschaften von Grasshopper ist die Faeigkeit schnell Listen von Daten zu erstellen und zu bearbeiten. Dieses Kapitel wird erklaeren, wie Datenlisten erstellt, veraendert und visulisiert werden koennen.



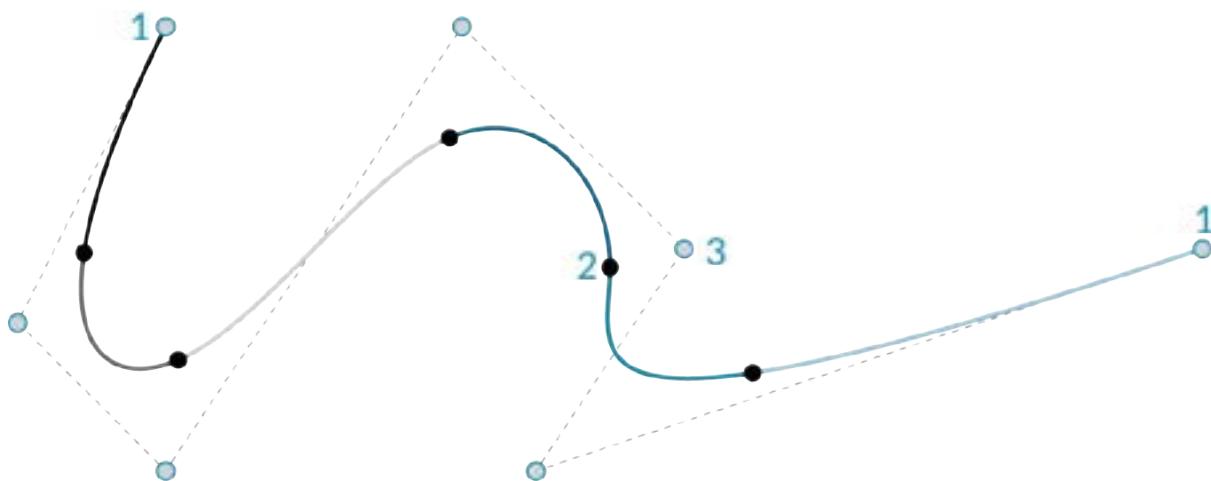
### ### 1.4.1. KURVENGEOMETRIEN

**NURBS** (non-uniform rational B-splines) sind mathematische Repraesentationen, die akurat jede Form, von einer 2D Linie, einem Kreis, Boden oder einer Kiste, bis hin zur komplexesten, organischen 3D Freiformflaeche darstellen koennen. Sie koennen in jedem Prozess von Illustration und Animation bis zur Herstellung verwendet werden, da NURBS Modelle flexibel und akurat sind.

Da Kurven geometrische Objekte sind, besitzen sie eine Anzahl von Eigenschaften und Charakteristika, die genutzt werden koennen um sie zu beschreiben oder zu analysieren. Zum Beispiel hat jede Kurve eine Startkoordinate und eine Endkoordinate. Wenn die Distanz zwischen den beiden Koordinaten null ergibt, ist die Kurve geschlossen. Ebenso hat jede Kurve eine Anzahl an Kontrollpunkten. Wenn all diese Punkte in der selben Ebene angeordnet sind, ist die gesamte Kurve planar. Einige Eigenschaften beziehen sich auf die Kurve als Ganzes, waehrend andere sich nur auf bestimmte Punkte auf der Kurve beziehen. Beispielsweise ist die Ebenheit eine globale Eigenschaft, waehrend Tangentenvektoren eine lokale Eigenschaft sind. Einige Eigenschaften sind auch nur auf bestimmte Kurventypen anzuwenden. So weit habrn wir einigen von Grasshopper's primitiven Kurvenkomponenten besprochen: Linien, Kreise, Ellipsen und Boegen.



- 1. Linie
- 2. Polylinie
- 3. Kreis
- 4. Ellipse
- 5. Bogen
- 6. NURBS Kurve
- 7. Polykurve



- 1. Endpunkt
- 2. Bearbeitungspunkt
- 3. Kontrollpunkt

#### 1.4.1.1. NURBS KURVEN

**Degree:** Der Grad ist eine positive ganze Zahl. Diese Zahl ist ueblicherweise 1, 2, 3 oder 5, kann aber jede positive ganze Zahl annehmen. Der Grad einer Kurve legt die Reichweite des Einflusses der Kontrollpunkte auf die Kurve

fest; wobei je höher der Grad, desto größer der Einflussbereich. NURBS Linien und Polylinien haben den Grad 2, und die meisten Freiformkurven haben den Grad 3 oder 5.

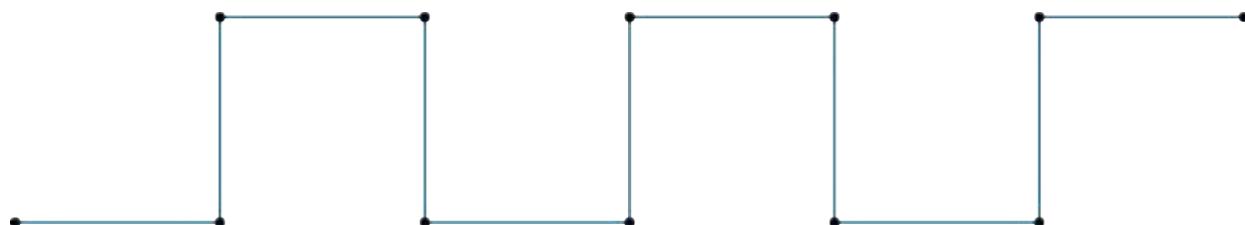
**Control Points:** Die Kontrollpunkte sind eine Liste von mindestens Grad+1 Punkten. Einer der einfachsten Wege die Form einer NURBS Kurve zu verändern ist es die Kontrollpunkte zu bewegen.

**Weight:** Kontrollpunkte haben eine mit ihnen assoziierte Nummer, Gewicht genannt. Gewichtete sind üblicherweise positive Zahlen. Wenn alle Kontrollpunkte einer Kurve das gleiche Gewicht (normalerweise 1) haben, ist die Kurve nicht-rational, sonst ist die Kurve rational. Die meisten NURBS Kurven sind nicht-rational. Wenige NURBS Kurven, wie Kreise und Ellipsen sind immer rational.

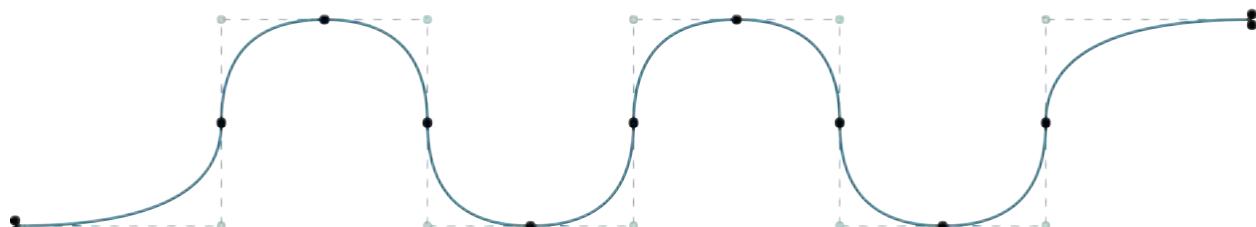
**Knots:** Knoten sind eine Liste von (degree+N-1) Zahlen, wobei N die Anzahl der Kontrollpunkte darstellt.

**Edit Points:** Punkte auf einer Kurve werden nach ihrem Knotendurchschnitt ausgewertet. Bearbeitungspunkte sind wie Kontrollpunkte, außer dass sie sich immer auf der Kurve befinden und die Veränderung der Lage eines Bearbeitungspunktes generell die Form der gesamten Kurve verändert (das Bewegen eines Kontrollpunktes kann die Form der Kurve nur lokal beeinflussen). Bearbeitungspunkte sind nützlich, wenn es darum geht einen Punkt im Inneren einer Kurve genau durch eine bestimmte Position zu legen.

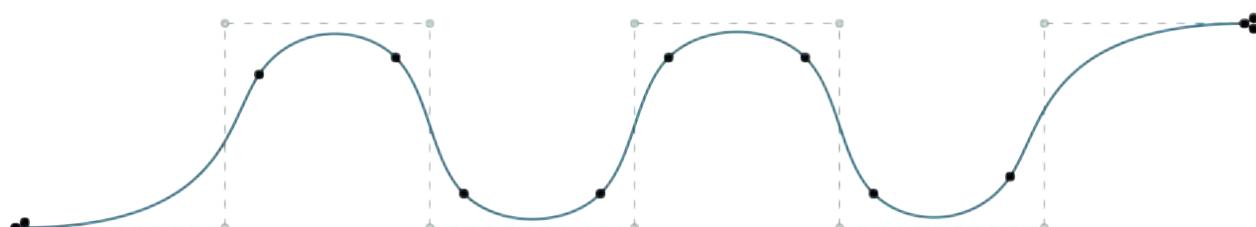
NURBS Kurvenknoten sind das Resultat von sich ändernden Graden:



AD<sup>1</sup> NURBS Kurven verhalten sich wie Polylinien. Eine D<sup>1</sup> Kurve hat einen Knoten für jeden Bearbeitungspunkt.



D<sup>2</sup> NURBS Kurven sind typischerweise nur selten angewandt um Bögen und Kreise anzunähern. Die Splinekurve schneidet das Kontrollpolygon auf der Hälfte eines jeden Segmentes.



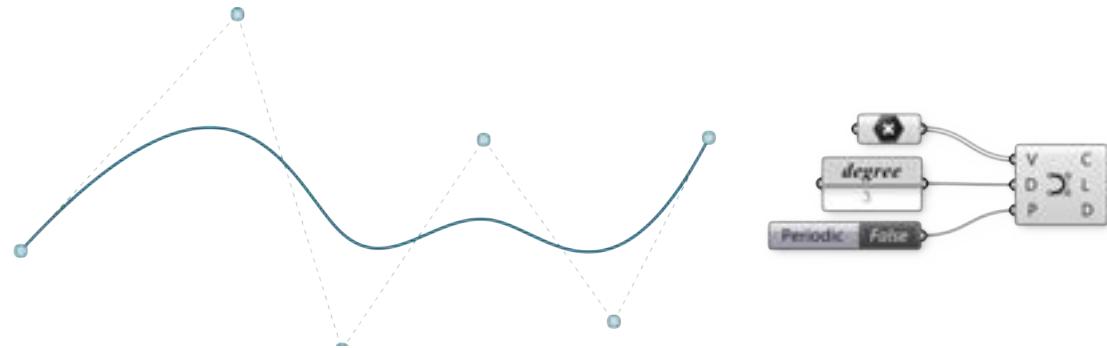
D<sup>3</sup> ist die häufigste NURBS Kurve und ist der Standard in Rhino. Du bist wahrscheinlich mit der visuellen Progression eines Splines vertraut, auch wenn die Knoten in seltsamen Orten zu liegen scheinen.

## 1.4.1.2. GRASSHOPPER SPLINE KOMPONENTEN

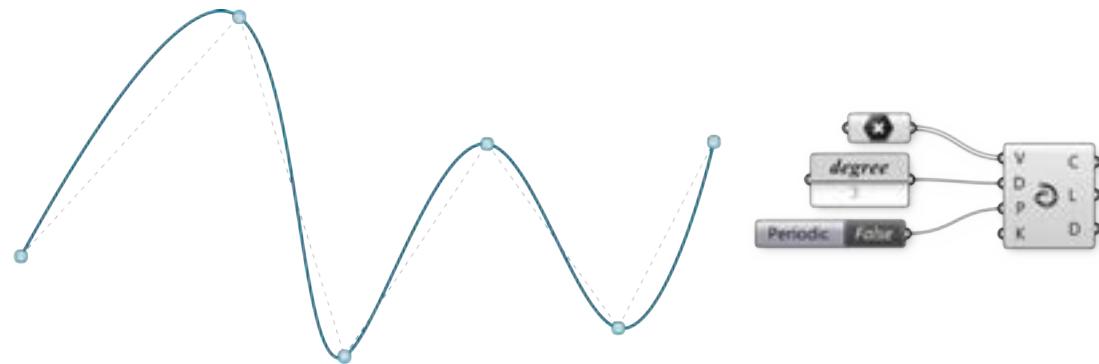
Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

Grasshopper hat eine Menge Werkzeuge um Rhino's fortgeschrittene Kurventypen wie NURBS Kurven und Polylinien auszudruecken. Diese Werkzeuge koennen unter dem "Curve/Splines" Reiter gefunden werden.

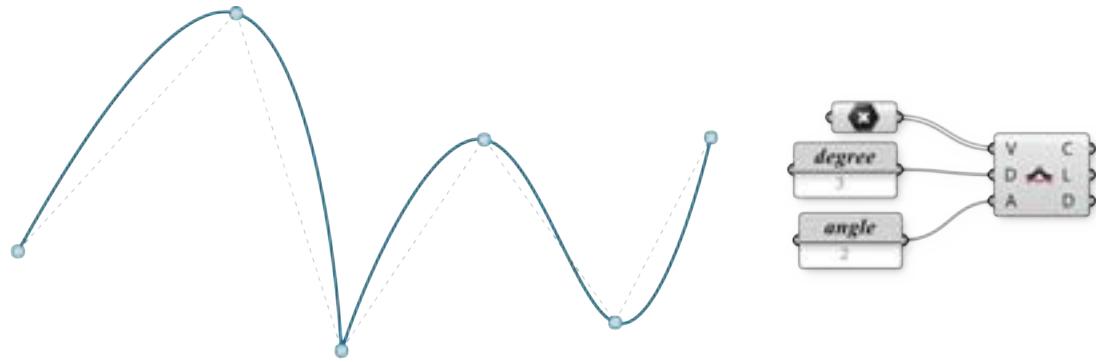
**Nurbs Curve** (Curve/Spline/Nurbs curve): Die NURBS Komponente konstruiert NURBS Kurven von Kontrollpunkten. Der V Eingabeparameter definiert diese Punkte, welche implizit durch die Auswahl von Punkten in der Rhinoszene oder durch die Eingabe volatiler Daten von anderen Komponenten beschrieben werden koennen. Der D Eingabeparameter der NURBS Komponente legt den Grad der Kurve fest.



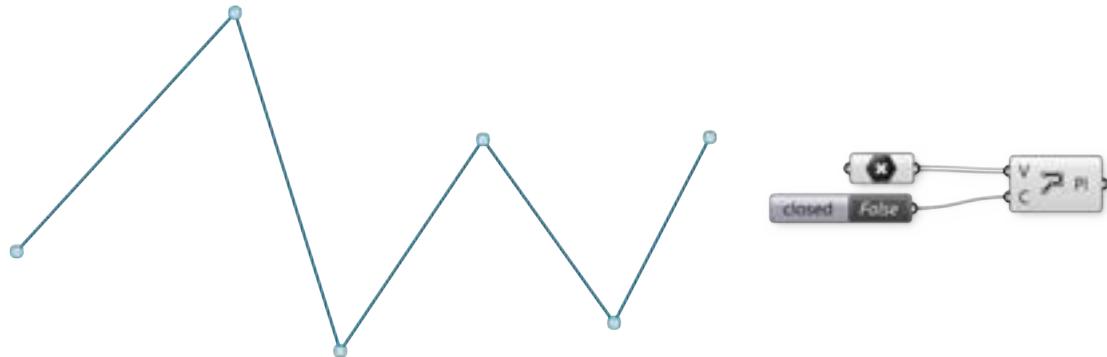
**Interpolate Curve** (Curve/Spline/Interpolate): Interpolierung von Kurven verhaelt sich etwas unterschiedlich zu NURBS Kurven. Der V Eingabeparameter der Komponente ist aehnlich zu dem der NURBS Komponente, das heisst er fordert eine bestimmte Menge von Punkten um eine Kurve zu bestimmen. Jedoch wird die resultierende Kurve bei der interpolierten Kurve genau durch diese Punkte hindurchfuehren, egal welchen Grad die Kurve aufweist. Wie bei der NURBS Kurve beschreibt der D Eingabeparameter der Komponente den Grad der erzeugten Kurve. In diesem Fall werden jedoch nur ungerade Werte fuer den Grad angenommen. Der P Eingabeparameter bestimmt wieder, ob die Kurve periodisch ist. Du wirst beginnen ein Muster in den Ausgabeparametern fuer viele der Kurvenkomponenten zu erkennen, wie z.B. dass C, L und D Ausgabeparameter generell die resultierende Kurve bestimmen, deren Laenge und die Domaeine der Kurve.



**Kinky Curve** (Curve/Spline/Kinky Curve): Die "Kinky curve" Komponente erlaubt es den spezifischen Winkelschwellenwert Afestzulegen, der bestimmt, wann die Kurve von einer abgehackten Kurve zu einer glatten, interpolierten Kurve uebergeht. Es soll hier angemerkt werden, dass der A Eingabeparameter im Bogenmass angegeben werden muss.



**Polyline (Curve/Spline/Polyline):** Eine Polylinien ist eine Sammlung von Liniensegmenten, die zwei oder mehrere Punkte verbinden, deren resultierende Linie immer durch ihre Kontrollpunkte verlaeuft; aehnlich wie bei der interpolierten Kurve. Wie die oben genannten Kurventypen, gibt der V Eingabeparameter der Polylinien Komponente eine Menge an Punkten vor, welche die Grenzen eines jeden Liniensegmentes bestimmen, aus dem die Polylinie besteht. Der C Eingabeparameter der Komponente definiert ob oder ob nicht die Polylinie geschlossen oder offen ist. Wenn der erste Punkt nicht die selbe Lage har wie der letzte Punkt, wird ein Liniensegment eingefuegt um die Kurve zu schliessen. ie Ausgabe der Polylinienkomponente ist unterschiedlich von den anderen, da der einzige uebergebene Parameter die Kurve selbst ist.

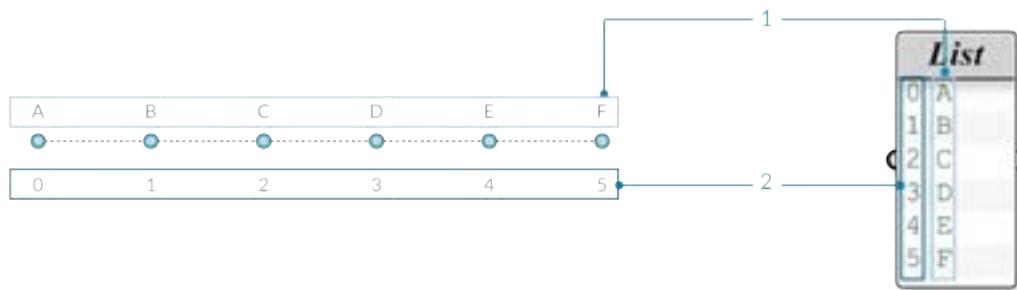


### ### 1.4.2. Was ist eine Liste?

**Es ist hilfreich ueber Grasshopper in den Begriffen von Fluessen zu denken, da es eine graphische Benutzeroberflaeche zur Bewegung von Datenfluessen in bestimmte Komponenten ist. Es sind jedoch die Daten die den Informationsfluss in und aus bestimmten Typen von Komponenten bestimmen. Zu verstehen, wie Listen manipuliert werden ist kritisch im Verstaendnis des Grasshopper Plug-Ins.**

Grundsätzlich hat Grasshopper zwei Datentypen: persistente und volatile. Auch wenn die Datentypen verschiedene Charakteristika haben, speichert Grasshopper diese Daten typischerweise in einem Array, eine Liste von Variablen.

Wenn Daten in einer Liste gespeichert werden, ist es hilfreich zu wissen an welcher Position in der Liste jedes Element gespeichert wurde, so dass wir beginnen koennen darauf zuzugreifen oder bestimmte Elemente zu bearbeiten. Die Position eines Elementes in der Liste wir Index genannt.

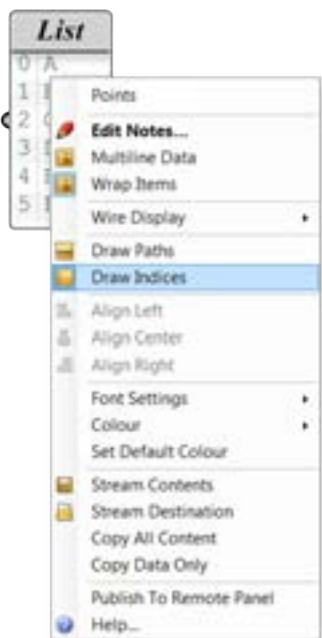


1. Listenelement
2. Index

Das einzige, was zu beginn etwas ungewoehnlich aussieht ist, dass der erste Index einer Liste immer 0 ist und nicht 1. Deshalb beziehen wir uns immer auf den Index 0, wenn wir ueber das erste Element einer Liste reden.

Zum Beispiel, wenn wir die Finger an unserer rechten Hand zaehlen, stehen die Chancen gut, dass Du von 1 bis 5 gezaehlt hast. Jedoch in Bezug auf Listen, die in Arrays gespeichert sind, wuerde die Liste von 0 bis 4 gezaehlt. Merke, dass wir immer noch 5 Elemente in der Liste haben; es ist lediglich so, dass ein Array ein null-basiertes Zaehlsystem verwendet. Sie koennen jeden Datentyp enthalten, den Grasshopper unterstuetzt, wie Punkte, Kurven, Flaechen, Polygonnetze, etc.

Oft ist es die einfachste Art festzustellen, welcher Datentyp in einer Liste gespeichert ist, ein "Text Panel" (Params/Input/Panel) mit dem Ausgabeparameter der zu bestimmenden Komponente zu verbinden. Standardmaessig, wird das Textpaneel alle Indices an der linken Seite des Paneels anzeigen und die entsprechenden Daten auf der rechten Seite des Paneels. Die Indices werden ein entscheidendes Element, wenn wir beginnen mit Listen zu arbeiten. Du kannst die Indices an- und ausschalten, indem Du auf das Textpaneel rechtsklicks und dann auf das "Draw Indices" Element des Untermenüs klickst. Momentan lassen wir die Indices erst mal fuer alle Textpaneele eingeschalten.

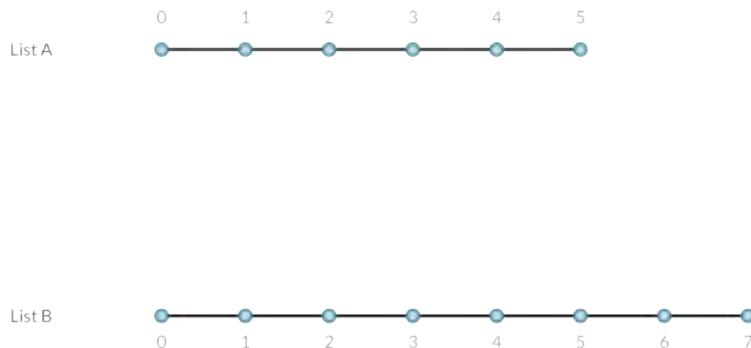


### ### 1.4.3. Abgleichung von Datenstroemen

Beispiele zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

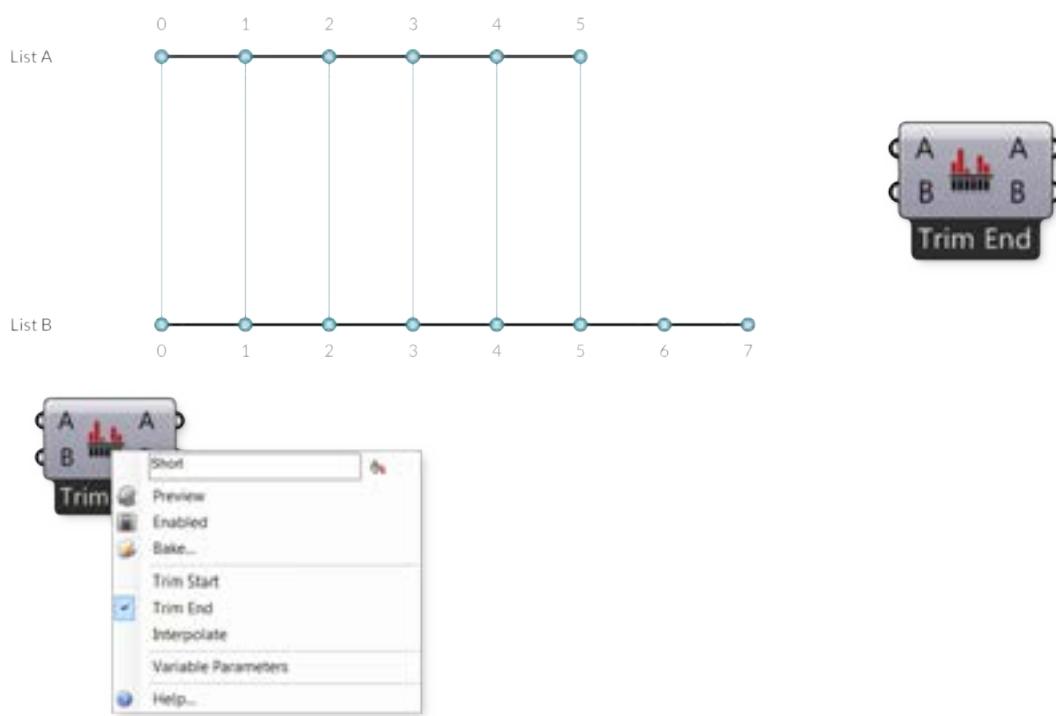
**Datenabgleich ist ein Problem ohne saubere Loesung. Es taucht auf, wenn eine Komponente Zugriff auf Eingabeparameter mit unterschiedlicher Anzahl von Datenelementen hat. Die Veraenderung des Datenabgleichalgorithmus kann zu sehr verschiedenen Ergebnissen fuehren.**

Stelle Dir eine Komponente vor, die eine Linie zwischen Punkten zeichnet. Sie wird zwei Eingabeparameter haben, die beide Punktkoordinaten beinhalten (Liste A und Liste B):



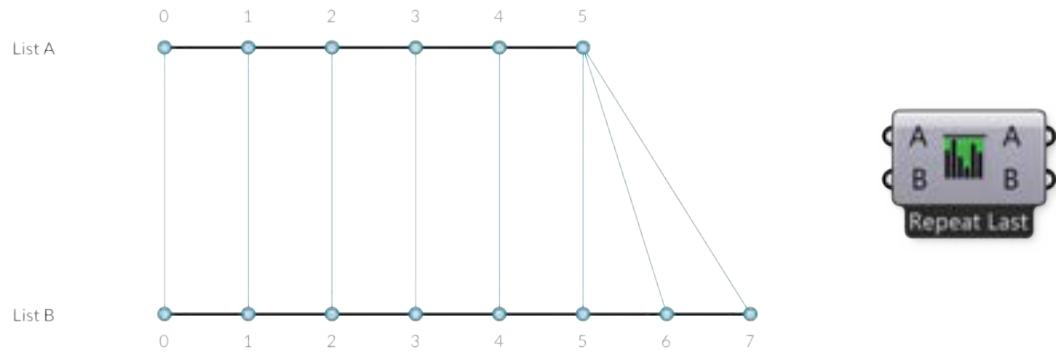
Wie Du hier sehen kannst gibt es verschiedene Arten die Linien von einer Menge von Punkten zur anderen zu zeichnen. Neu in Grasshopper 0.9 sind die drei Komponenten zum Datenabgleich, die im "Sets/List" Reiter gefunden werden: "Shortest List", "Longest List", und "Cross Reference". Diese neuen Komponenten erlauben groessere Flexibilitaet der drei grundlegenden Datenabgleichalgorithmen. Ein Rechtsklick auf die jeweilige Komponente gibt die Moeglichkeit mehrere Datenabgleichoptionen aus einem Menu auszuwaehlen.

Der einfachste Weg ist es die Elemente des Eingabeparameters eins zu eins zu verbinden bis der Datenstrom abreisst. Dies nennt man **"Shortest List"** Algorithmus:

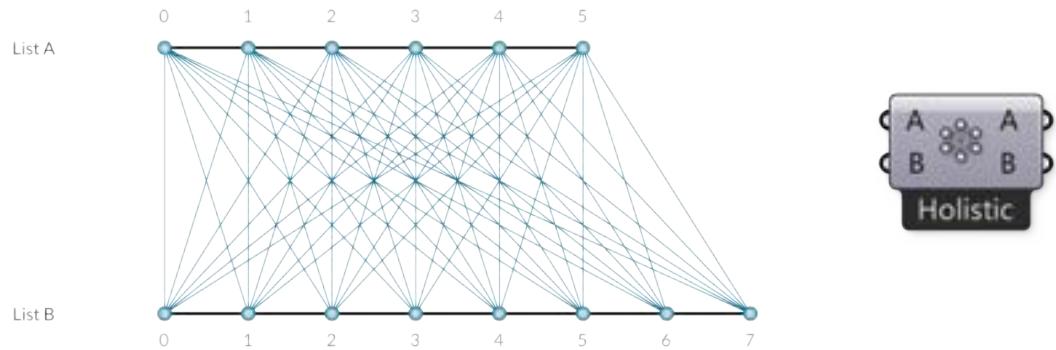


Wahle eine Option fuer den Datenabgleichalgorithmus aus dem Komponentenmenu, indem Du auf die Komponente rechtsklickst.

Der “**Longest List**” Algorithmus verbindet Elemente des Eingabeparameters bis beide Datenstroeme abreissen.  
Dis ist das Standardverhalten der Komponente:



Schliesslich macht die “**Cross Reference**” Methode alle moeglichen Verbindungen:

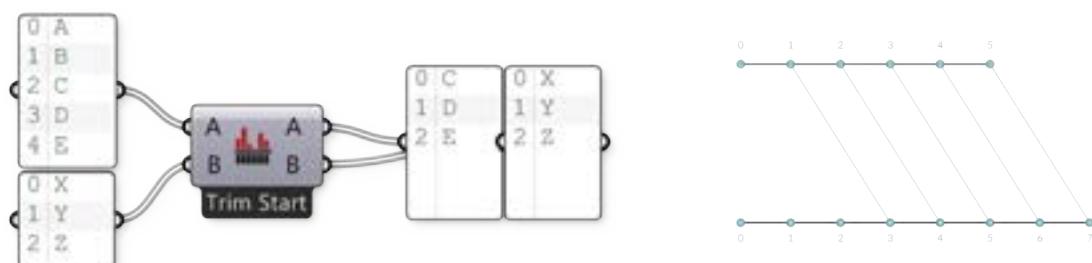


Dies ist grundsaezlich gefaehrlich, da die Anzahl von Elementen im Ausgabeparameter enorm sein kann. Das Problem wird offensichtlicher, wenn mehr Eingabeparameter involviert sind und die volatile Datenvererbung die Daten vervielfacht, aber die Logik der Definition die selbe bleibt.

Lass uns die "Shortest List" Komponente etwas genauer ansehen:

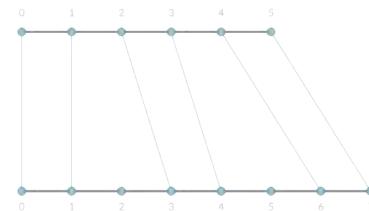
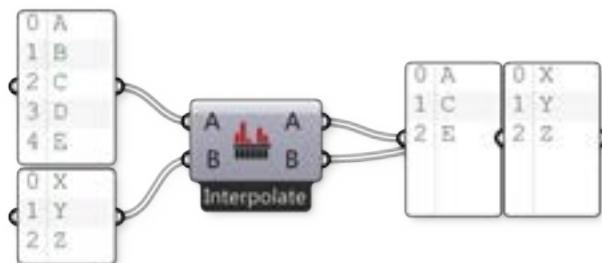


In diesem Fall haben wir zwei Eingabelisten {A,B,C,D,E} und {X,Y,Z}. Benutzen wir die "Trim End" Option, so werden die beiden letzten Elemente der ersten Liste ausser acht gelassen, do dass die beiden Listen die gleiche Laenge haben.

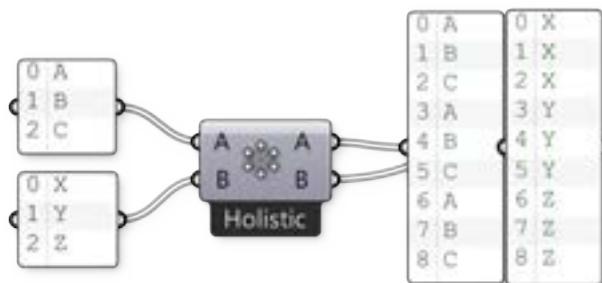


Die "Trim Start" Option laesst die beiden ersten Elemente der ersten Liste asser acht, so dass die beiden Listen

die gleiche Laenge haben.



Die "Interpolate" Option ueberspringt das zweite und vierte Element der ersten Liste. Nun sehen wir uns die "Cross Reference" Komponente an:

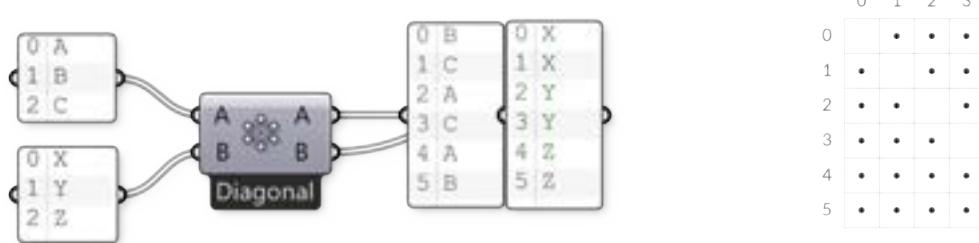


Nun haben wir zwei Eingabelisten {A,B,C} und {X,Y,Z}. Normalerweise wuerde Grasshopper ueber beide Listen iterieren und die Kombinationen {A,X}, {B,Y} und {C,Z} bilden. Hier jedoch werden sechs weitere Kombinationen beruecksichtigt: {A,Y}, {A,Z}, {B,X}, {B,Z}, {C,X} und {C,Y}. Wie Du sehen kannst besteht die Ausgabe der "Cross Reference" Komponente momentan aus neun Permutationen.

Wir koennen das Verhalten der "Cross Reference" Komponente in einer Tabelle darstellen. Die Zeilen repraesentieren die erste Liste und ihre Elemente, die Spalten die zweite. Wenn wir alle moeglichen Permutationen erzeugen, wird die Tabelle in jeder Zelle einen Punkt haben, da jede Zelle eine einzigartige Kombination von zwei Quellelementen darstellt.

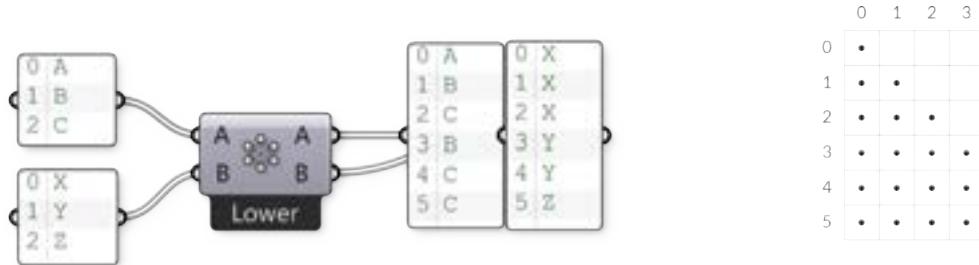
	0	1	2	3
0	•	•	•	•
1	•	•	•	•
2	•	•	•	•
3	•	•	•	•
4	•	•	•	•
5	•	•	•	•

Manchmal jedoch moechtest Du nicht alle moeglichen Permutationen erhalten. Manchmal willst Du bestimmte Zonen aussparen, weil sie bedeutungslose oder falsche Berechnungen liefern wuerden. Ein haeufiges Ausschlussprinzip ist es alle Zellen auszusparen, die auf der Diagonale der Tabelle liegen (diese Option ist als "diagonal" in der "Cross Reference" Komponente enthalten) und schliesst die Kombinationen {0,0}, {1,1}, {2,2} und {3,3} aus. Wenn wir dies auf unsere Listen {A,B,C}, {X,Y,Z} anwenden, wuerden wir erwarten die Kombinationen {A,X}, {B,Y} und {C,Z} vorzufinden:

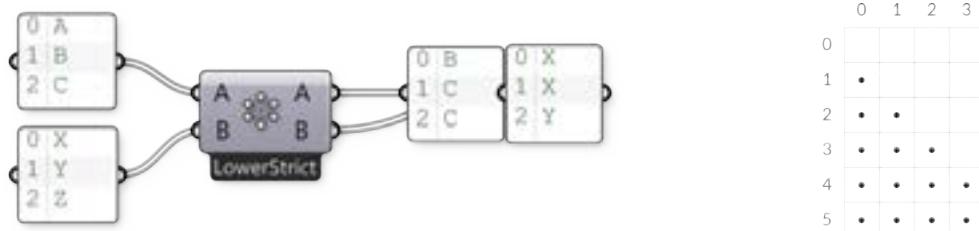


Die Regel fuer den diagonalen Datenabgleich ist: "Ueberspringe alle Permutationen, in denen die Elemente jeweils den selben Listenindex haben". 'Coincident' Datenabgleich ist der selbe Algorithmus wie Datenabgleich, wenn zwei Listen als Eingabe vorhanden sind, aber die Regel ist etwas anders: "Lasse alle Permutationen aus, in denen zwei Elemente die selben Listenindices aufweisen".

Die vier verbleibenden Datenabgleichsalgorithmen sind alle Variationen des selben Themas. 'Lower triangle' Datenabgleich wendet folgende Regel an: "Lasse alle Permutationen aus, in denen der Index eines Elementes niedriger ist als der Index des Elementes in der naechsten Liste", was zu einem leeren Dreieck fuehrt, das aber Elemente auf der Diagonale beinhaltet.



'Lower triangle (strict)' Datenabgleich geht einen Schritt weiter, indem er auch die Elemente auf der Diagonale ausspart:



'Upper Triangle' und 'Upper Triangle (strict)' spiegeln das Verhalten des vorher beschriebenen Algorithms, was zu einem leeren Dreieck auf der anderen Seite der Diagonale fuehrt.

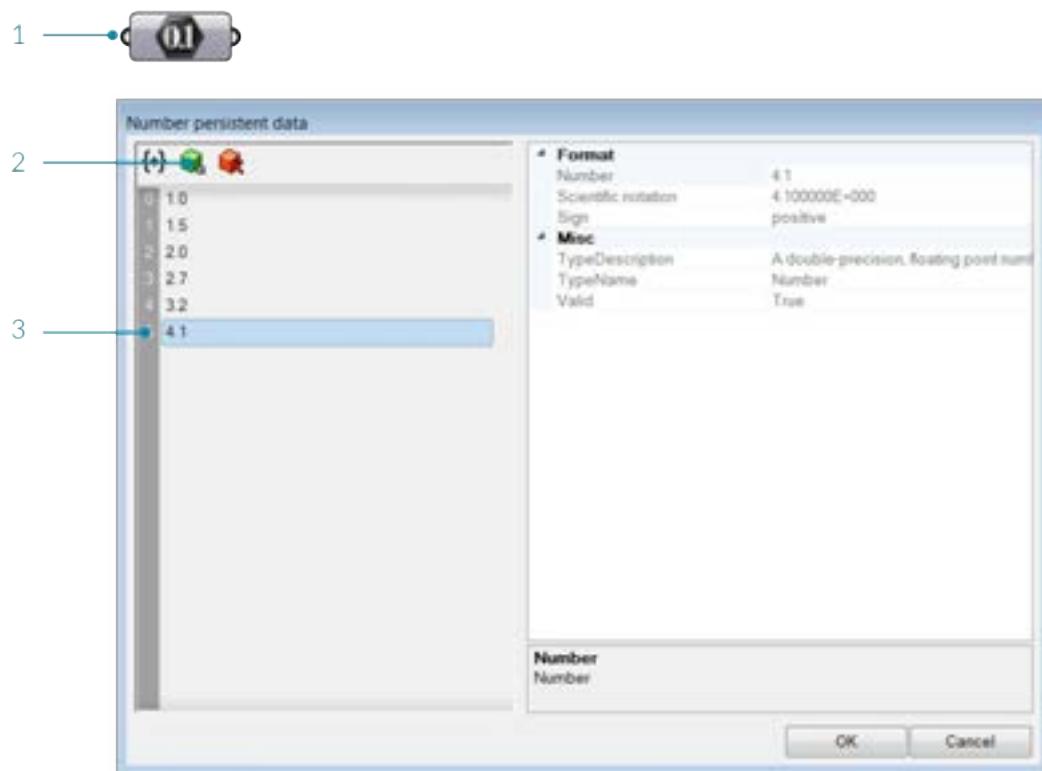
## 1.4.4. Listen erstellen

Beispieldatei fuer diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

**Es gibt viele verschiedene Wege um Listen in Grasshopper zu erzeugen. Weiter unten werden wir ein paar Methoden zur Erzeugung von Listen anschauen und dann genauer untersuchen, wie die Daten genutzt werden können um Informationen im Ansichtsfenster zu visualisieren.**

### 1.4.4.1. HAENDISCHE ERZEUGUNG VON LISTEN

Vielleicht der einfachste Weg um Listen zu erstellen (und die am meisten übersehene Methode) ist die handische Eingabe von Werten einer Liste in einen Parameter. Die Nutzung dieser Methode erhöht die Verantwortung die der Nutzer trägt, weil diese Methode auf der direkten Eingabe durch den Nutzer beruht (d.h. persistente Daten) um die Liste zu erstellen. Damit die Werte der Liste verändert werden muss der Nutzer jeden einzelnen Wert individuell eingeben, was sich entsprechend schwierig gestaltet, wenn die Liste viele Werte beinhaltet. Es gibt einige Wege um Listen handisch zu erstellen. Ein Weg ist es einen Zahlenparameter zu nutzen. Rechtsklick auf den Zahlenparameter und wähle "Manage Number Collection".



1. Rechtsklicke den Zahlenparameter um den "Number collection Manager" zu öffnen.
2. Klicke das "Add Item" Symbol um eine Zahl zur Liste hinzuzufügen.
3. Doppelklicke eine Zahl um ihren Wert zu ändern.

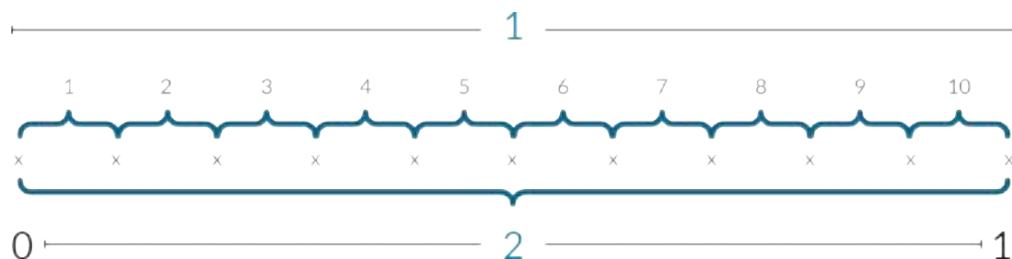
Eine andere Methode ist die manuelle Eingabe der Liste in ein Panel. Versichere Dich, dass die Option "Multiline Data" ausgeschaltet ist.



#### 1.4.4.2. RANGE

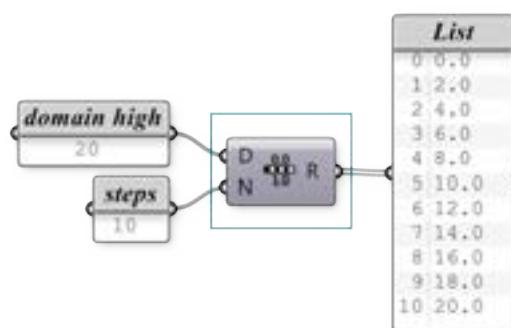
Die "Range" Komponente, die unter "Sets/Sequence/Range" gefunden werden kann, erzeugt eine Liste von gleichmaessig verteilten Zahlen zwischen einem Tief- und einem Hochwert, genannt Domaene. Eine Domaene (manchmal auch Intervall genannt) besteht aus jeder moeglichen Zahl zwischen zwei nummerischen Extremen.

Eine "Range" Komponente unterteilt eine nummerische Domaene in gleiche Segmente und gibt eine entsprechende Liste von Werten aus.



1. Anzahl von Schritten = 10
2. Domaene erstreckt sich von 0 bis 1
3. Gesamtanzahl von Punkten = 13

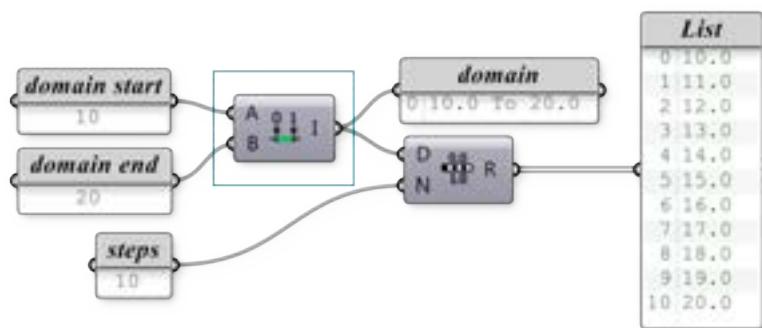
Im Beispiel unterhalb, wurde die nummerische Domaene zwischen 0 und 20 definiert. Die "Range" Komponente nimmt diese Domaene und teile sie durch die Anzahl von Schritten (in diesem Fall 10). Nun haben wir 10 gleichmaessig verteilte Segmente. Die "Range" Komponente gibt die Liste von Werten zurueck. Weil der erste und letzte Wert der Liste erhalten bleiben, ergibt die Ausgabe einer "Range" Komponente immer eine Zahl mehr als die angegebene Anzahl von Schritten. Im Beispiel oben haben wir 10 Schritte angegeben, also liefert die "Range" Komponente 11 Werte.



Erstelle eine Liste mit der "Range" Komponente indem Du eine Domaene und eine Anzahl von Schritten definierst.

Dir ist vielleicht aufgefallen, dass an dem eben erstellten Setup etwas seltsam ist. Wir wissen, dass eine domaene immer durch zwei Werte definiert wird (dem Hoch- und Tiefwert). Jedoch haben wir in unserer Definition nur einen einzelnen Wert mit dem Eingabeparameter der Domaene verbunden. Um Fehler zu vermeiden, nimmt Grasshopper an, dass wir versuchen eine Domaene zu erstellen, die von null bis zu einem bestimmten Wert (dem

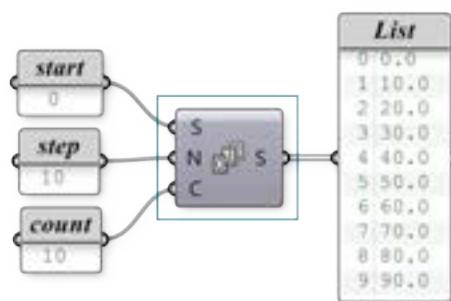
Wert unseres Schiebereglers) geht. Damit die Reihe zwischen zwei Werten erstellt wird, die nicht bei null beginnen, muessen wir die "Construct Domain" Komponente nutzen um die Domaene zu spezifizieren.



Um eine Reihe mit einer Domaene zu erstellen, die nicht bei null beginnt, nutze die "Construct Domain" Komponente.

#### 1.4.4.3. SERIEN

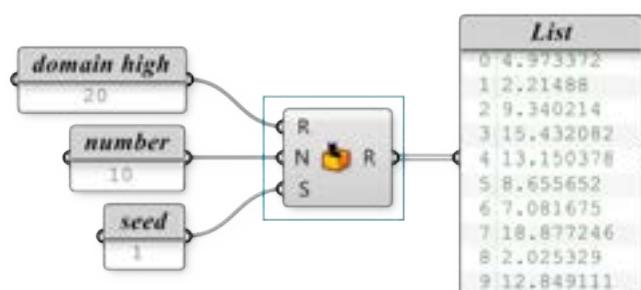
Die "Series" Komponente ist aehnlich zur "Range" Komponente, da sie auch eine Liste mit Zahlen erstellt. Jedoch ist die "Series" Komponete auch verschieden, weil sie eine Reihe von bestimmten Zahlen erstellt, die durch einen Startwert, die Schrittgroesse und die Anzahl der gewuenschten Werte beschrieben wird.



Die "Series" Komponente erstellt eine Liste basierend auf einem Startwert, der Schrittgroesse und der Anzahl von Werten, die in der Liste enthalten sein sollen.

#### 1.4.4.4. RANDOM

Die "Random" Komponente (Sets/Sequence/Random) kann sogenannte pseudorandomierte Werte erzeugen. Sie werden "pseudo" randomiert gennant, da sie eine einzigartige Zahlenfolde beschreiben, die stabil aus einem Samenwert hervorgehen. Deshalb kannst Du eine komplett zufaellige Zahl nur dadurch erzeugen, dass Du den Samenwert veraenderst (S Eingabeparameter). Die Domaene, wie im vorangegangenen Beispiel, wird durch ein Intervall zwischen zwei numerischen Extremen definiert.

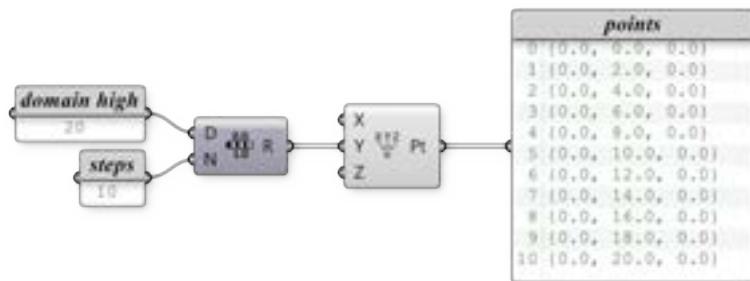


### ### 1.4.5. Listen Visualisieren

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

**Listen in Grasshopper zu verstehen kann schwierig sein, ohne die Daten von einer Komponente zur anderen fliessen zu sehen. Dafuer gibt es mehrere Wege um Listen zu visualisieren, die Dir helfen koennen Daten zu verstehen und zu veraendern.**

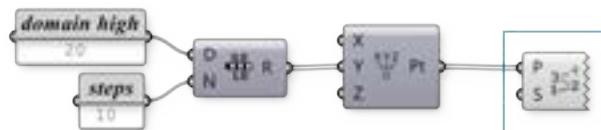
Es gibt viele verschiedene Wege Listen von Daten zu visualisieren. Der uebliche Weg ist es eine Geometrie mit der Liste zu erstellen. Indem Du den R Ausgabeparameter der "Range" Komponente mit dem Y Eingabeparameter der "Construct Point" Komponente verbindest, kannst Du sehen, wie eine Reihe von Punkten in der Y Richtung dargestellt wird.



Lass uns einige Komponenten betrachten, die uns helfen Daten zu verstehen.

#### 1.4.5.1. DIE PUNKTLISTENKOMPONENTE

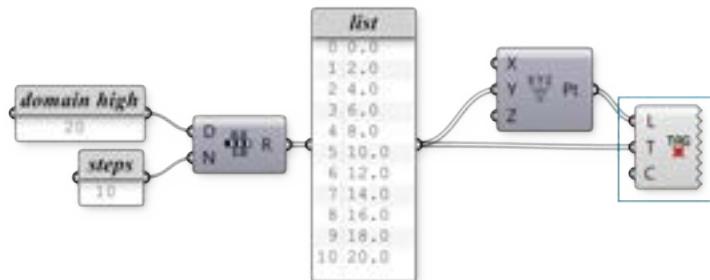
Die "Point List" Komponente ist ein nuetzliches Werkzeug um die Reihenfolge einer Menge von Punkten in einer Liste darzustellen. Essential platziert die "Point List" Komponente einen Index neben die Punktgeometrie im Ansichtsfenster. Du kannst auch angeben, ob Du die Nummerierung, die Verbindungslien darstellen willst und welche Groesse der Text haben soll.



Du kannst die Reihenfolge einer Punktmenge mit der "Point List" Komponente darstellen.

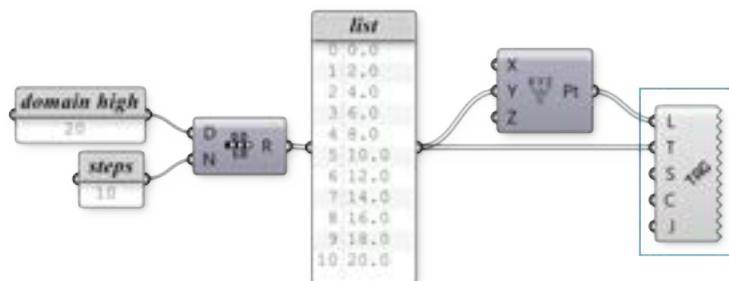
#### 1.4.5.2. TEXTMARKEN

Die "Text Tag" Komponente ermoeglicht es kurze Strings zu zeichnen (ein String ist eine Reihe von ASCII characters) um eine visuelle Ausgabe im Ansichtsfenster zu erhalten. Text und Position koennen als Eingabeparameter angegeben werden. Wenn Textmarken in die Szene gebacken werden, werden sie zu "Text Dots". Die andere interessante Sache ueber Textmarken ist, dass sie unabhaengig von Ansichtsfenstern sind - das bedeutet, dass die Marken immer zur Kamera zeigen (inkl. Perspektive) und dass sie, unabhaengig von der Vergroesserung) immer in der selben Groesse dargestellt werden.



Du kannst Stringinformation im Ansichtsfenster mit der "Text Tag" Komponente darstellen. In diesem Setup haben wir uns entschieden die Werte eines jeden Punktes oberhalb der Punktposition anzugeben. Wir koennen jeden Text entsprechend darstellen.

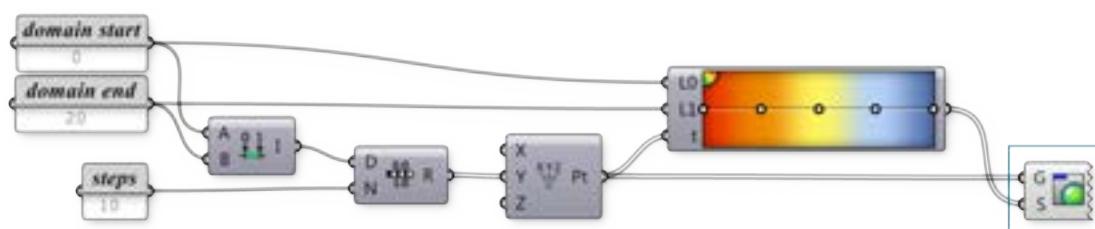
Die "Text Tag 3d" Komponente funktioniert aehnlich wie die "Text Tag" Komponente. Sie unterscheiden sich dadurch, dass wenn ein "Text Tag 3d" Objekt gebacken wird, es auch in Rhino ein Textobjekt wird. Die Skalierung der "Text Tag 3d" Schriftart kann auch durch einen Eingabeparameter bestimmt werden (was bei der "Text Tag" Komponente nicht geht).



Du kannst die "Text Tag 3d" Komponente nutzen um Informationen als Textobjekt in Rhino darzustellen.

#### 1.4.5.3. FARBE

Eines der anderen Dinge die wir zur Visualisierung von Datenlisten nutzen koennen ist die Zuweisen von Farben zu Geometrien. Grasshopper hat nur begrenzte Renderingmoeglichkeiten, aber wir koennen einfache Open GL Einstellungen, wie Farbe, Glanzfarbe, Transparenz usw. festlegen. Der L0 Wert repraesentiert das untere Ende (Linke Seite) eines Gradienten, wobei der L1 Wert entsprechend das obere Ende (rechte Seite) darstellt. Diese Werte entsprechen dem Start- und Endwert unserer Domäne. Die t Werte sind die Elemente der Liste, die auf die Spanne zwischen L0 und L1 abgebildet werden. Die Ausgabe des Gradienten ist eine Liste von RGB Werten, welche jeweils einem Punkt in der Liste entsprechen. Rechtsklicke den Gradienten um ein Present fuer den Gradienten zu waehlen oder bestimme Deine eigenen Farben an den Farbknotenpunkten.





1. Punkte
2. Punktliste
3. Text Tag
4. Text Tag 3D
5. Benutzerdefinierte Farbvorschau

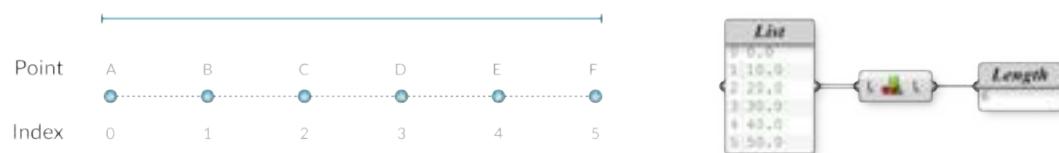
## 1.4.6. Listen Managen

Beispiele zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

Eine der maechtigsten Eigenschaften von Grasshopper ist die Moeglichkeit schnell Listen zu erstellen und zu bearbeiten. Wir speichern viele verschiedene Datentypen in einer Liste (Nummern, Punkte, Vektoren, Kurven, Flaechen, Breps, usw.) und es gibt eine Menge nuetzlicher Werzeuge hierfuer unter der "Sets/List" Unterkategorie.

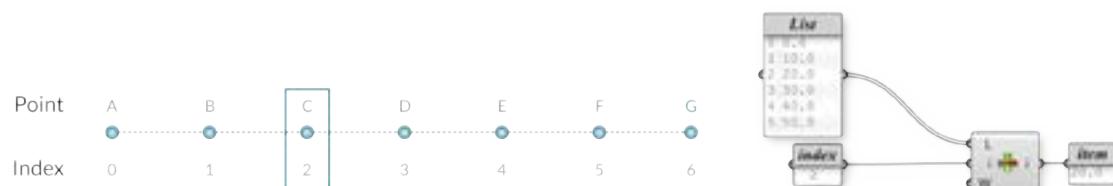
### 1.4.6.1. LISTENLAENGE

Die "List Length" Komponente (Sets/List/List Length) misst grundszaetlich die Laenge einer Liste. Da unsere Listen immer bei null beginnen, ist der hoechste moegliche Wert unseres Indexes einer Liste gleich der Laenge der Liste minus eins. In diesem Beispiel haben wir unsere Ausgangsliste mit dem L Eingabeparameter der "List Length" Komponente verbunden, um zu zeigen, dass in der Liste 6 Eintraege vorhanden sind.



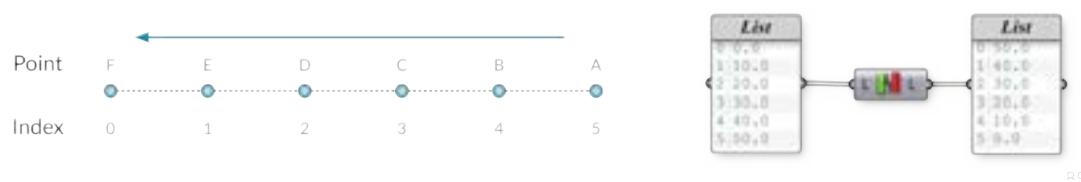
### >#1.4.6.2. LISTENELEMENTE

Unsere Liste wird mit einer "List Item" Komponente (Sets/List/List Item) verbunden, um einen bestimmten Eintrag aus der Liste zu erhalten. Wenn wir auf ein individuelles Element einer Liste zugreifen wollen, muessen wir den i Eingabeparameter bestimmen; welcher mit dem Index uebereinstimmt, auf den wir zugreifen wollen. Wir koennen einen einzelnen Integerwert oder eine Liste von Integerwerten in den i Eingabeparameter eingeben, abhaengig davon, welche Elemente wir erhalten wollen. Der L Eingabeparameter definiert die Ausgangsliste, die wir analzsieren wollen. In diesem Beispiel haben wir den i Eingabeparameter mit 5.0 angegeben, so dass die "List Item" Komponente uns die Daten, die mit dem fuenften Eintrag der Liste verknuept sind, ausgibt.



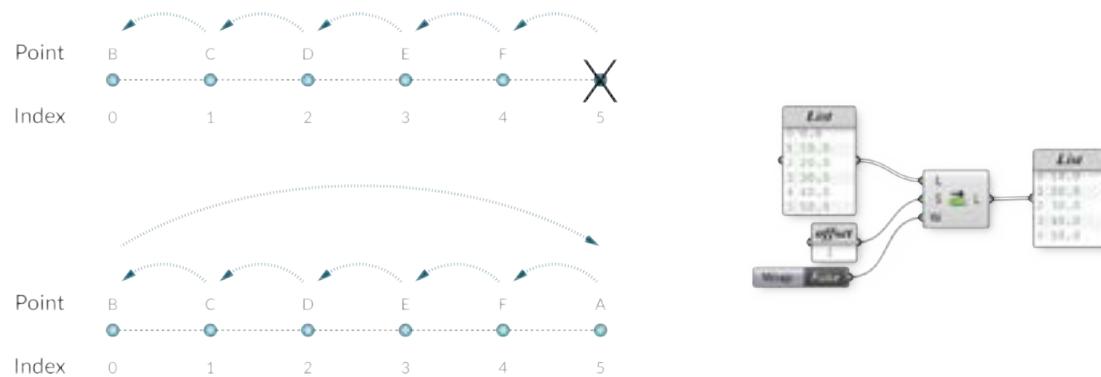
### 1.4.6.3. LISTEN UMKEHREN

Wir koennen eine Liste in ihrer Reihenfolge umkehren, indem wir die "Reverse List" Komponente (Sets/List/Reverse) anwenden. Wenn wir eine aufsteigende Liste mit Zahlen von 0.0 bis 9.0 in die "Reverse List" Komponente eingeben, gibt uns der Ausgabeparameter eine absteigende Liste von 9.0 bis 0.0 zurueck.



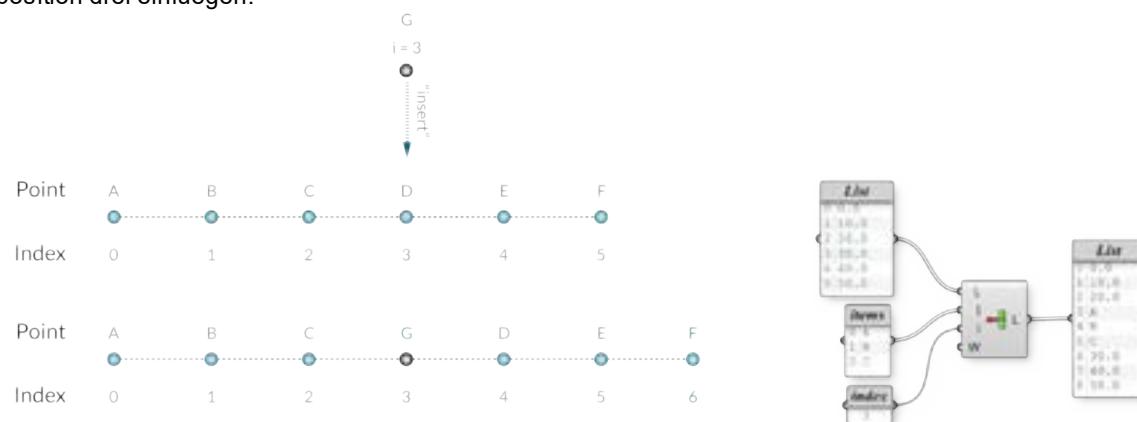
### 1.4.6.4. VERSCHIEBEN VON LISTEN

Die "Shift List" Komponente (Sets/Sequence/Shift List) wird die Reihenfolge der Liste nach oben oder unten verschieben, je nachdem welchen Wert wir fuer die Verschiebung angeben. Wir haben den Ausgabeparameter der Liste mit dem L Eingabeparameter der "Shift List" Komponente verbunden, während wir eine Zahl mit dem S Eingabeparameter verknuepfen. Wenn wir die Verschiebung mit -1 angeben, werden alle Werte in der Liste um einen Eintrag nach unten verschoben. Ebenso werden alle Werte in der Liste um einen Eintrag nach oben verschoben, wenn wir die Verschiebung mit +1 angeben. In diesem Beispiel haben wir die Verschiebung mit +1 festgelegt, so dass unsere Leiste um einen Eintrag nach oben verschoben wird. Wenn wir den "Wrap" Wert auf "false" stellen, wird der letzte Eintrag nach oben verschoben und aus der Liste fallen, was den Wert grundsätzlich aus der Liste entfernt (so dass die Laenge der Liste sich um eins reduziert). Jedoch, wenn wir den "Warp" Wert auf "true" setzen, wird der letzte Eintrag der Liste wieder unten an der Liste angefuegt.



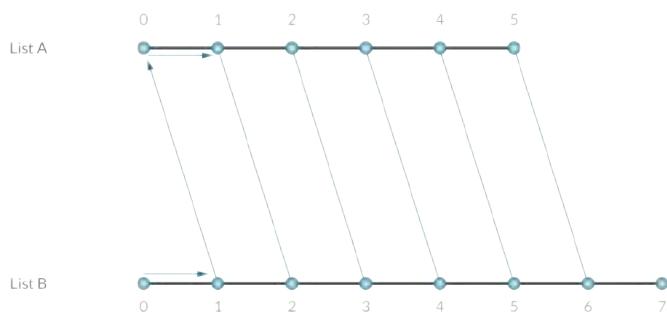
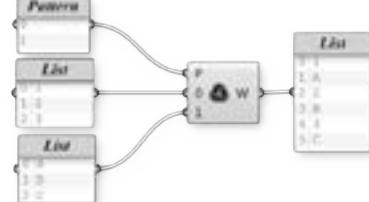
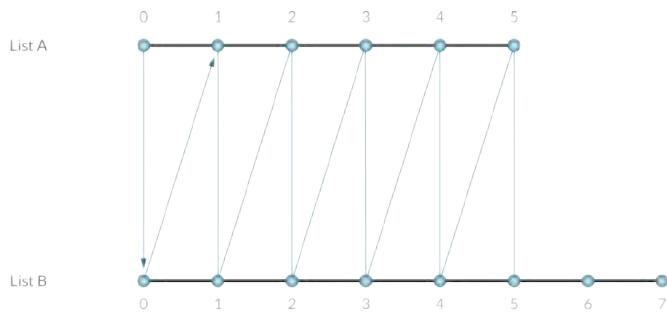
## 1.4.6.5. ELEMENTE EINFUEGEN

Die "Insert Items" Komponente (Sets/Lists/Insert Items) ermoeglicht es eine Menge an Werten in eine Liste einzufuegen. Damit dies anstaendig funktioniert, muessen wir fuer jedes Element wissen, welche Elemente wir an welcher Indexposition einfuegen wollen. Im unteren Beispiel werden wir die Buchstaben A, B und C an die Indexposition drei einfuegen.



## 1.4.6.6. WEBEN

Die "Weave" Komponente (Sets/Lists/Weave) fuehrt zwei Listen basierend auf einem Webemuster (P Eingabeparameter) zusammen. Wenn das Muster und die Datenstroeme nicht perfekt aufeinander abgebildet werden koennen, wird die Komponente entweder "null" Werte in die Eingabestroeme einfuegen, oder sie kann Stroeme ignorieren, die noch nicht erschoepft wurden.



### 1.4.6.7. AUSSORTIEREN MIT MUSTER

Die "Cull" Komponente (Sets/Sequence/Cull Pattern) entfernt Elemente einer Liste mit einer sich wiederholenden Maske. Die Maske ist definiert als eine Liste boolscher Werte (wahr oder falsch). Die Maske wird wiederholt, bis alle Elemente der Datenliste ausgewertet wurden.

Point	A	B	C	D	E	F
Index	0	1	2	3	4	5
Pattern	True	False	.....	(Repeat)		

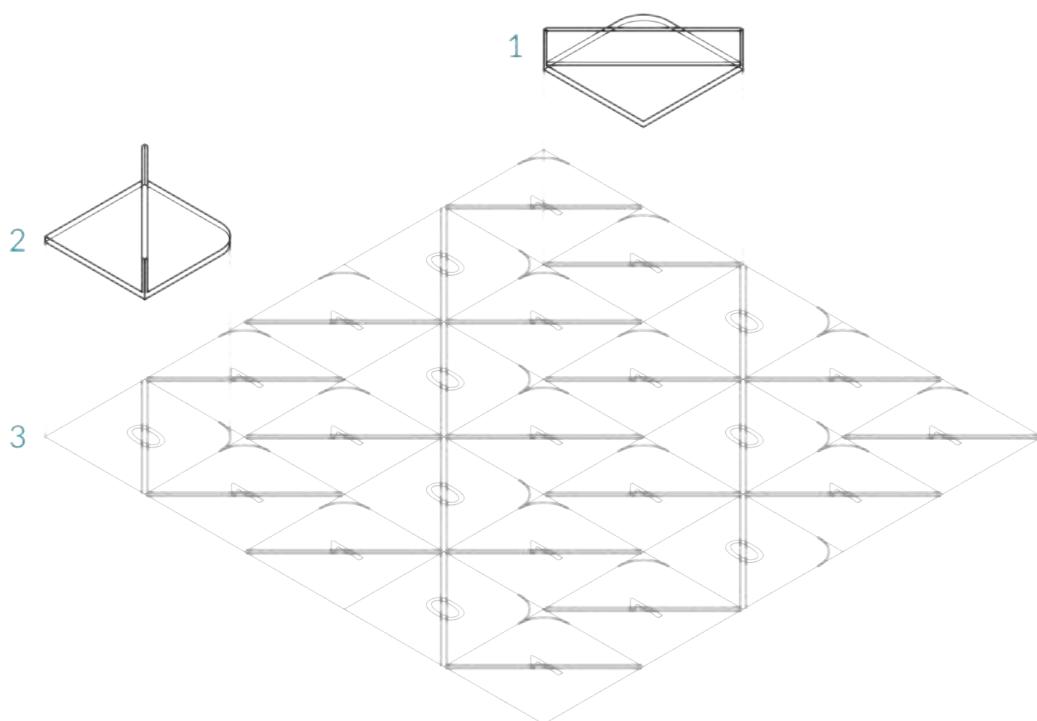
Point	A	B	C	D	E	F
Index	0	1	2	3	4	5
Pattern	True	False	True	False	True	False

Point	A		C		F	
Index	0		1		2	
Pattern	True	False	True	False	True	False

## 1.4.7. MIT LISTEN ARBEITEN

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

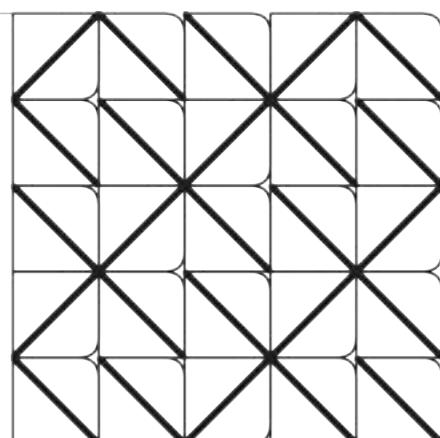
Lass uns einen Blick in ein Beispiel werfen, das die Komponenten des vorausgegangenen Abschnitts beinhaltet. In diesem Beispiel, werden wir ein Fliesenmuster erzeugen, das durch die Abbildung von Geometrie auf ein Raster erzeugt wird. Das Muster wird durch die Nutzung der "List Item" Komponente erzeugt um die gewuenschte Fliese aus einer Liste von Geometrien zu beziehen.



- 1. Geometrie entsprechend Index 1
- 2. Geometrie entsprechend Index 0
- 3. Rechtwinkliges Raster

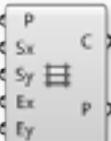
0	1	1	0	1
1	1	0	1	1
1	0	1	1	0
0	1	1	0	1
1	1	0	1	1

1

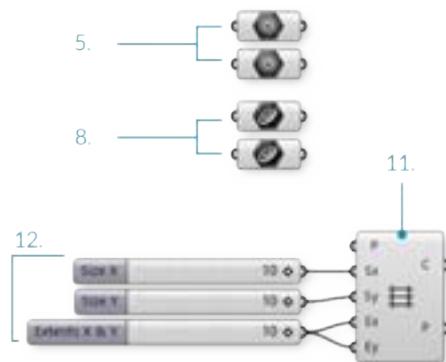


2

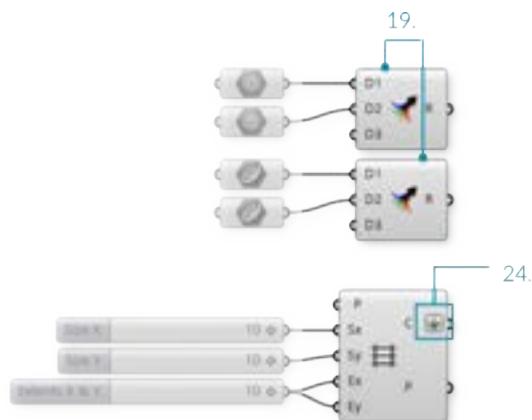
- 1. Abbildungsmuster
- 2. Abgebildete Geometrie

01.	Beginne eine Rhinoceros Datei.	
02.	Erstelle zwei gleichgrosse Quadrate.	
03.	<p>Erstelle verschiedene Geometrien innerhalb der beiden Quadrate.            Im oben dargestellten Beispiel haben wir eine einfache Flaeche mit einem Knick erstellt. Der Knick ist ausgerundet, um die Ausrichtung darzustellen und die Basis ist ausgerundet um die beiden Geometrien zu unterscheiden.</p>	
04.	Beginne eine neue Definition, druecke Strg+N (in Grasshopper).	
05.	<b>Params/Geometry/Geometry</b> – Ziehe zwei <b>Geometry</b> Parameter auf die Leinwand.	
06.	Rechtsklicke den ersten <b>Geometry</b> Parameter und waehle "Set one Geometry". Referenziere die erste Geometrie.	
07.	<p>Rechtsklicke auf den zweiten <b>Geometry</b> Parameter und waehle "Set one Geometry". Waehle die zweite Geometrie, die Du referenzieren willst.            Es ist moeglich mehrere Geometrien in einem einzigen Parameter zu referenzieren, aber der Einfachheit halber werden wir zwei verschiedene Parameterkomponenten benutzen.</p>	
08.	<b>Params/Geometry/Curve</b> – Ziehe zwei <b>Curve</b> Parameter auf die Leinwand.	
09.	Rechtsklicke den ersten <b>Curve</b> Parameter und waehle "Set one Curve". Waehle die erste Kurve, die Du referenzieren moechtest.	
10.	Rechtsklicke den zweiten <b>Curve</b> Parameter und waehle "Set one Curve". Waehle die zweite Kurve, die Du referenzieren moechtest. Versichere Dich, dass die Geometrie und das Quadrat, die Du auswaehlst miteinander korrespondieren.	
11.	<b>Vector/Grid/Rectangular</b> – Ziehe eine <b>Rectangular Grid</b> Komponente auf die Leinwand.	
12.	<b>Params/Input/Slider</b> - Ziehe drei <b>Number Sliders</b> auf die Leinwand	
13.	Doppelklicke auf den ersten <b>Number Slider</b> und setze folgende Werte: Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10	
14.	Doppelklicke auf den zweiten <b>Number Slider</b> und setze folgende Werte: Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10	
15.	Doppelklicke den dritten <b>Number Slider</b> und setze folgende Werte: Name: Extents X & Y Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10	

16.	Verbinde den ersten <b>Number Slider</b> mit dem Groesse X (Sx) Eingabeparameter der <b>Rectangular Grid</b> Komponente.	
17.	Verbinde den zweiten <b>Number Slider</b> mit dem Groesse Y (Sy) Eingabeparameter der <b>Rectangular Grid</b> component.	
18.	Verbinde den dritten <b>Number Slider</b> mit dem Abmessung X (Ex) Eingabeparameter und dem Abmessung Y (Ey) Eingabeparameter der <b>Rectangular Grid</b> Komponente.	



19.	<b>Sets/Tree/Merge</b> – Ziehe zwei <b>Merge</b> Komponenten auf die Leinwand.	
20.	Verbinde den ersten <b>Geometry</b> Parameter mit dem Datenstrom 1 (D1) Eingabeparameter der ersten <b>Merge</b> Komponente.	
21.	Verbinde den zweiten <b>Geometry</b> Parameter mit dem Datenstrom 2 (D2) Eingabeparameter der ersten <b>Merge</b> Komponente.	
22.	Verbinde den ersten <b>Curve</b> Parameter mit dem Datenstrom 1 (D1) Eingabeparameter der zweiten <b>Merge</b> Komponente.	
23.	Verbinde den zweiten <b>Curve</b> Parameter mit dem Datenstrom 2 (D2) Eingabeparameter der zweiten <b>Merge</b> Komponente.	
24.	Rechtsklicke auf den Zellen (C) Ausgabeparameter der <b>Rectangular Grid</b> Komponente und waehle "Flatten".	



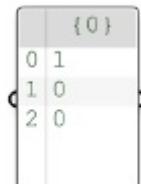
25.	<b>Sets/List/List Length</b> – Ziehe eine <b>List Length</b> Komponente auf die Leinwand.	
26.	Verbinde den Zellen (C) Ausgabeparameter der <b>Rectangular Grid</b> Komponente mit dem Liste (L) Eingabeparameter der <b>List Length</b> Komponente.	

27.	Sets/Sequence/Repeat Data – Ziehe eine <b>Repeat Data</b> Komponente auf die Leinwand.	
28.	Verbinde den Laenge (L) Ausgabeparameter der <b>List Length</b> Komponente mit dem Laenge (L) Eingabeparameter der <b>Repeat Data</b> Komponente.	
29.	<b>Params/Input/Panel</b> – Ziehe ein <b>Panel</b> auf die Leinwand.	
30.	Doppelklicke das <b>Panel</b> . Deaktiviere "Multiline Data", "Wrap Items" und "Special Codes".	

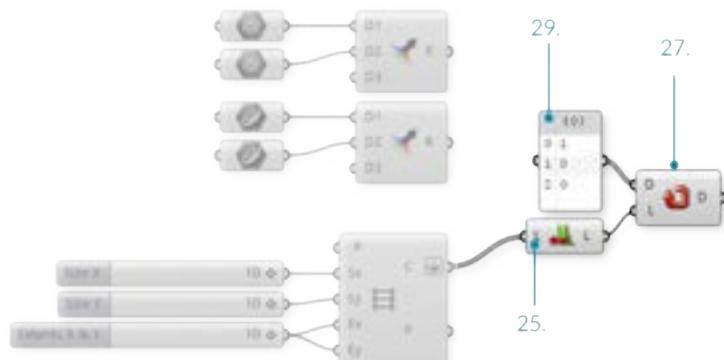
Gebe folgenden Text ein:

1  
0  
0

Dies ist das Muster in dem die Geometrien verteilt werden. 0 ruft die erste referenzierte Geometrie auf und 1 ruft die zweite referenzierte Geometrie auf. Änderungen an der Reihenfolge der Zahlen oder an den Abmessungen des Rasters werden das Muster verändern.

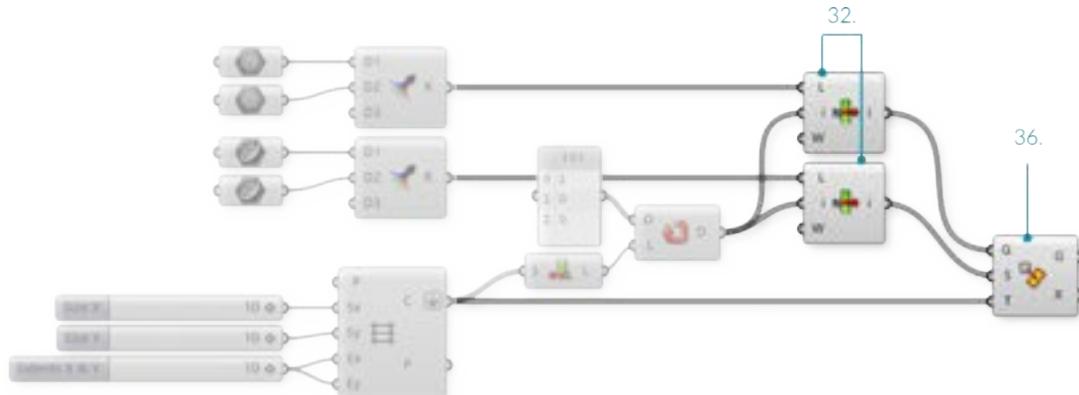


| [31.] Verbinde das **Panel** mit dem Daten (D) Eingabeparameter der **Repeat Data** Komponente.|||

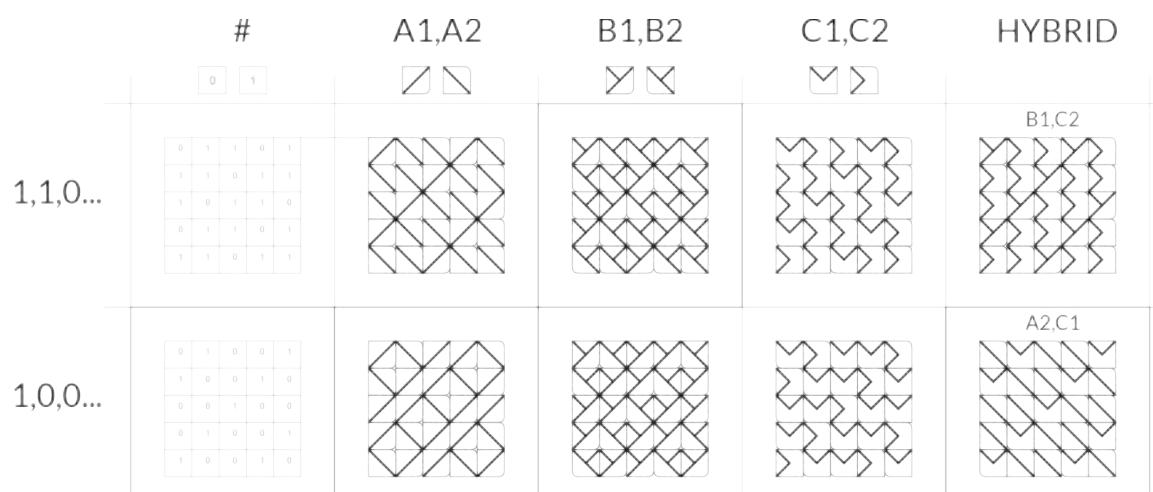


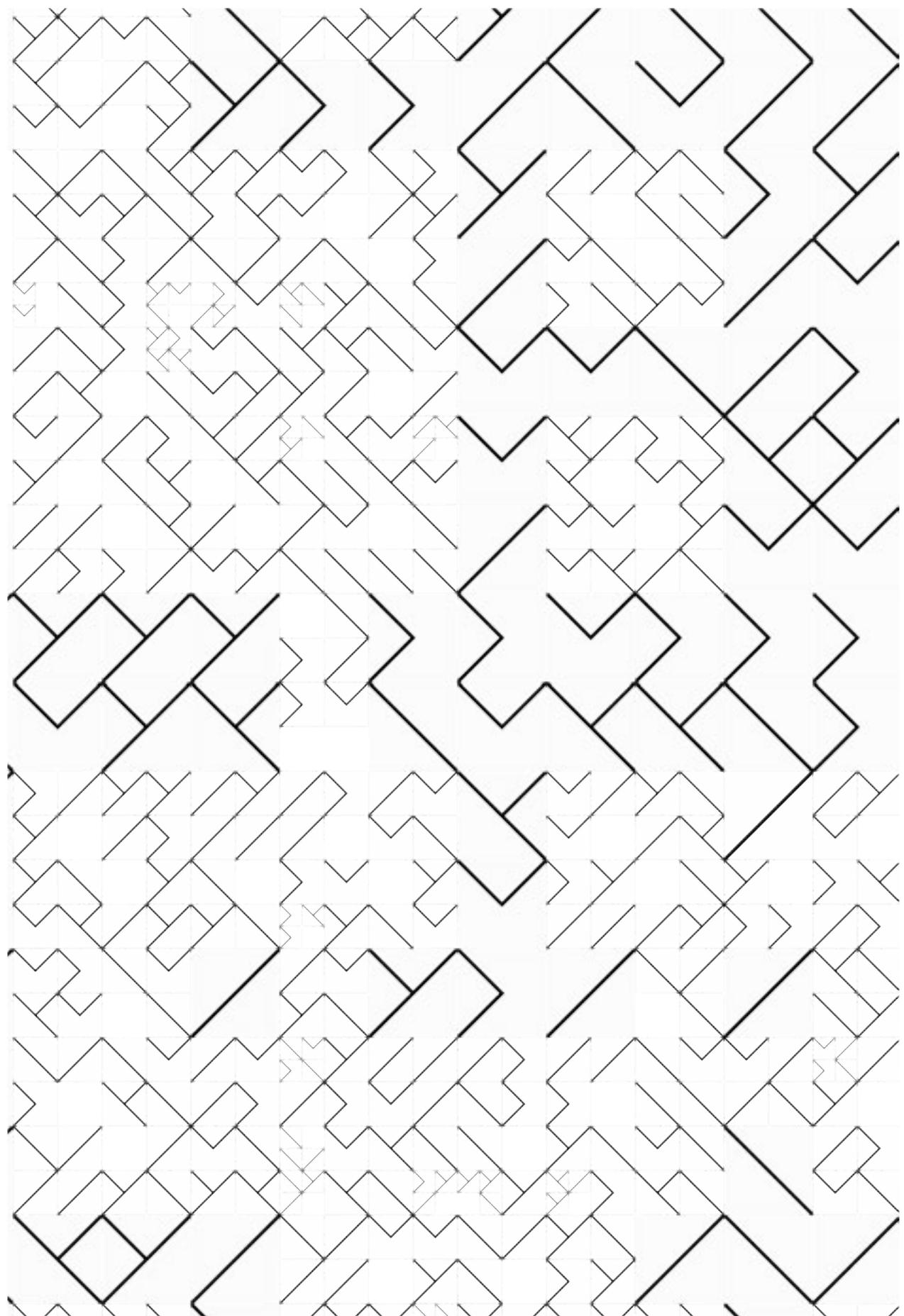
32.	Sets/List/List Item – Ziehe zwei <b>List Item</b> Komponenten auf die Leinwand.	
33.	Verbinde den Ergebnis (R) Ausgabeparameter der ersten <b>Merge</b> Komponente mit dem Liste (L) Eingabeparameter der ersten <b>List Item</b> Komponente.	
34.	Verbinde den Ergebnis (R) Ausgabeparameter der zweiten <b>Merge</b> Komponente mit dem Liste (L) Eingabeparameter der zweiten <b>List Item</b> Komponente.	
35.	Verbinde den Daten (D) Ausgabeparameter der <b>Repeat Data</b> Komponente mit dem Index (i) Eingabeparameter der ersten und zweiten <b>List Item</b> Komponente.	

36.	<b>Transform/Affine/Rectangle Mapping</b> – Ziehe eine <b>Rectangle Mapping</b> Komponente auf die Leinwand.	
37.	Verbinde den Zellen (C) Ausgabeparameter der <b>Rectangular Grid</b> Komponente mit dem Ziel (T) Eingabeparameter der <b>Rectangular Mapping</b> Komponente.	
38.	Verbinde den Elementen (I) Ausgabeparameter der ersten <b>List Item</b> Komponente mit dem Geometrie (G) Eingabeparameter der <b>Rectangular Mapping</b> Komponente.	
39.	Verbinde den Elementen (I) Ausgabeparameter der zweiten <b>List Item</b> Komponente mit dem Quelle (S) Eingabeparameter der <b>Rectangular Mapping</b> Komponente.	



Veraenderungen an der Eingabegeometrie und am Muster werden das Fliesenmuster veraendern.





## 1.5. ENTWERFEN MIT DATENBAEUMEN

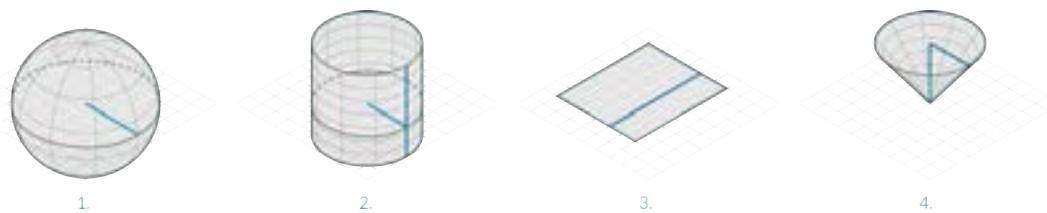
Da Deine Definitionen in Komplexitaet zunehmen, wird der Umfang der uebertragenen Daten auch steigen. Um Grasshopper effektiv zu nutzen, ist es wichtig zu verstehen, wie grosse Datenmengen gespeichert, zugaenglich gemacht und veraendert werden koennen.



### ###1.5.1. Flaechen Geometrien

**NURBS** (non-uniform rational B-splines) sind mathematische Repraesentationen, die ein Modell einer beliebigen Form akkurat aus einer einfachen 2D Linien, einem Kreis, Boden oder einer Kiste die komplexeste, organische 3D Freiformflaeche oder -koerper entwickeln koennen. Auf Grund der Flexibilitaet und Praezision der koennen NURBS Modelle in jedem beliebigen Prozess, von der Illustration und Animation bis zur Herstellung genutzt werden.

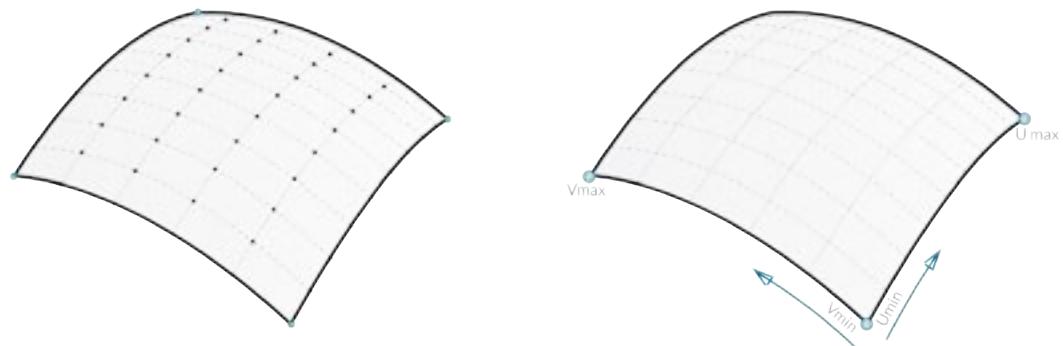
Abgesehen von ein paar primitiven Flaechentypen, wie Kugeln, Kegel, Flaechen und Zylinder, unterstuetzt Thino drei verschiedenen Arten von Freiform Flaechentypen, wobei der nuetlichste die NURBS Flaeche ist. Aehnlich wie bei den Kurven, kann mit der NURBS Flaeche jede moegliche Form als Flaeche dargestellt werden, was die Rueckfallebene in Rhino darstellt. Sie ist außerdem die bei weitem nuetlichste Flaechendefinition und die, auf die wir uns konzentrieren werden.



1. Kugel Primitivkoerper [plane, radius]
2. Zylinder Primitivkoerper [plane, radius, height]
3. Flaeche Primitivkoerper [plane, width, height]
4. Kegel Primitivkoerper [plane, radius, height]

#### 1.5.1.1. NURBS FLAECHEN

NURBS Flaechen sind sehr aehnlich zu NURBS Kurven. Die selben Algorithmen werden verwendet, um die Form, Normalen, Tangenten, Kruemmungen und anderen Eigenschaften zu berechnen, aber es gibt auch klare Unterschiede. Zum Beispiel haben Kurven Tangentenvektoren und Normalenebenen, waehrend Flaechen Normalenvektoren und Tangentialebenen besitzen. Das bedeutet, dass Kurven an definierten Positionen keine Orientierung haben, waehrend Flaechen keine definierte Richtung besitzen. Im Fall von NURBS Flaechen gibt es genau zwei Richtungen, die in der Geometrie einbeschrieben sind, weil NURBS Flaechen aus rechtwinkligen Rastern aus  $\{u\}$  und  $\{v\}$  Kurven aufgebaut sind. Auch wenn diese Richtungen oft beliebig sind, werden wir sie sowieso nutzen, weil sie uns das Leben so viel einfacher machen.

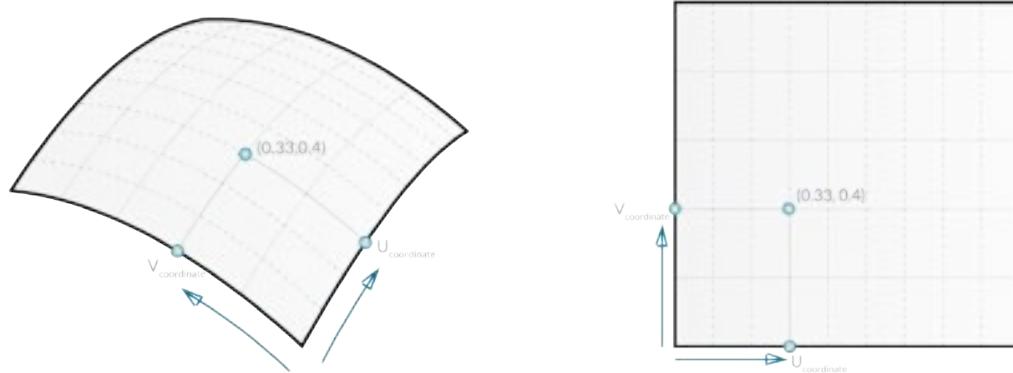


Du kannst Dir NURBS Flaechen als Raster aus NURBS Kurven in zwei Richtungen vorstellen. Die Form einer NURBS Flaeche ist durch eine Anzahl von Kontrollpunkten und den Grad der Flaeche in  $u$  und  $v$  Richtung definiert. NURBS Flaechen sind eine effiziente Art um Freiformflaechen mit einem hohen Grad an Praezision zu speichern und zu repraesentieren.

**Flaechen Domaenen** Eine Flaechendomaene ist durch einen Bereich von  $(u,v)$  Parametern bestimmt, der in einen 3D Punkt auf der Flaeche ausgewertet werden kann. Die Domaene ist normalerweise in jeder Richtung ( $u$  oder  $v$ )

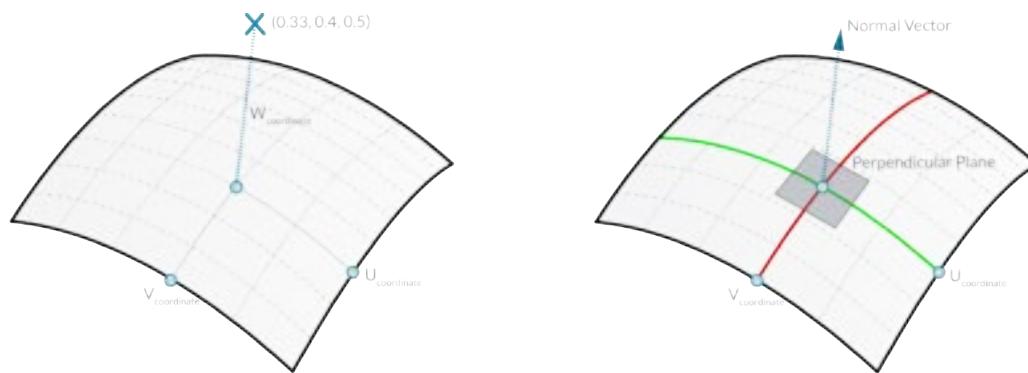
durch zwei reelle Zahlen ( $u_{\min}$  und  $u_{\max}$ ) und ( $v_{\min}$  und  $v_{\max}$ ) beschrieben. Die Domaene einer Fläche zu verändern wird Reparametrisierung genannt.

In Grasshopper ist es oft nützlich eine NURBS Fläche zu reparametrisieren, so dass die  $u$  und  $v$  Domaene jeweils von 0 bis 1 reicht. Dies ermöglicht es uns die Fläche einfach auszuwerten und mit ihr zu arbeiten.



Auswertung von Parametern eines gleichmässigen Intervalls im 2D Parameterraster wird nicht notwendigerweise eine gleimässige Unterteilung des 3D Raums bedeuten.

**Flächenauswertung** Die Auswertung einer Fläche an einem Parameter innerhalb der Flächendomaene resultiert in einem Punkt auf der Fläche. Merke Dir, dass der Mittelpunkt einer Domaene (mid-u, mid-v) nicht notwendigerweise der Mittelpunkt der 3D Fläche ist. Wenn Du nun  $u$ - und  $v$ -Werte ausserhalb der Domaene der Fläche auswerten willst, wirst Du kein nützliches Ergebnis erhalten.



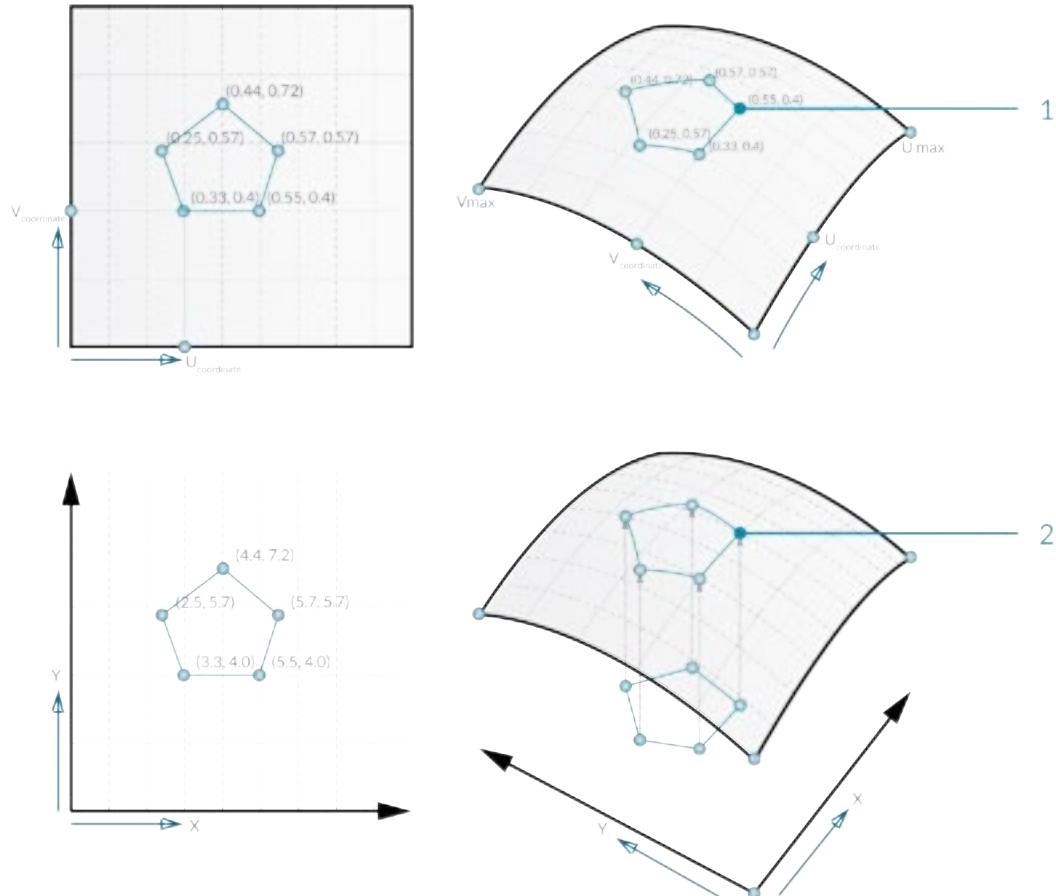
**Normalenvektoren und Tangentialebenen** Die Tangentialebene einer Fläche an einem bestimmten Punkt ist die Ebene, welche die Fläche an einem Punkt berührt. Die z-Richtung der Tangentialfläche repräsentiert die Normalentrichtung der Fläche an diesem Punkt.

Grasshopper behandelt NURBS Flächen ähnlich wie Rhino, weil es diese auf dem selben Kern von Operationen aufbaut, die notwendig sind um die Fläche zu erzeugen. Da Grasshopper jedoch die Fläche oberhalb des Rhinoansichtsfenster darstellt (aus diesem Grund kann man die Geometrien, welche in Grasshopper erzeugt werden nicht auswählen, wiss sie in die Szene gebacken werden) sind manche der Mesheinstellungen etwas niedriger, um die Geschwindigkeit von Grasshopper relativ hoch zu halten. Du hast vielleicht festgestellt, dass manche Polygonnetze (Meshes) facettiert sind, aber das ist zu erwarten und ist ein Ergebnis der Grasshopper Darstellungsoptionen. Jede gebackene Geometrie wird die höheren Mesheinstellungen von Rhino verwenden.

### 1.5.1.2. FLÄCHEN PROJIZIEREN

Im vorherigen Abschnitt, haben wir erklärt, dass NURBS Flächen ihren eigenen Koordinatenraum besitzen und

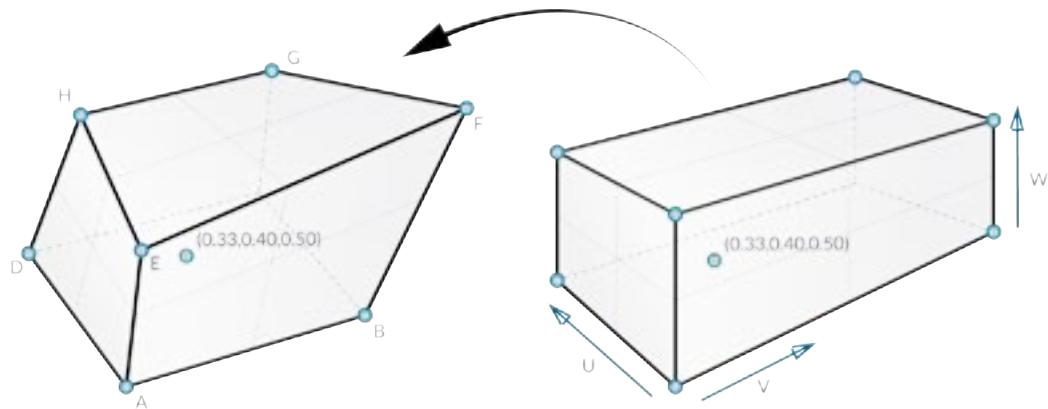
dass dieser durch u und v Domaenen beschrieben wird. Dies bedeutet, dass zweidimensionale Geometrien, welche durch x und y Koordinaten beschrieben werden auf den uv Raum der Flaeche abgebildet werden koennen. Die Geometrien wird sich veraendern und verzerren um sich an die Kruemmung der Flaeche anzupassen. Dies ist unterschiedlich zu einer einer einfachen Projektion auf eine Flaeche, bei der Vektoren von der 2D Geometrie in eine bestimmte Richtung gezeichnet werden, bis sie mit einer Flaeche verschneiden.



Du kannst Dir eine Projektion als einen Schattenwurf auf eine Flaeche vorstellen und eine Abbildung als eine Geometrie, die ueber die Flaeche gezogen wird.

1. Abgebildete Geometrie definiert durch uv Koordination
2. Projizierte Geometrie auf einer Flaeche

So wie 2D Geometrie auf den uv Raum einer Flaeche abgebildet werden kann, ist es moeglich eine Geometrie, die in einer Kiste enthalten ist, auf eine verzerrte Kiste auf einem Flaechenabschnitt abzubilden. Diese Operation wird "Box Morph" genannt und ist nuetzlich um eine gekruemmte Flaeche mit dreidimensionalen geometrischen Komponenten zu bevoelkern.

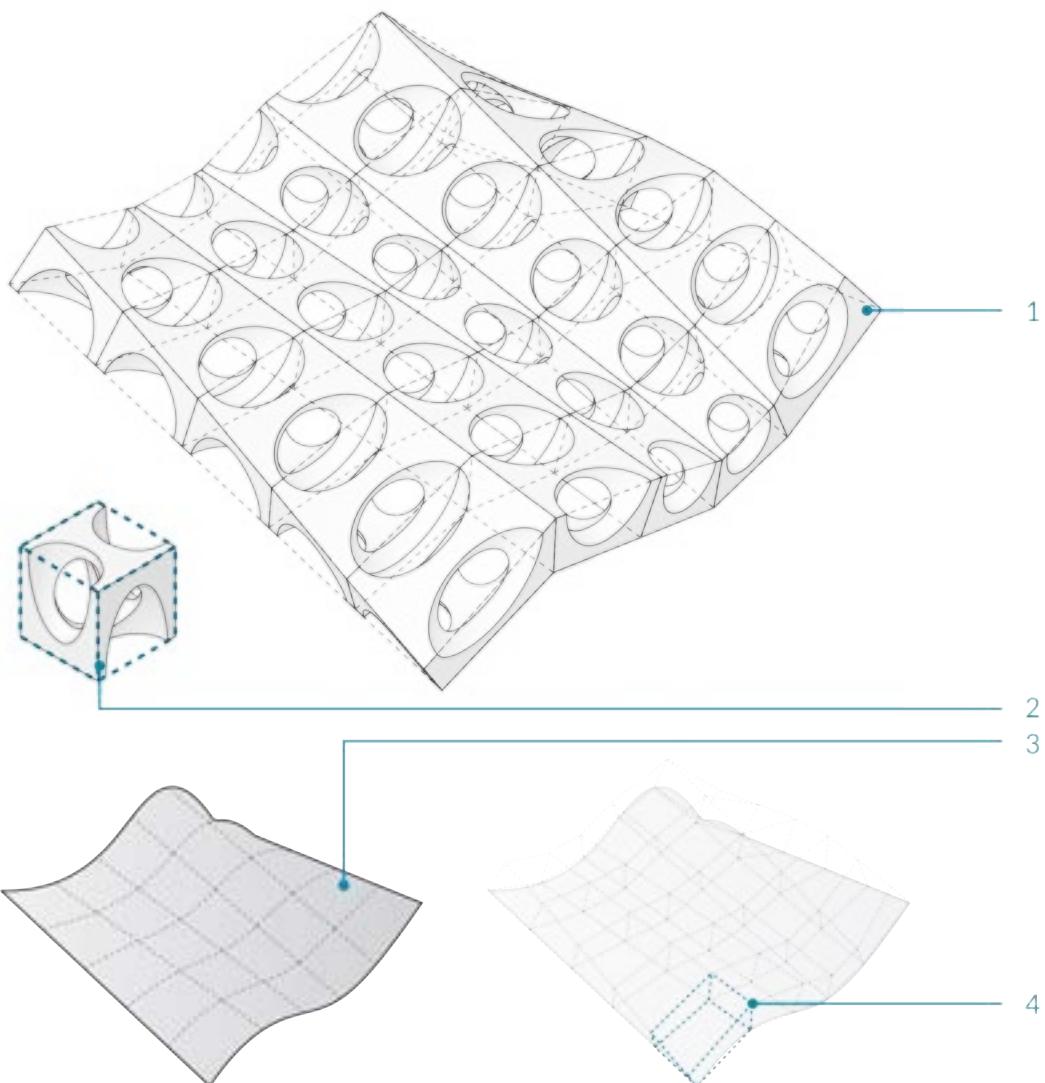


Um eine Reihe verzerrter Kisten auf einer Flaeche anzugeben, muss die Flaechendomäne unterteilt werden, um ein Raster von Flaechenabschnitten zu erzeugen. Die verzerrten Kisten werden erzeugt, indem Normalenvektoren an den Ecken eines jeden Flaechenabschnitts bis zur gewünschten Höhe angebracht werden und eine Kiste erzeugen, deren Eckpunkte durch die Endpunkte dieser Vektoren bestimmt werden.

### 1.5.1.3. MORPH DEFINITION

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

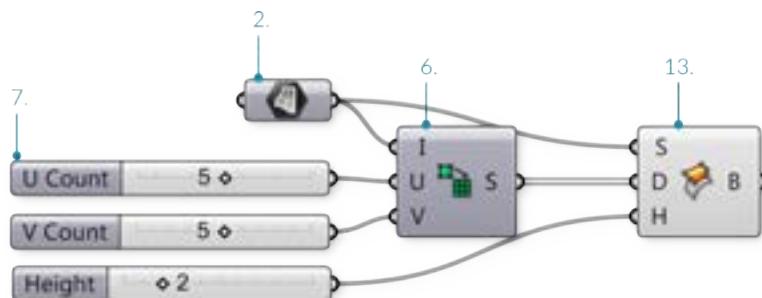
In diesem Beispiel werden wir die "Box Morph" Komponente einsetzen um eine NURBS Flaeche mit geometrischen Komponenten zu bevoelkern.



1. NURBS Flaeche bevoelkert mit Komponenten.
2. Urspruengliche Komponente in ihrer Referenzkiste.
3. Flaeche unterteilt in Flaechenabschnitte.
4. Reihe verzerrter Kisten auf einer Flaeche.

01.	Beginne eine neue Definition, druecke Strg+N (in Grasshopper)	
02.	<b>Params/Geometry/Surface</b> – Ziehe einen <b>Surface</b> Parameter auf die Leinwand Diese Flaeche werden wir mit geometrischen Komponenten bevoelken.	
03.	<b>Params/Geometry/Geometry</b> – Ziehe einen <b>Geometry</b> Parameter auf die Leinwand Dies ist die Komponente, die auf der Flaeche aufgereiht wird.	
04.	Rechtsklicke auf den <b>Surface</b> Parameter und waehle "Set One Surface" – waehle eine Flaeche im Rhinoansichtsfenster, die Du referenzieren moechtest	
05.	Rechtsklicke den <b>Geometry</b> Parameter und waehle "Set One Geometry" – waehle Deine Rhinogeometrie	
06.	<b>Maths/Domain/Divide Domain2</b> – Ziehe eine <b>Divide Domain2</b> Komponente auf die Leinwand	

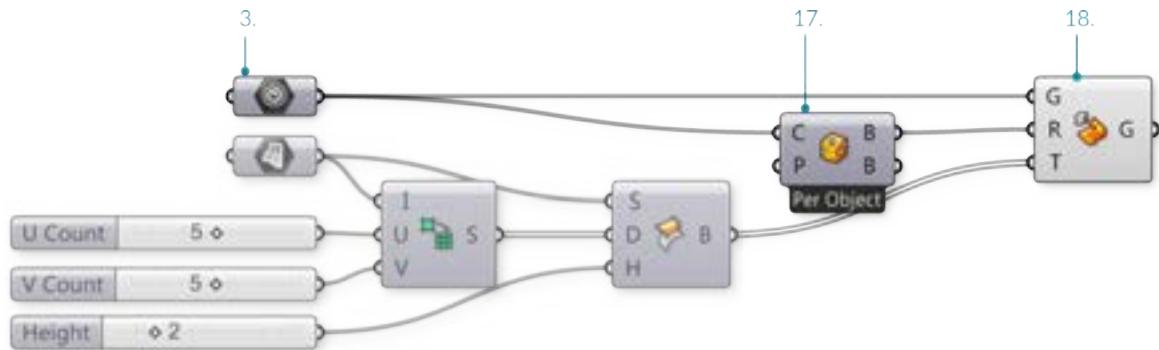
07.	Params/Input/Number Slider – Ziehe drei <b>Number Sliders</b> auf die Leinwand	
08.	Doppelklicke den ersten <b>Number Slider</b> und setze folgende Werte: Rounding: Integer Lower Limit: 0 Upper Limit: 10 Value: 5	
09.	Setze die selben Werte beim zweiten und dritten <b>Number Sliders</b>	
10.	Verbinde den Ausgabeparameter des <b>Surface</b> Parameter mit dem Domaene (I) Eingabeparameter der <b>Divide Domain2</b> Komponente	
11.	Verbinde den ersten <b>Number Slider</b> mit dem Anzahl U (U) Eingabeparameter der <b>Divide Domain2</b> Komponente	
12.	Verbinde den zweiten <b>Number Slider</b> mit dem Annzahl V (V) Eingabeparameter der <b>Divide Domain2</b> Komponente	
13.	<b>Transform/Morph/Surface Box</b> – Ziehe die <b>Surface Box</b> Komponente auf die Leinwand	
14.	Verbinde den Ausgabeparameter des <b>Surface</b> Parameter mit dem Flaeche (S) Eingabeparameter der <b>Surface Box</b> Komponente	
15.	Verbinde den Segmente (S) Ausgabeparameter der <b>Divide Domain2</b> Komponente mit dem Domaene (D) Eingabeparameter der <b>Surface Box</b> Komponente	



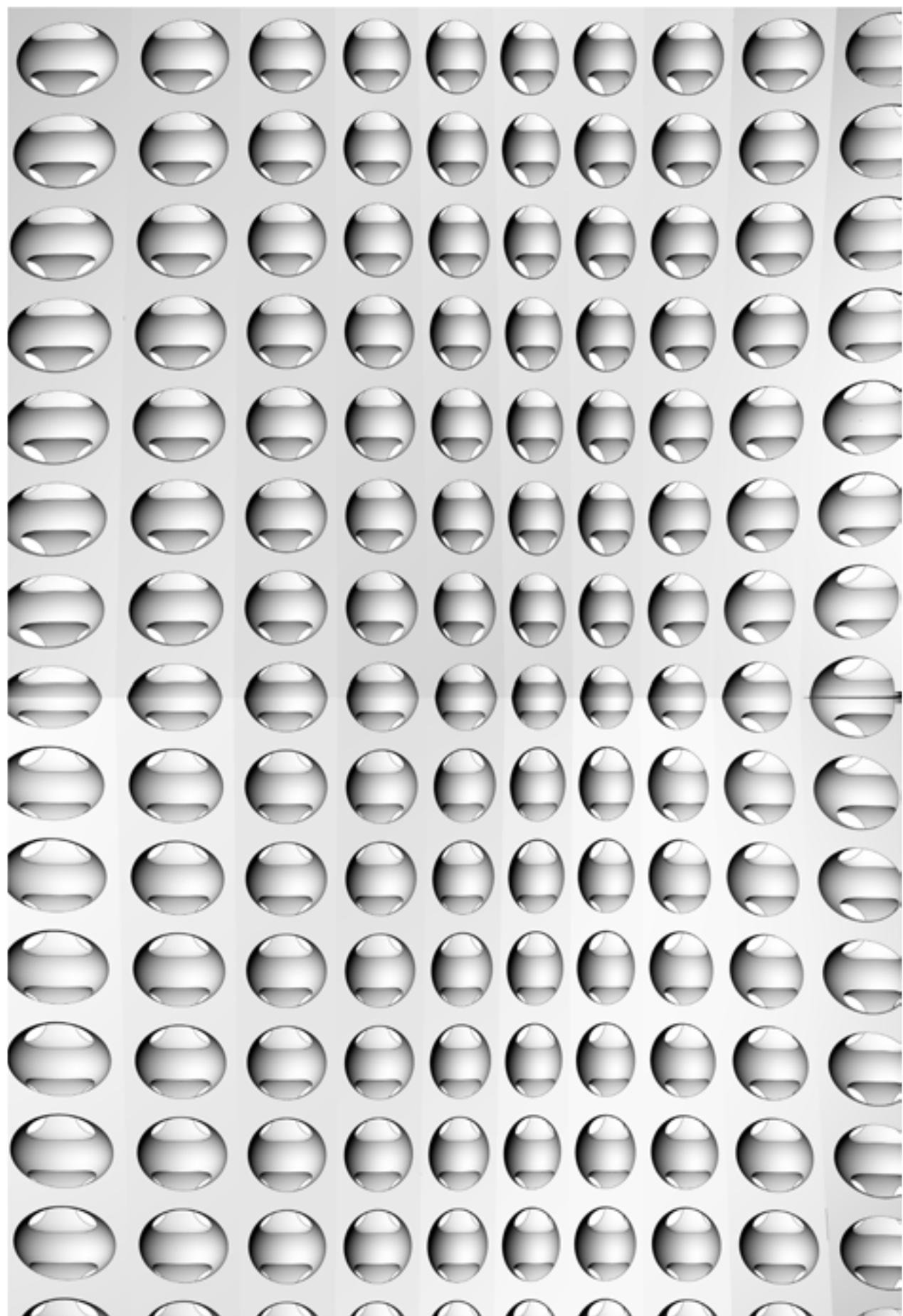
Du musst nun ein Raster aus verzerrten Kisten sehen, die Deine Referenzflaeche bevoelkern. Veraendere die Anzahl U und V Schiebereglern um die Zahl der Kisten zu veraendern und bewege den Hoehenschieberegler um ihre Hoehe anzupassen.

16.	Verbinde den dritten <b>Number Slider</b> mit dem Hoehe (H) Eingabeparameter der <b>Surface Box</b> Komponente	
17.	<b>Surface/Primitive/Bounding Box</b> – Ziehe eine <b>Bounding Box</b> Komponente auf die Leinwand	
18.	<b>Transform/Morph/Box Morph</b> – Ziehe eine <b>Box Morph</b> Komponente auf die Leinwand	
19.	Verbinde den Ausgabeparameter des <b>Geometry</b> Parameter mit dem Inhalt (C) Eingabeparameter der <b>Bounding Box</b> Komponente	
20.	Verbinde den Ausgabeparameter des <b>Geometry</b> Parameter mit dem Geometrie (G) Eingabeparameter der <b>Box Morph</b> Komponente	

21.	Verbinde den Kiste (B) Ausgabaparameter der <b>Bounding Box</b> Komponente mit dem Referenz(R) Eingabeparameter der <b>Box Morph</b> Komponente	
22.	Verbinde den verzerrte Kiste (B) Ausgabeparameter der <b>Surface Box</b> Komponente mit dem Ziel(T) Eingabeparameter der <b>Box Morph</b> Komponente	



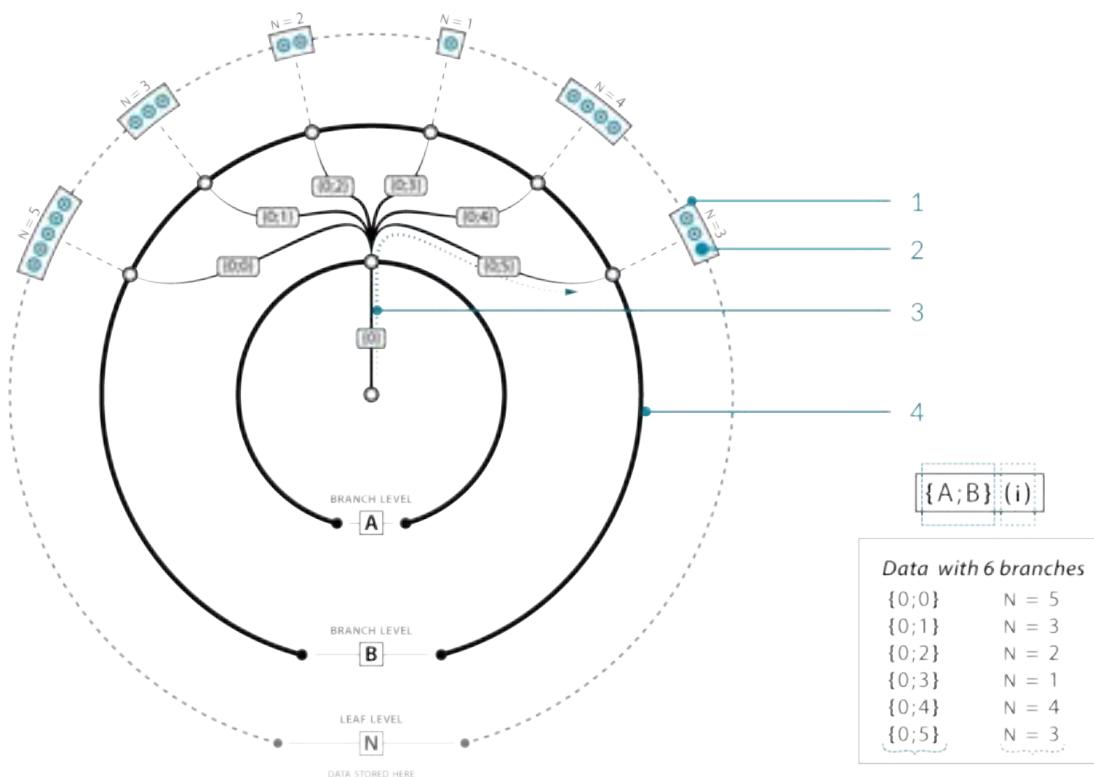
Du solltest nun sehen, wie die Flaeche mit Deiner Geometrie bevoelkert wurde.



### ### 1.5.2. Was ist ein Datenbaum?

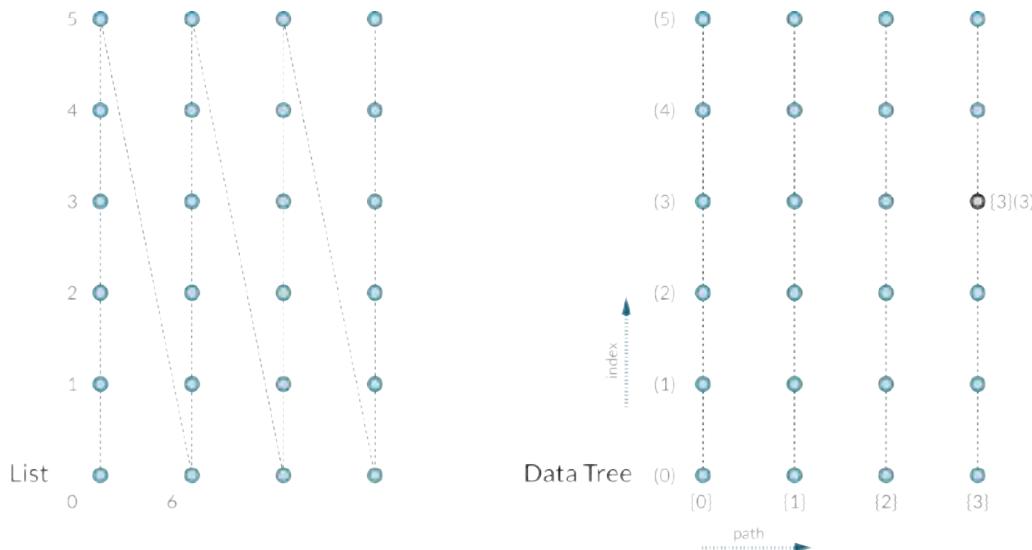
Ein Datenbaum ist eine hierarchische Struktur zur Speicherung von Daten in einer verschachtelten Liste. Datenbaume werden erstellt, wenn Grasshopperkomponenten so strukturiert sind, dass sie einen Datensatz als Eingabe annehmen und mehrere Datensaetze ausgeben. Grasshopper behandelt diese neuen Daten, indem es sie in Form von Unterlisten verschachtelt. Diese verschachtelten Unterlisten verhalten sich in der selben Weise wie Ordnerstrukturen auf Deinem Computer, indem sie den Zugriff auf indizierte Elemente durch eine Pfadnavigation ermöglichen. Diese wird bestimmt durch die Erzeugung von Elternlisten und den entsprechenden Unterindizes.

Es ist möglich mehrere Datenlisten in einem einzelnen Parameter zu speichern. Da mehrere Listen verfügbar werden, benötigen wir eine Möglichkeit die individuellen Listen zu identifizieren. Ein Datenbaum ist essentiell für die Navigation von Listen von Listen oder manchmal auch Listen von Listen von Listen (usw.).



In dem oberhalb dargestellten Bild befindet sich ein einzelner Grundast (Du kannst diesen Stamm nennen, aber da es möglich ist mehrere Grundaeste zu haben, hinkt dieser Vergleich etwas) auf dem Pfad {0}. Dieser Pfad enthält keine Daten, hat aber 6 Unteraeste. Jeder dieser Unteraeste teilt den Index des Elternastes {0} und hat seinen eigenen Unterindex (0, 1, 2, 3, 4, und 5 entsprechend). Es würde falsch sein diese "Index" zu nennen, da sich dieses Wort auf eine bestimmte Zahl bezieht. Es ist wahrscheinlich besser diese als Pfad zu beschreiben, da es die Idee der Ordnerstruktur auf dem Computer wiederspiegelt. An jedem dieser Unteraeste werden wir Daten vorfinden. Jedes Datenelement ist nun also Teil eines (und nur eines) Astes des Baums, und jedes Element hat einen spezifischen Index für seine Position innerhalb des Astes. Jeder Ast hat einen Pfad, der seine Position im Baum beschreibt.

Das Bild unterhalb illustriert den Unterschied zwischen Liste und Datenbaum. Auf der linken Seite wird eine Reihe von fünf Spalten zu je sechs Punkten innerhalb einer Liste dargestellt. Die erste Spalte ist mit 0-5 nummeriert, die zweite mit 6-11, usw. Auf der rechten Seite ist die selbe Reihe von Punkten in einem Datenbaum enthalten. Dieser Datenbaum ist eine Liste der vier Spalten und jede Spalte ist eine Liste von sechs Punkten. Der Index eines jeden Punktes ist (Spaltennummer, Zeilennummer). Dies ist ein viel nützlicherer Ansatz zur Organisation von Daten, weil Du einfacher auf die Daten von allen Punkten einer bestimmten Spalte oder Zeile zugreifen, jede zweite Reihe von Punkten löschen, sich verändernde Punkte verbinden kannst, usw.

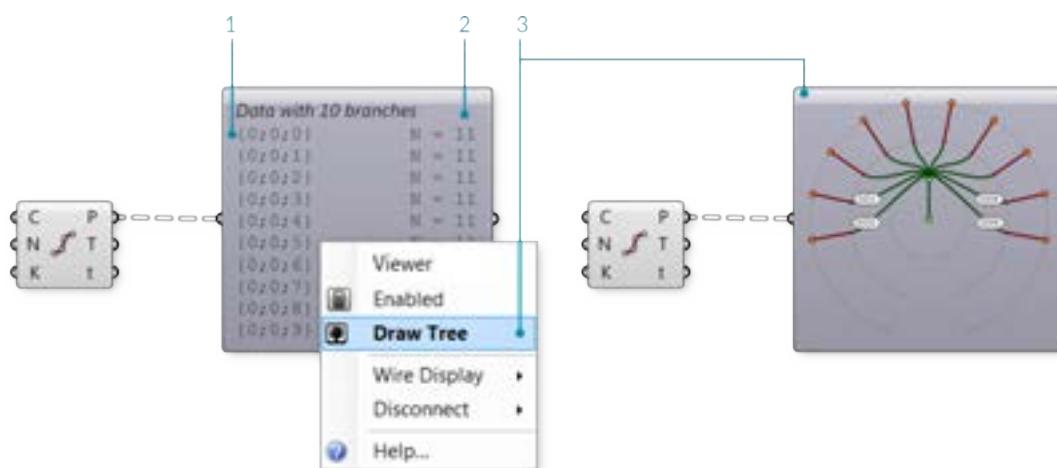


### 1.5.2.1. VISULISIERUNG VON DATENBAEUMEN

Beispieldateien fuer diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

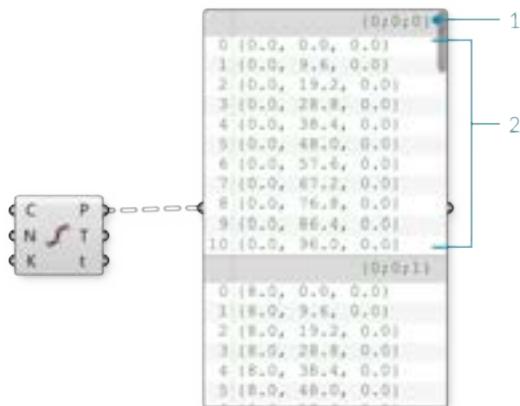
Wegen der Komplexitaet koennen Datenbaeume schwierig zu verstehen sein. Grasshopper hat einige Werkzeuge zur Visualisierung von in Baeumen gespeicherten Daten, um diese besser zu verstehen.

**The Param Viewer** Der "Param Viewer" (Params/Util/Param Viewer) ermoeglicht es die Daten eines Baumes in Bild oder Text darzustellen. Verbinde einen Ausgabeparameter, der Daten enthaelt, mit dem Eingabeparameter des "Param Viewer". Um den Baum anzuzeigen, rechtsklicke auf den "Param Viewer" und waehle "Draw Tree". In diesem Beispiel werden wir den "Param Viewer" mit dem Punkte (P) Ausgabeparameter der "Divide Curve" Komponente verbinden, der 10 Kurven jeweils in 10 Teile zerlegt.



1. Pfad jeder Liste
2. Anzahl von Elementen in jeder Liste
3. Waehle "Draw Tree" um den Datenbaum darzustellen

Wenn wir ein Paneel mit dem selben Ausgabeparameter verbinden, zeigt es zehn Listen mit je elf Elementen an. Du kannst sehen, dass jedes Element ein Punkt ist, der durch drei Koordinaten beschrieben wird. Der Pfad wird an der Spitze einer jeden Liste angezeigt und entspricht dem Pfad der im "Param Viewer" aufgefuehrt ist.

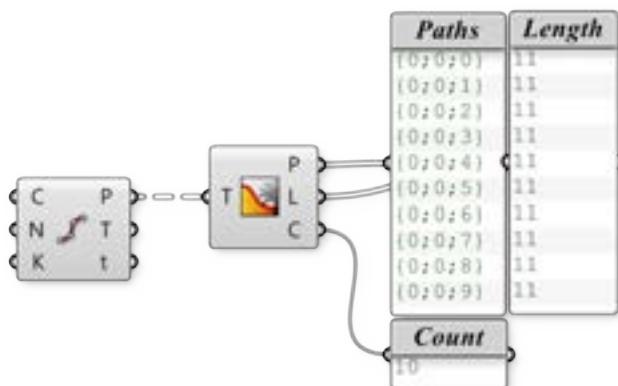


1. Pfad
2. Liste mit 11 Elementen

**Baumstatistiken** Die "Tree Statistics" Komponente (Sets/Tree/Tree Statistics) gibt einige Statistiken zu Datenbaeumen aus. Diese beinhalten:

- P - Alle Pfade des Baumes
- L - Die Laenge jedes Astes im Baum
- C - Anzahl der Pfade und Aeste in einem Baum

Wenn wir dem Punkt Ausgabeparameter der selben "Divide Curve" Komponente anschliessen, koennen wir die Pfade, die Laenge und die Anzahl der Pfade in einem Panel darstellen. Diese Komponente ist hilfreich, da sie die Statistiken in drei Ausgabeparameter unterteilt, was es uns erlaubt nur die relevanten Daten darzustellen.



Der "Param Viewer" und die "Tree Statistics" Komponente sind beide hilfreich fuer die Visulisierung von Veraenderungen innerhalb der Struktur eines Datenbaums. Im naechsten Abschnitt werden wir uns die Operationen ansehen, die angewendet werden koennen um die Struktur von Baeumen zu veraendern.

### ### 1.5.3. Erstellen von Datenbaeumen

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

**Grasshopper enthält Werkzeuge um die Struktur von Datenbaeumen zu veraendern. Diese Werkzeuge helfen auf spezifische Daten innerhalb des Baumes zuzugreifen und die Art in der sie gespeichert, geordnet oder identifiziert sind zu veraendern.**

Lass uns einige Moeglichkeiten zur Manipulation und Visualisierung von Datenbaeumen und ihre Wirkungsweise ansehen.

#### 1.5.3.1. EINEBNEN VON DATENBAEUMEN

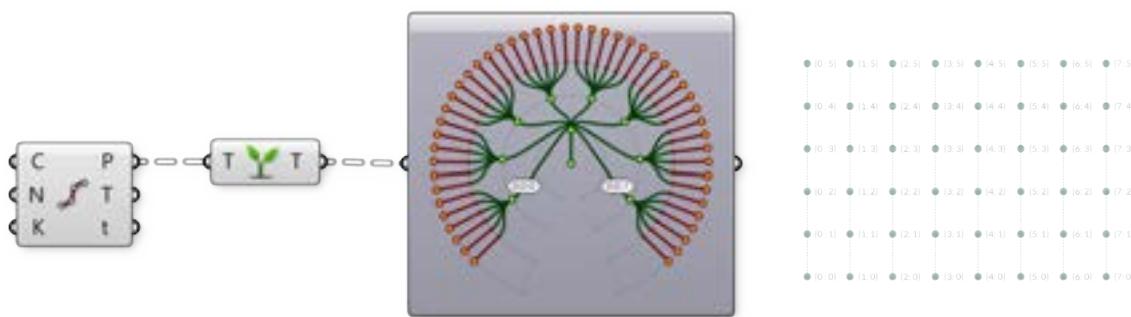
Einebnen von Datenbaeumen entfernt alle Ebenen eines Datenbaumes und ergibt eine einfache Liste. Wende die "Flatten" Komponente (Sets/Tree/Flatten) auf den P Ausgabeparameter der "Divide Curve" Komponente an und visualisiere die neue Datenstruktur mit dem "Param Viewer".



Im "Param Viewer" koennen wir sehen, dass wir nun nur noch einen Ast haben, der aus einer Liste mit 48 Punkten besteht.

#### 1.5.3.2. AUPFROPFEN VON DATENBAEUMEN

Aufprofen erzeugt einen neuen Ast fuer jedes Datenelement. Wenn wir die Daten durch die "Graft Tree" Komponente (Sets/Tree/Graft Tree) senden, wird jeder Teilungspunkt einen individuellen Ast bilden, anstatt einen Ast mit den anderen Teilungspunkten auf der selben Kurve zu bilden.

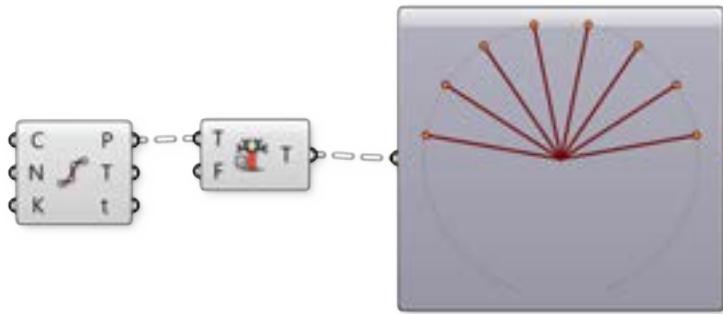


Im "Param Viewer" koennen wir nun sehen, dass wir anstatt einer Datenstruktur mit 8 Aesten zu je 6 Elementen nun 8 Aeste mit je sechs Unteraesten haben, die jeweils ein Element enthalten.

#### 1.5.3.3. VEREINFACHEN VON DATENBAEUMEN

Das vereinfachen von Datenbaeumen entfernt ueberlappende Aeste eines Datenbaumes. Wenn wir die Daten zur "Simplify Tree" Komponente (Sets/Tree/Simplify Tree) senden, wird der erste Ast, der keine Daten enthaelt, entfernt

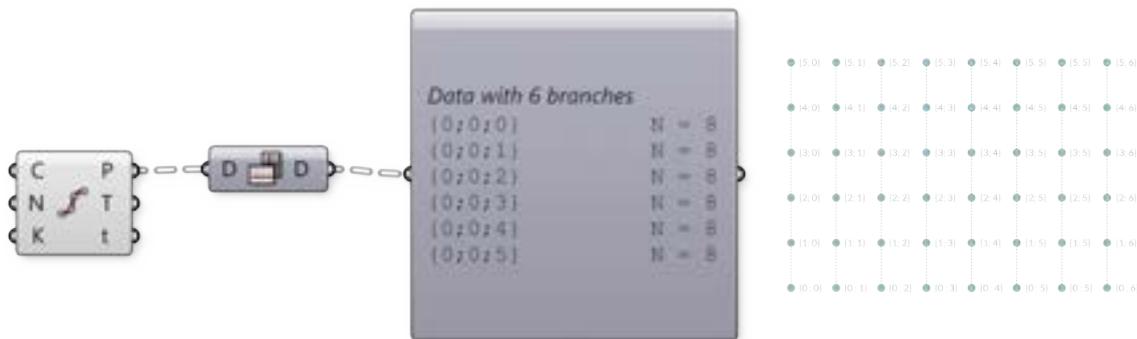
werden.



In "Param Viewer" sehen wir immer noch 8 Äste zu je 6 Elementen, aber der erste Ast wurde entfernt.

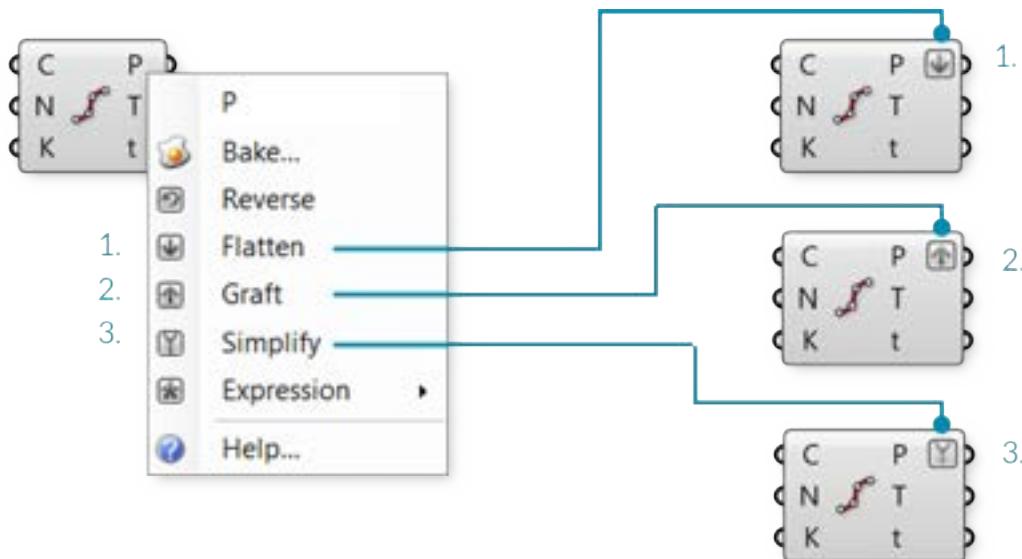
#### 1.5.3.4. DREHEN DER MATRIX

Die "Flip Matrix" Komponente (Sets/Tree/Flip Matrix) vertauscht die Zeilen und Spalten eines Datenbaumes mit zwei Pfadebenen.



In "Param Viewer" koennen wir sehen, dass anstatt der 8 Ästen mit 6 Elementen nun 6 Äste zu je 8 Elementen vorliegen.

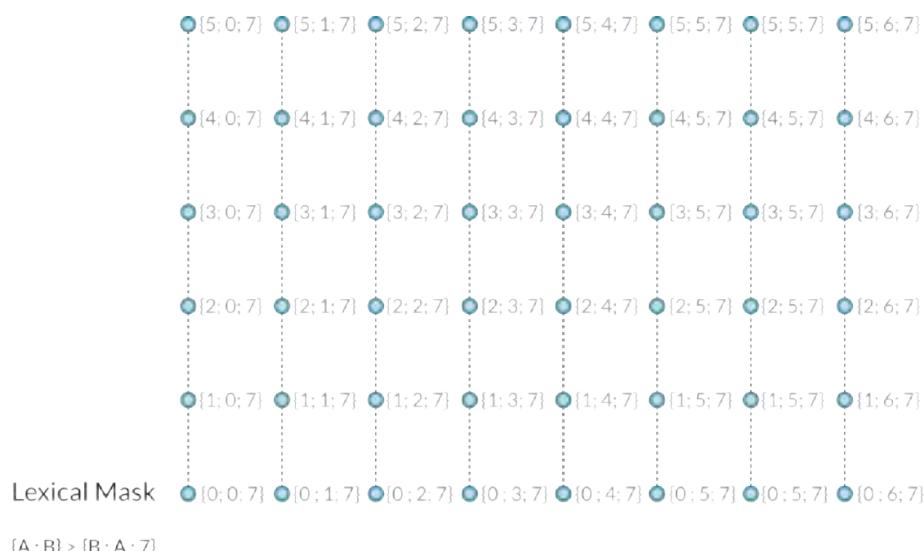
Das einebnen, aufprofen und vereinfachen sind Operationen die auch direkt auf die Eingabe- und Ausgabeparameter angewendet werden koennen, anstatt die Daten durch eine separate Komponente zu schleifen. Rechtsklicke einfach auf den gewuenschten Eingabe- oder Ausgabeparameter und waehle entsprechend "Flatten", "Graft", oder "Simplify" aus dem Menu. Die Komponente wird ein Symbol darstellen, das anzeigen, dass der Baum modifiziert wurde. Denke an den Programmfluss von Grasshopper. Wenn Du einen Komponenteneingabeparameter einebnest, dann werden die Daten vor der eigentlichen Operation der Komponente eingeebnet. Wenn Du die Ausgabekomponente einebnest, dann werden die Daten erst nach Ausfuehrung der Funktion der Komponente eingeebnet.

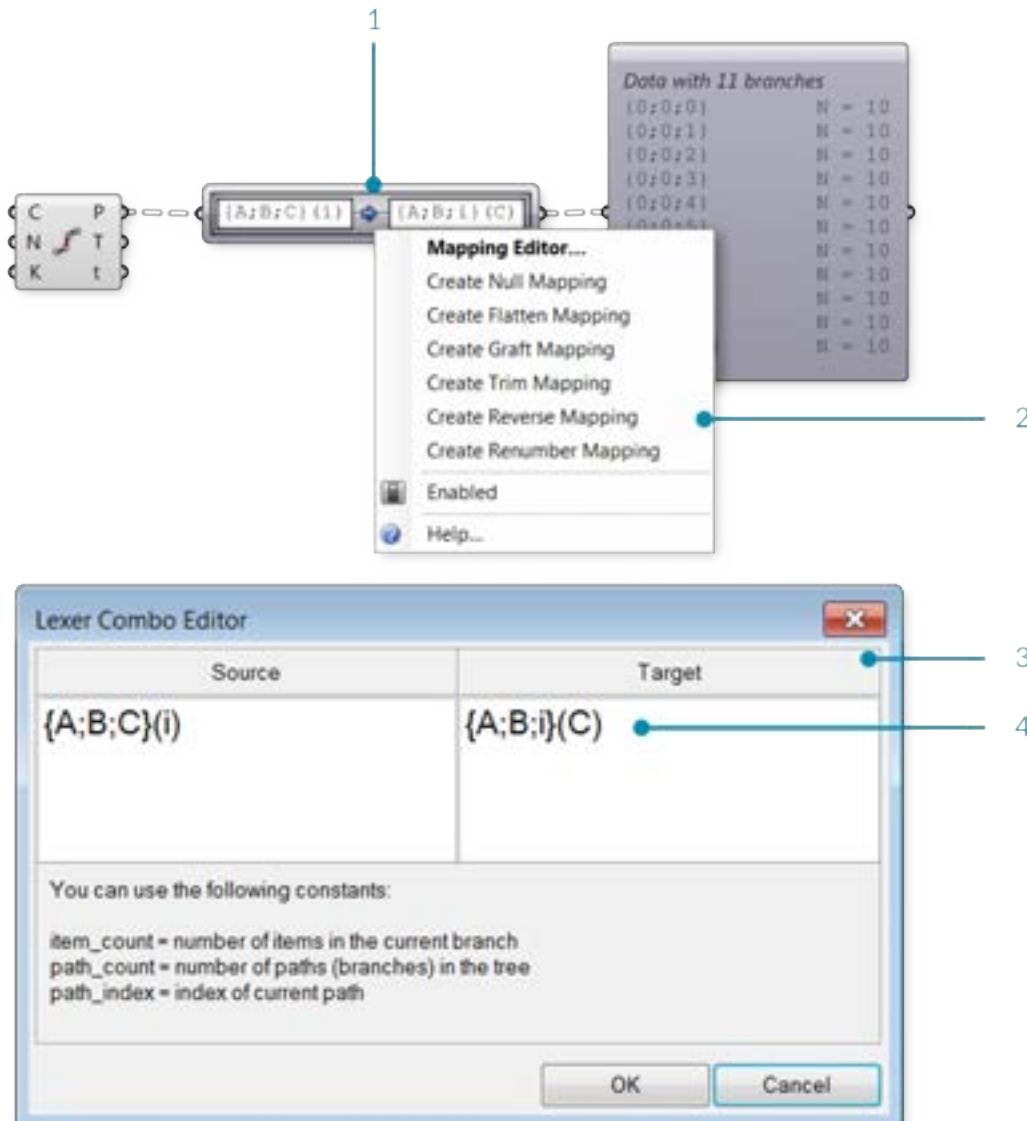


1. Einebnen des Ausgabeparameters P
2. Aufpropfen des Ausgabeparameters P
3. Vereinfachen des Ausgabeparameters P

### 1.5.3.5. DER PATH MAPPER

Die "Path Mapper" Komponente (Sets/Tree/Path Mapper) erlaubt es lexikale Operationen auf Datenbaeume anzuwenden. Lexikale Operationen sind logische Abbildungsverfahren zwischen Dateipfaden und Indizes, die mit text-basierten (lexikalischen) Masken und Mustern definiert werden.



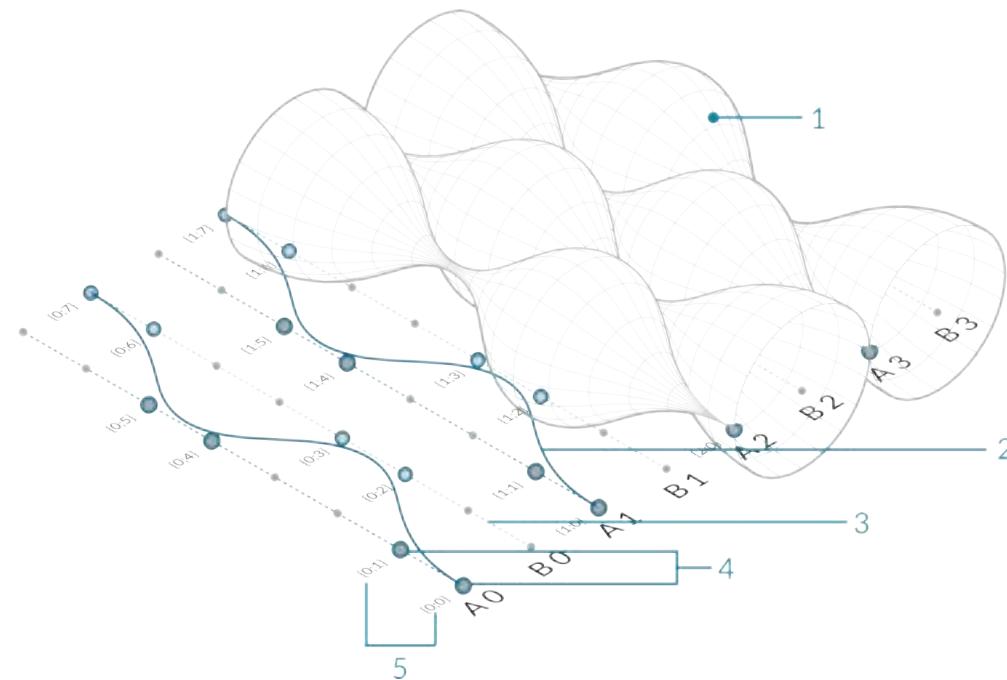


1. Die "Path Mapper" Komponente
2. Rechtsklicke auf die "Path Mapper" Komponente und wähle eine vordefinierte Abbildungsoption aus dem Menü oder öffne den Abbildungseditor.
3. Der Abbildungseditor
4. Du kannst den Datenbaum verändern, indem Du die Pfadindizes und gewünschten Äste neu zuweist

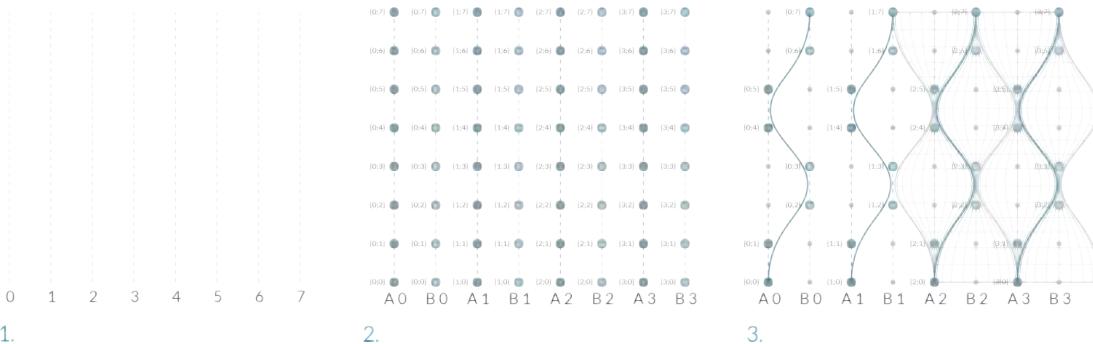
### 1.5.3.6. WEBEN DEFINITION

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

In diesem Beispiel werden wir Listen und Datenbäume manipulieren um Listen von Punkten miteinander zu verweben, um damit ein Muster und eine Flächengeometrie zu erzeugen.



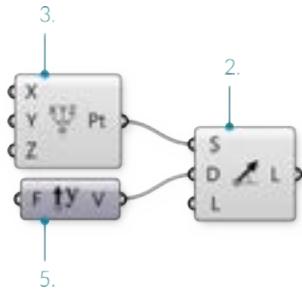
1. Rotierte NURBS Flaeche
2. NURBS Kurve
3. Kurvenarray
4. Teilungspunkte
5. Pfade (Indizes) von Punkten



1. Kurven als Reihe anordnen
2. Teile die Kurven in zwei Listen A und B und unterteile die Kurven
3. Entferne Punkte, verwebe die Listen und rotiere die Flaeche

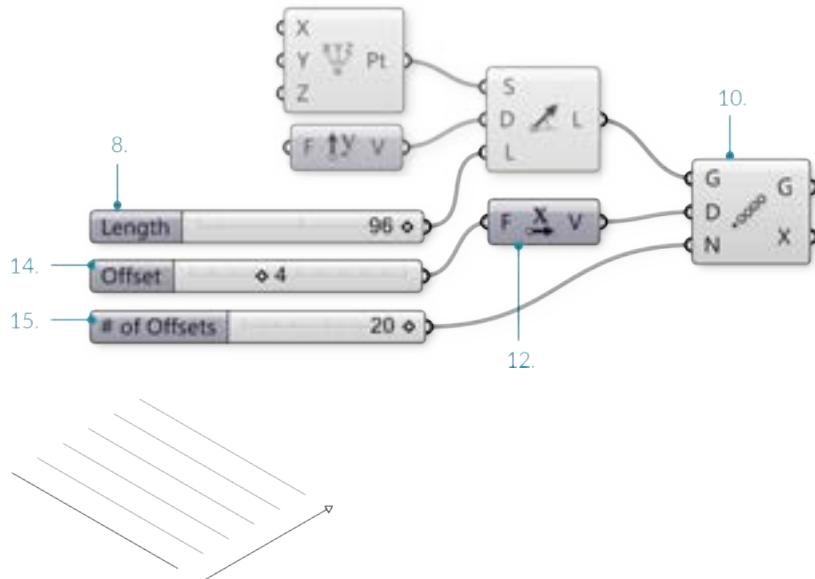
01.	Beginne eine neue Definition, druecke Strg+N (in Grasshopper)	
02.	<b>Curve/Primitive/Line SDL</b> – Ziehe eine <b>Line SDL</b> Komponente auf die Leinwand	
03.	<b>Vector/Point/Construct Point</b> – Ziehe eine <b>Construct Point</b> Komponente auf die Leinwand	
04.	Verbinde den Punkte (Pt) Ausgabeparameter der <b>Construct Point</b> Komponente mit dem Start (S) Eingabeparameter der <b>Line SDL</b> Komponente	
05.	<b>Vector/Vector/Unit Y</b> – Ziehe eine <b>Unit Y</b> Vektorkomponente auf die Leinwand Der Faktor der Einheitsvektor Y Komponente ist 1.0 als Standard.	

05.	Der Faktor der Einheitsvektor Y Komponente ist 1.0 als Standard.	
06.	Verbinde die <b>Unit Y</b> Komponente mit dem Richtung (D) Eingabeparameter der <b>Line SDL</b> Komponente	



07.	<b>Params/Input/Number Slider</b> – Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand	
08.	Doppelklicke den <b>Number Slider</b> und setze folgende Werte: Name: Length Rounding: Integer Lower Limit: 0 Upper Limit: 96 Value: 96	
09.	Verbinde den <b>Number Slider</b> mit dem Laenge (L) Eingabeparameter der <b>Line SDL</b> Komponente	
10.	<b>Transform/Array/Linear Array</b> – Ziehe eine <b>Linear Array</b> Komponente auf die Leinwand	
11.	Verbinde den Linie (L) Ausgabeparameter der <b>Line SDL</b> Komponente mit dem Geometrie (G) Eingabeparameter der <b>Linear Array</b> Komponente	
12.	<b>Vector/Vector/Unit X</b> – Ziehe eine Einheitsvektor X <b>Unit X</b> Komponente auf die Leinwand	
13.	<b>Params/Input/Number Slider</b> – Ziehe zwei <b>Number Slider</b> Komponenten auf die Leinwand	
14.	Doppelklicke auf den ersten <b>Number Slider</b> und setze folgende Werte: Name: Offset Distance Rounding: Integer Lower Limit: 1 Upper Limit: 10 Value: 4	
15.	Doppelklicke den zweiten <b>Number Slider</b> und setze die folgenden Werte: Name: # of Offsets Rounding: Even Lower Limit: 2 Upper Limit: 20 Value: 20	
16.	Verbinde den <b>Number Slider</b> (Versatzdistanz) mit dem Faktor (F) Eingabeparameter der <b>Unit X</b> Komponente	
	Verbinde den Vektor (V) Ausgabeparameter der <b>Unit X</b> Komponente mit dem	

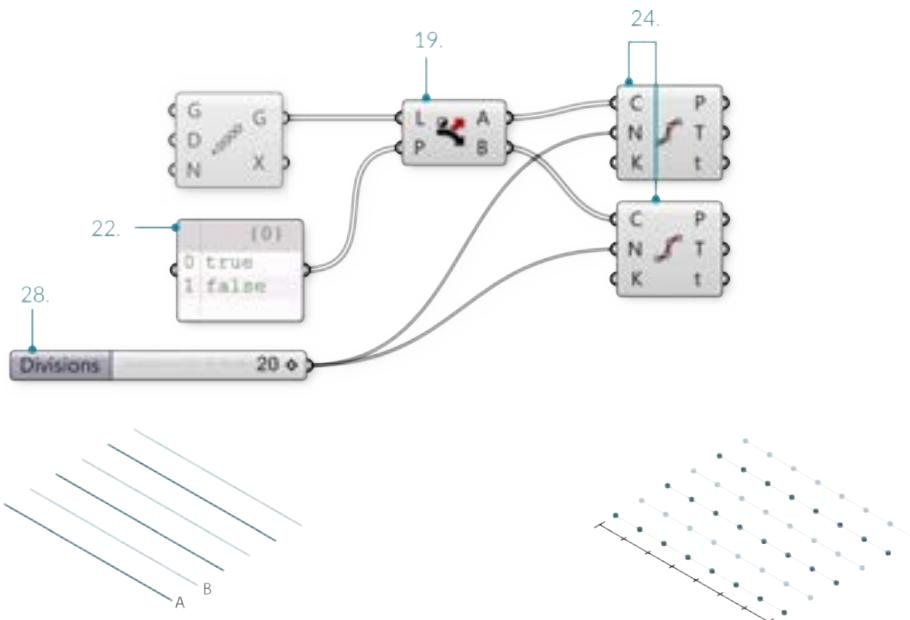
17.	Verbinde den Vektor (V) Ausgabeparameter der <b>Unit X</b> Komponente mit dem Richtung (D) Eingabeparameter der <b>Linear Array</b> Komponente	
18.	Verbinde den <b>Number Slider</b> (Anzahl der Versatzaktionen) mit dem Anzahl (N) Eingabeparameter der <b>Linear Array</b> Komponente	



Du solltest nun eine Reihe von Linien in Deinem Rhinoansichtsfenster sehen. Die drei Schieberegler erlauben es Dir die Laenge der Linien, deren Abstand zueinander und die Anzahl der Linien in der Reihe zu veraendern.

19.	<b>Sets/Lists/Dispatch</b> – Ziehe eine <b>Dispatch</b> Komponente auf die Leinwand	
20.	Verbinde den Geometrie (G) Ausgabeparameter der <b>Linear Array</b> Komponente mit dem Liste (L) Eingabeparameter der <b>Dispatch</b> Komponente	
21.	<b>Params/Input/Panel</b> – Ziehe eine <b>Panel</b> Komponente auf die Leinwand	
22.	Doppelklicke das <b>Panel</b> , entferne die Auswahl fuer "Multiline Data", "Wrap Items" und "Special Codes", und gebe folgendes ein: true false	
23.	Verbinde das <b>Panel</b> mit der Muster (P) Eingabeparameter der <b>Dispatch</b> Komponente	
24.	<b>Curve/Division/Divide Curve</b> – Ziehe zwei <b>Divide Curve</b> Komponenten auf die Leinwand	
25.	Verbinde den Liste A (A) Ausgabeparameter der <b>Dispatch</b> Komponente mit dem Kurve (C) Eingabeparameter der ersten <b>Divide Curve</b> Komponente	
26.	Verbinde den Liste B (B) Ausgabeparameter der <b>Dispatch</b> Komponente mit dem Kurve (C) Eingabeparameter der zweiten <b>Divide Curve</b> Komponente	
27.	<b>Params/Input/Number Slider</b> – Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand	
	Doppelklicke auf den <b>Number Slider</b> und setze die folgenden Werte: Name: Divisions Rounding: Integer	

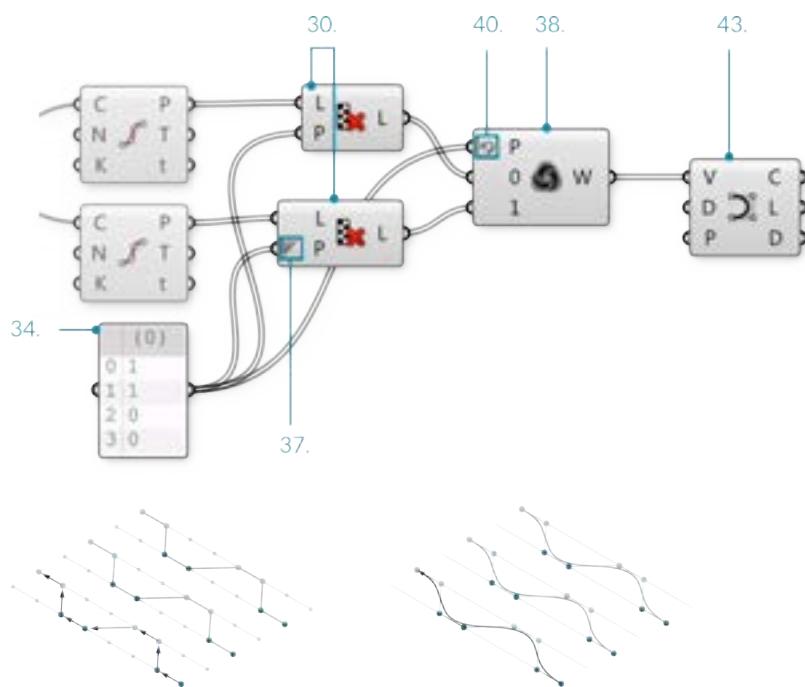
	Value: 20	
29.	Verbinde den <b>Number Slider</b> (Teilungen) mit dem Anzahl (N) Eingabeparameter der beiden <b>Divide Curve</b> Komponenten.	



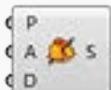
1. Die "Dispatch" Komponente sendet jede zweite Kurve des Arrays in eine separate Liste.
2. Die "Divide Curve" Komponente teilt die Kurven in eine Anzahl von Segmenten die vom Schieberegler bestimmt wird.

30.	<b>Sets/Sequence/Cull Pattern</b> – Ziehe zwei <b>Cull Pattern</b> Komponenten auf die Leinwand	
31.	Verbinde den Punkte (P) Ausgabeparameter der ersten <b>Divide Curve</b> Komponente mit dem Liste (L) Eingabeparameter der ersten <b>Cull Pattern</b> Komponente	
32.	Verbinde den Punkte (P) Ausgabeparameter der zweiten <b>Divide Curve</b> Komponente mit dem Liste (L) Eingabeparameter der zweiten <b>Cull Pattern</b> Komponente	
33.	<b>Params/Input/Panel</b> – Ziehe eine zweite <b>Panel</b> Komponente auf die Leinwand	
34.	Doppelklicke die zweite <b>Panel</b> Komponente und entferne die Auswahl von: "Multiline Data", "Wrap Items", und "Special Codes". Dann gebe Folgendes ein: 1 1 0 0	
	Wir benutzen 0 und 1 anstatt "wahr" und "falsch". Dies sind die beiden Arten boolsche Werte einzugeben, die Grasshopper akzeptiert.	
35.	Verbinde das zweite <b>Panel</b> mit dem Muster (P) Eingabeparameter der ersten <b>Cull Pattern</b> Komponente	
36.	Verbinde das zweite <b>Panel</b> mit dem Muster (P) Eingabeparameter der zweiten <b>Cull Pattern</b> Komponente	
	Rechtsklicke auf den Muster (P) Eingabeparameter der zweiten <b>Cull Pattern</b> Komponente und wähle "Invert"	

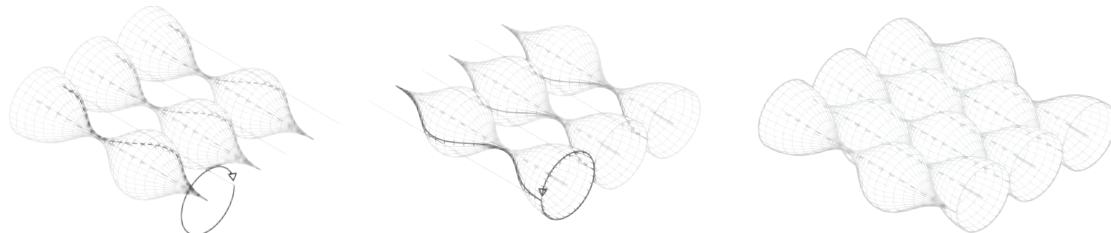
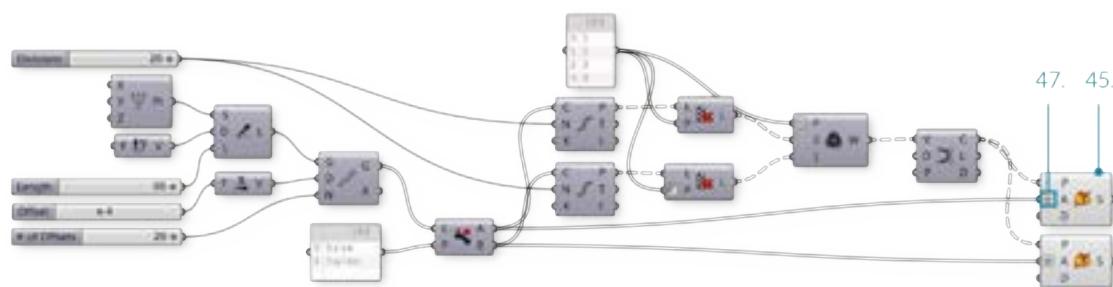
	Komponentw und waehle "Invert" 37. Dies wird die **Cull Pattern** umkehren, ein nuetzlicher Trick um Definitionen kurz zu halten.	
38.	Sets/List/Weave – Ziehe eine <b>Weave</b> Komponente auf die Leinwand	
39.	Verbinde das zweite <b>Panel</b> mit dem Muster (P) Eingabeparameter der <b>Weave</b> Komponente	
40.	Rechtsklicke den Muster (P) Eingabeparameter der <b>Weave</b> Komponente und waehle "Reverse"	
41.	Verbinde den Liste (L) Ausgabeparameter der ersten <b>Cull Pattern</b> Komponente mit dem Strom 0 (0) Eingabeparameter der <b>Weave</b> Komponente	
42.	Verbinde den Liste (L) Ausgabeparameter mit der zweiten <b>Cull Pattern</b> Komponente mit dem Strom 1 (1) Eingabeparameter der <b>Weave</b> Komponente	
43.	Curve/Spline/Nurbs Curve – Ziehe eine <b>Nurbs Curve</b> Komponente auf die Leinwand	
44.	Verbinde den Gewebe (W) Ausgabeparameter der <b>Weave</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Nurbs Curve</b> Komponente.	

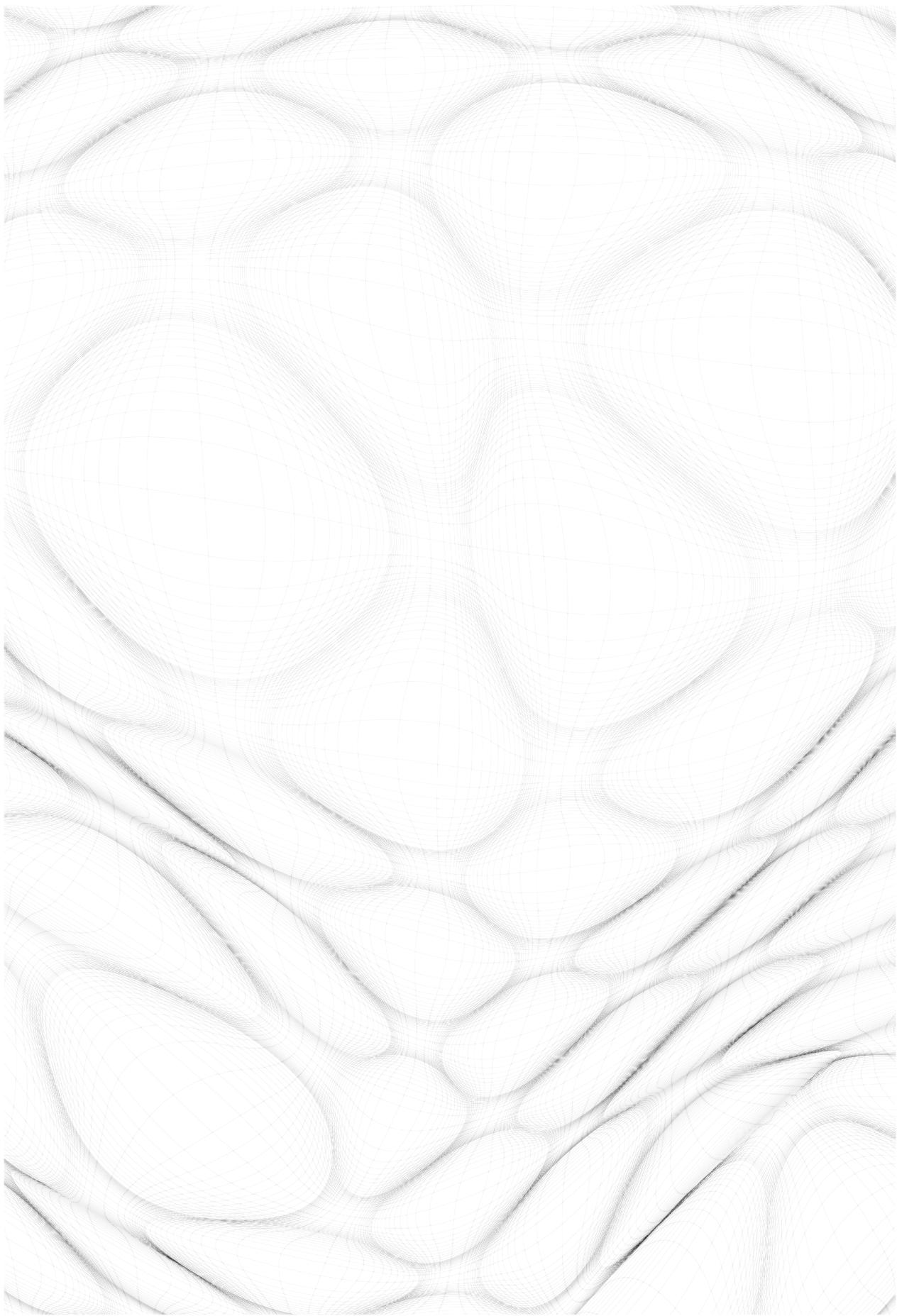


1. Die "Cull Pattern" Komponenten entfernen alternierend Punkte von der jeweiligen Liste.
2. Die "Weave" Komponente sammelt Daten von der Punktliste entsprechend dem benutzerdefinierten Muster. Diese Daten werde in eine "Interpolate Kurve" Komponente eingegeben, um die Kurven zu erstellen.

45.	Surface/Freeform/Revolution – Ziehe zwei <b>Revolution</b> Komponenten auf die Leinwand	
46.	Verbinde den Kurven (C) Ausgabeparameter der <b>Nurbs Curve</b> Komponente mit dem	

46.	Verbinde den Kurven (C) Ausgabeparameter der <b>Nurbs Curve</b> Komponente mit dem Profilkurve (P) Eingabekomponenten der beiden <b>Revolution</b> Komponenten.	
47.	Rechtsklicke auf den Achse (A) Eingabeparameter beider <b>Revolution</b> Komponenten und waehle "Graft".	
48.	Verbinde den Liste A(A) Ausgabeparameter der <b>Dispatch</b> Komponent mit dem Achse (A) Eingabeparameter der ersten <b>Revolution</b> Komponente	
49.	<p>Verbinde den Liste B (B) Ausgabeparameter der <b>Dispatch</b> Komponente mit dem Achse (A) Eingabeparameter der zweiten <b>Revolution</b> Komponente</p> <p>Waehle alle Komponenten ausser den beiden "Revolution" Komponenten und schalte die Grasshoppervorschau aus - es ist hilfreich die Vorschau auszuschalten und waehrend dem Aufbau einer Definition auf den letzten Stand der Geometrie zu fokussieren.</p>	

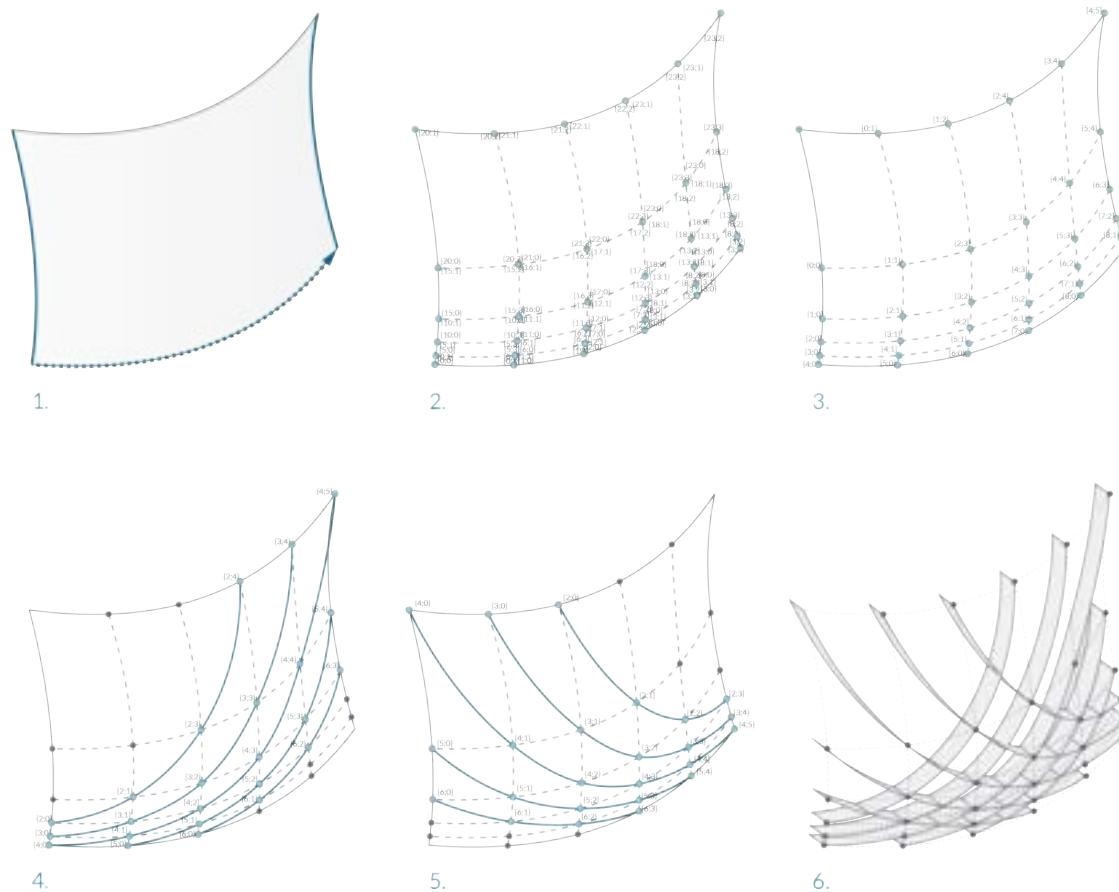




## 1.5.4. Arbeiten mit Datenbaeumen

Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

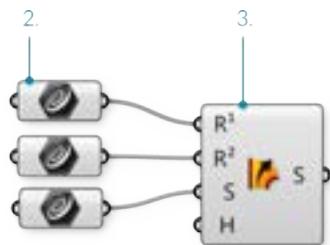
In diesem Beispiel werden wir die Grasshopper Werkzeuge zur Manipulation von Datenbaeumen nutzen um Daten zu entnehmen, neu zu organisieren und zwischen den gewuenschten Punkten zu interpolieren um ein Raumtragwerk aus miteinander verschneidenden Finnen zu erstellen.



1. Ziehe eine Flaeche entlang zwei Schienen auf, um eine NURBS Flaeche zu erhalten.
2. Teile die Flaeche in Segmente variabler Groesse auf und extrahiere die Eckpunkte. Die erstellten Daten bestehen aus einer List mit vier Elementen je Segment.
3. Drehe die Matrix, um die Datenstruktur zu veraendern. Die Daten bestehen nun aus vier Listen mit je einem Eckpunkt pro Element in jeder Liste.
4. Nutze "Explode Tree" um den Datenbaum aufzugliedern und die Punkte miteinander zu verbinden, um die Diagonlen eines jeden Elements zu zeichnen.
5. Mit "Prune Tree" entferne die Aeste mit unzureichenden Punkten um eine NURBS Kurve mit Grad 3 zu erstellen und interpoliere die Punkte.
6. Extrudiere die Kurve, um die verschneidenden Finnen zu erstellen.

01.	Beginne eine neue Definition, druecke Strg+N (in Grasshopper)	
02.	Params/Geometry/Curve – Ziehe drei <b>curve</b> Parameter auf die Leinwand	

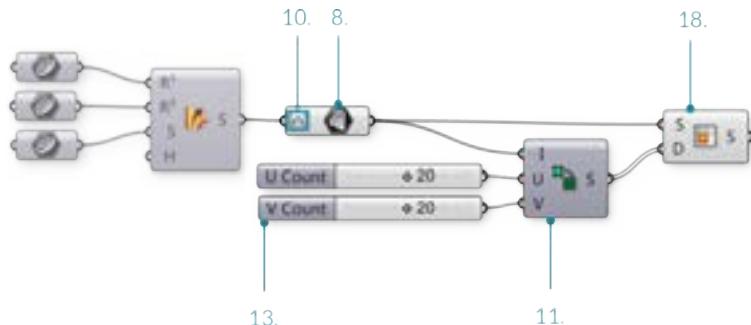
03.	Surface/Freeform/Sweep2 – Ziehe eine <b>Sweep2</b> Komponente auf die Leinwand	
04.	Rechtsklicke den ersten <b>Curve</b> Parameter und waelle “Set one curve.” Waelle die erste Schienenkurve im Rhinoansichtsfenster aus	
05.	Rechtsklicke den zweiten <b>Curve</b> Parameter und waelle “Set one curve.” Waelle die zweite Schienenkurve im Rhinoansichtsfenster aus	
06.	Rechtsklicke auf den dritten <b>Curve</b> Parameter und waelle “Set one curve.” Waelle die Schnittkurve im Rhinoansichtsfenster aus	
07.	Verbinde den Ausgabeparameter der <b>Curve</b> Parameter mit den Schiene 1 (R1), Schiene 2 (R2), und Schnittkurve (S) Eingabeparameter der <b>Sweep2</b> Komponente entsprechend	



Wir haben gerade eine NURBS Flaeche erstellt.

08.	Params/Geometry/Surface – Ziehe einen <b>Surface</b> Parameter auf die Leinwand	
09.	Verbinde den Brep (S) Ausgabeparameter der <b>Sweep2</b> Komponente mit dem Eingabeparameter des <b>Surface</b> Parameter	
10.	Rechtsklicke auf den <b>Surface</b> Parameter und waelle “Reparameterize”. In diesem Schritt, bilden wir die u und v Domaene der Flaeche auf eine Domaene von 0 bis 1 ab. Dies wird die zukuenftigen Operationen erleichtern.	
11.	Maths/Domain/Divide Domain2 – Ziehe eine <b>Divide Domain2</b> Komponente auf die Leinwand	
12.	Params/Input/Number Slider – Ziehe zwei <b>Number Sliders</b> auf die Leinwand	
13.	Doppelklicke den ersten <b>Number Sliders</b> und setze die folgenden Werte: Rounding: Integer Lower Limit: 1 Upper Limit: 40 Value: 20	
14.	Setze die selben Werte beim zweiten Number Sliders	
15.	Verbinde den Ausgabeparameter des reparametrisierten <b>Surface</b> Parameters mit dem Domaene (I) Eingabeparameter der <b>Divide Domain2</b> Komponente	
16.	Verbinde den ersten <b>Number Sliders</b> mit dem Anzahl U (U) Eingabeparameter der <b>Divide Domain2</b> Komponente	
17.	Verbinde den zweiten <b>Number Sliders</b> mit dem Anzahl V (V) Eingabeparameter der <b>Divide Domain2</b> Komponente	

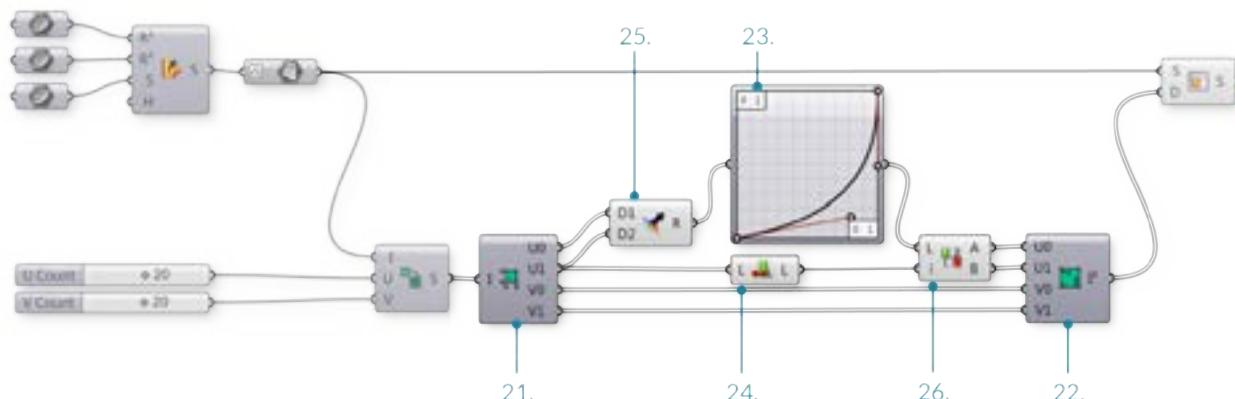
18.	Surface/Util/Isotrim – Ziehe eine <b>Isotrim</b> Komponente auf die Leinwand	
19.	Verbinde den Segmente (S) Ausgabeparameter der <b>Divide Domain2</b> Komponente mit dem Domaene (D) Eingabeparameter der <b>Isotrim</b> Komponente	
20.	Verbinde den Ausgabeparameter des <b>Surface</b> Parameter mit dem Flaeche (S) Eingabeparameter der <b>Isotrim</b> Komponente	



Wir haben jetzt eine Flaeche in kleinere, gleich grosse Flaechen unterteilt. Pass den Anzahl U und V Schieberegler an um die Anzahl der Teilungen zu veraendern. Lass uns einen "Graph Mapper" hinzufuegen, um den Segmenten eine variable Groesse zu geben.

21.	Maths/Domain/Deconstruct Domain2 – Ziehe eine <b>Deconstruct Domain2</b> Komponente auf die Leinwand	
22.	Maths/Domain/Construct Domain2 – Ziehe eine <b>Construct Domain2</b> Komponente auf die Leinwand	
23.	Params/Input/Graph Mapper – Ziehe einen <b>Graph Mapper</b> auf die Leinwand	
24.	Sets/List/List Length – Ziehe eine <b>List Length</b> Komponente auf die Leinwand	
25.	Sets/Tree/Merge – Ziehe eine <b>Merge</b> Komponente auf die Leinwand	
26.	Sets/List/Split List – Ziehe eine <b>Split List</b> Komponente auf die Leinwand Die "Merge" und "Split" Komponenten sind hier so angeordnet, dass sie den selben "Graph Mapper" fuer die U min und U max Werte nutzen koennen.	
27.	Verbinde den U min (U0) und U max (U1) Ausgabeparameter der <b>Deconstruct Domain2</b> Komponente mit dem Daten 1 (D1) und Daten 2 (D2) Eingabeparameter der <b>Merge</b> Komponente	
28.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Merge</b> Komponente mit dem Eingabeparameter des <b>Graph Mapper</b>	
29.	Rechtsklicke auf den <b>Graph Mapper</b> und waehle "Bezier" unter "Graph Types"	
	Verbinde ein zweites Kabel vom U max (U1) Ausgabeparameter der <b>Deconstruct</b>	

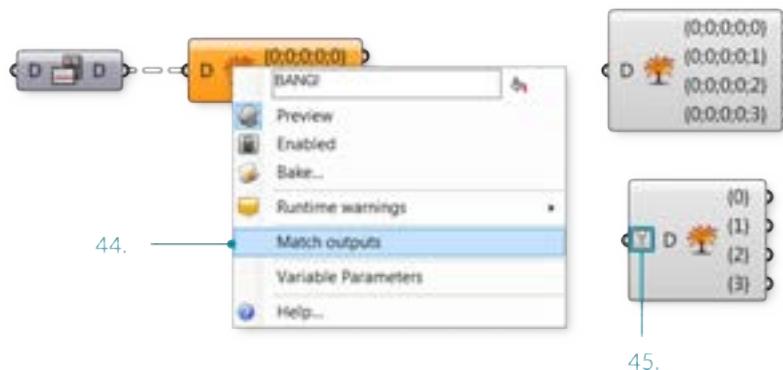
30.	Domain2 Komponente mit dem Liste (L) Eingabeparameter der <b>List Length</b> Komponente	
31.	Verbinde den <b>Graph Mapper</b> Ausgabeparameter mit dem Liste (L) Eingabeparameter der <b>Split List</b> Komponente	
32.	Verbinde den Laenge (L) Ausgabeparameter der <b>List Length</b> Komponente mit dem Index (i) Eingabeparameter der <b>Split List</b> Komponente	
33.	Verbinde den Liste A (A) Ausgabeparameter der <b>Split List</b> Komponente mit dem U min (U0) Eingabeparameter der <b>Construct Domain2</b> Komponente	
34.	Verbinde den Liste B (B) Ausgabeparameter der <b>Split List</b> Komponente mit dem U max (U1) Eingabeparameter der <b>Construct Domain2</b> Komponente	
35.	Verbinde den V min (V0) Ausgabeparameter der <b>Deconstruct Domain2</b> Komponente mit dem V min (V1) Eingabeparameter der <b>Construct Domain2</b> Komponente	
36.	Verbinde den V max (V1) Ausgabeparameter der <b>Deconstruct Domain2</b> Komponente mit dem V max (V1) Eingabeparameter der <b>Construct Domain2</b> Komponente	
37.	Verbinde den 2D Domain (I2) Ausgabeparameter der <b>Construct Domain2</b> Komponente mit dem Domaene (D) Eingabeparameter der <b>Isotrim</b> Komponente und ersetze dabei die bestehenden Verbindungen	



Wir haben gerade die Domaenen jedes Flaechenelements zerlegt, die U Werte mit dem "Graph Mapper" neu abgebildet und die Domaenen aus den Bestandteilen neu erstellt. Passe die Griffe der "Graph Mapper" Komponente an, um die Verteilung der Flaechenelemente zu veraendern. Lass uns die Datenbaeume der Flaechenteilung nun veraendern und sehen, was passiert.

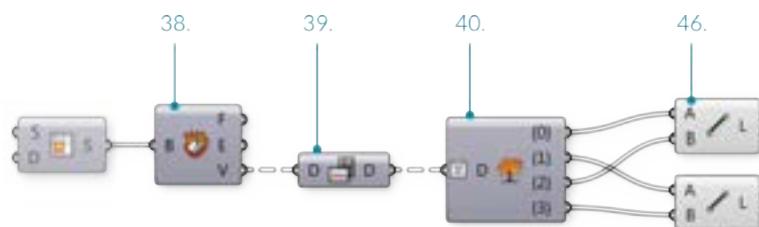
38.	Surface/Analysis/Deconstruct Brep – Ziehe eine <b>Deconstruct Brep</b> Komponente auf die Leinwand	
39.	Sets/Tree/Flip Matrix – Ziehe eine <b>Flip Matrix</b> Komponente auf die Leinwand	
40.	Sets/Tree/Explode Tree – Ziehe eine <b>Explode Tree</b> Komponente auf die Leinwand	
41.	Verbinde den Flaeche (S) Ausgabeparameter der <b>Isotrim</b> Komponente mit dem Brep (B) Eingabeparameter der <b>Deconstruct Brep</b> Komponente Die "Deconstruct Brep" Komponente zerlegt einen Brep in Oberflaechen, Kanten und Eckpunkte. Dies ist hilfreich, wenn Du einen spezifischen Bestandteil einer Flaeche bearbeiten willst.	

	Verbinde den Eckpunkte (V) Ausgabeparameter der <b>Deconstruct Brep</b> Komponente mit dem Daten (D) Eingabeparameter der <b>Flip Matrix</b> Komponente	
42.	Wir haben gerade die Datenstruktur von einer Liste mit vier Eckpunkten die eine Fläche definieren, zu vier Listen mit je einem Eckpunkt jeder Fläche transformiert.	
43.	Verbinde den Daten (D) Ausgabeparameter der <b>Flip Matrix</b> Komponente mit dem Daten (D) Eingabeparameter der <b>Explode Tree</b> Komponente	
44.	Rechtsklicke auf die <b>Explode Tree</b> Komponente und wähle "Match Outputs"	
45.	Rechtsklicke auf den Daten (D) Eingabeparameter der <b>Explode Tree</b> Komponente und wähle "simplify"	



Jeder Ausgabeparameter der "Explode Tree" Komponente enthält eine Liste mit einem Eckpunkt jeder einzelnen Fläche. In anderen Worten, eine Liste mit allen Ecken rechts oben, eine Liste mit allen Ecken rechts unten, eine Liste mit allen Ecken links unten und eine Liste mit allen Ecken links oben.

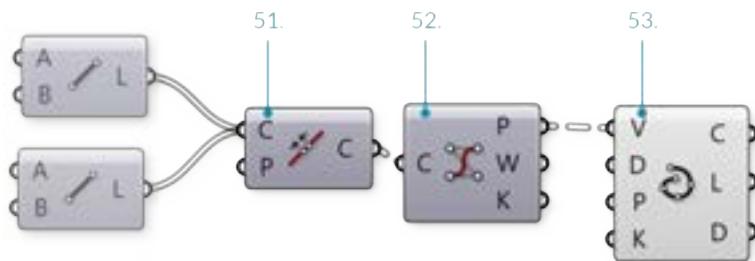
46.	Curve/Primitive/Line – Ziehe eine <b>Line</b> Komponente auf die Leinwand	
47.	Verbinde den Ast 0 {0} Ausgabeparameter der <b>Explode Tree</b> Komponente mit dem Startpunkt (A) Eingabeparameter der ersten <b>Line</b> Komponente	
48.	Verbinde den Ast 1 {1} Ausgabeparameter der <b>Explode Tree</b> Komponente mit dem Startpunkt (A) Eingabeparameter der zweiten <b>Line</b> Komponente	
49.	Verbinde den Ast 2 {2} Ausgabeparameter der <b>Explode Tree</b> Komponente mit dem Endpunkt (B) Eingabeparameter der ersten <b>Line</b> Komponente	
50.	Verbinde den Ast 3 {3} Ausgabeparameter der <b>Explode Tree</b> Komponente mit dem Endpunkt (B) Eingabeparameter der zweiten <b>Line</b> Komponente	



Wir haben nun die Eckpunkte einer jeden Fläche diagonal mit Linien verbunden.

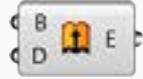
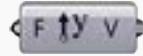
51.	Curve/Util/Join Curves – Ziehe eine <b>Join Curves</b> Komponente auf die Leinwand	
-----	--	--

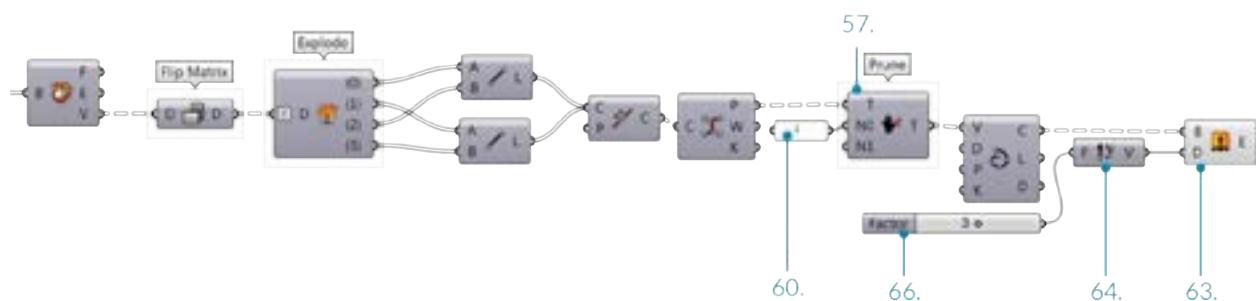
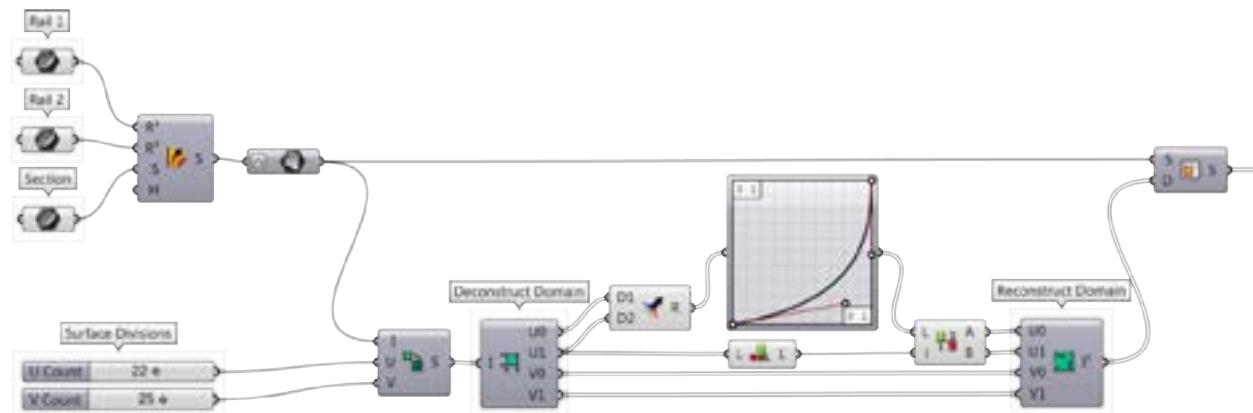
52.	<b>Curve/Analysis/Control Points</b> – Ziehe eine <b>Control Points</b> Komponente auf die Leinwand	
53.	<b>Curve/Spline/Interpolate</b> – Ziehe eine <b>Interpolate</b> Komponente auf die Leinwand	
54.	Verbinde den Linie (L) Ausgabeparameter jeder <b>Line</b> Komponente mit dem Kurven (C) Eingabeparameter der <b>Join Curve</b> Komponente Halte Shift gedreuekt, um mehrere Kabel mit einem Eingabeparameter zu verbinden.	
55.	Verbinde den Kurven (C) Ausgabeparameter der <b>Join Curves</b> Komponente mit dem Kurven (C) Eingabeparameter der <b>Control Points</b> Komponente	
56.	Verbinde den Punkte (P) Ausgabeparameter der <b>Control Points</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Interpolate</b> Komponente	



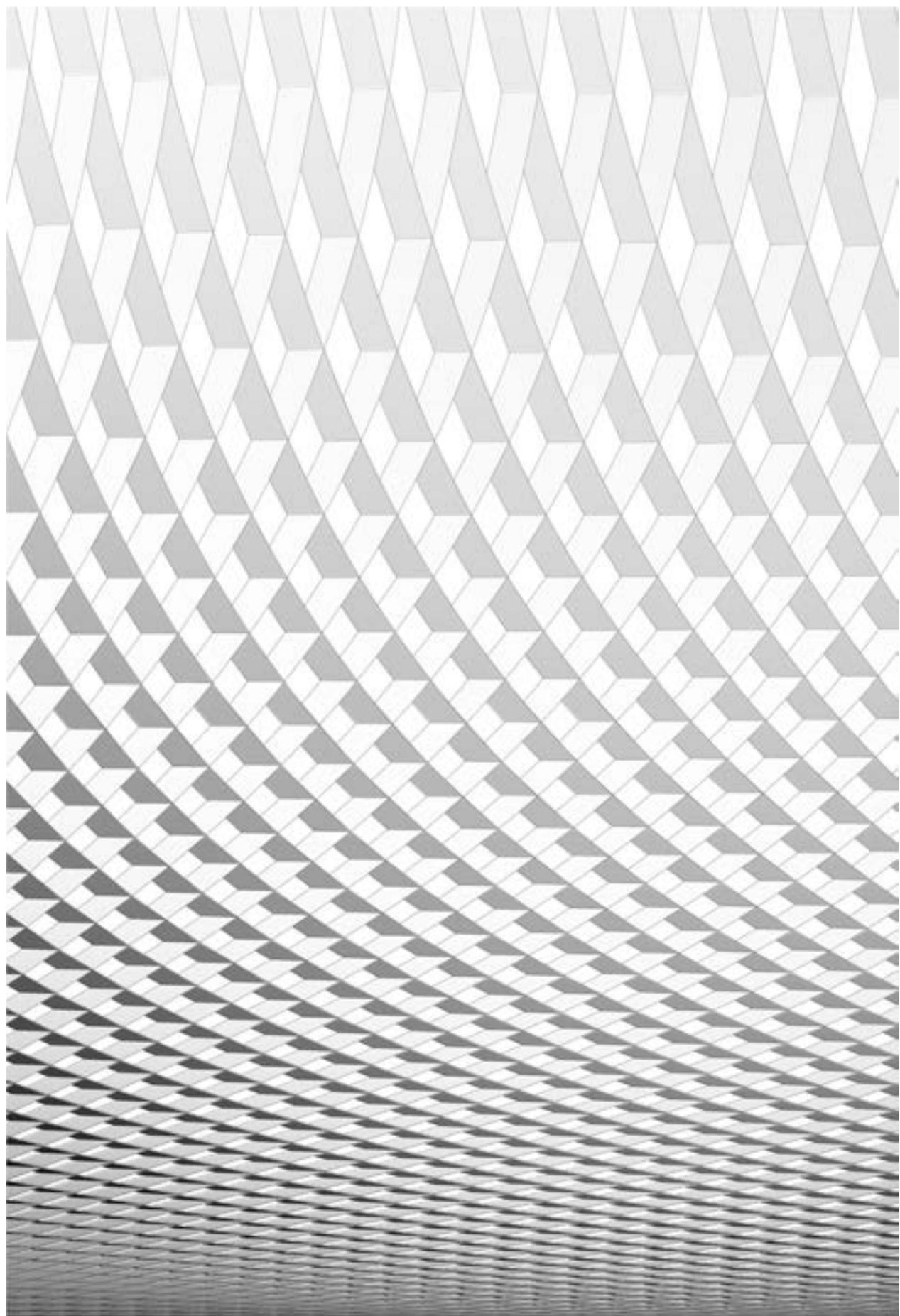
Wir haben nun unsere Linien zu Polylinien zusammengefuegt und als NURBS Kurven neu aufgebaut, indem wir ihre Kontrollpunkte interpoliert haben. Wie Du vielleicht gemerkt hast, sind die kuerzeren Kurven im Rhino Ansichtsfenster immer noch gerade Linien. Dies liegt darin begruendet, dass Du eine NURBS Kurve dritten Grades nicht mit weniger als vier Kontrollpunkten erstellen kannst. Lass uns den Datenbaum anpassen, um die Listen der Kontrollpunkte mit weniger als vier Elementen zu entfernen.

57.	<b>Sets/Tree/Prune Tree</b> – Ziehe eine <b>Prune Tree</b> Komponente auf die Leinwand	
58.	<b>Params/Input/Panel</b> – Ziehe ein Paneel auf die Leinwand	
59.	Verbinde den Punkte (P) Ausgabeparameter der <b>Control Points</b> Komponente mit dem Baum (T) Eingabeparameter der <b>Prune Tree</b> Komponente Wenn Du einen "Param Viewer" mit dem Punkte (P) Ausgabeparameter der "Control Points" Komponente verbindest und einen anderen mit dem Baum (T) Ausgabeparameter der "Prune Tree" Komponente, kannst Du sehen, dass die Anzahl der Aeste reduziert wurde.	
60.	Doppelklicke auf das <b>Panel</b> und gebe 4 ein.	
61.	Verbinde den Ausgabeparameter des <b>Panel</b> mit dem Minimum (N0) Eingabeparameter der <b>Prune Tree</b> Komponente	
62.	Verbinde den Baum (T) Ausgabeparameter der <b>Prune Tree</b> Komponente mit dem	

62.	Eckpunkte (V) Eingabeparameter der <b>Interpolate</b> Komponente	
63.	<b>Surface/Freeform/Extrude</b> – Ziehe eine <b>Extrude</b> Komponente auf die Leinwand	
	<b>Vector/Vector/Unit Y</b> – Ziehe eine <b>Unit Y</b> Komponente auf die Leinwand	
64.	Du benoegst vielleicht einen Y Einheitsvektor, je nachdem, in welcher Orientierung Du die Geometrie in Rhino referenziert hast	
65.	<b>Params/Input/Number Slider</b> – Ziehe einen <b>Number Slider</b> auf die Leinwand	
66.	Doppelklicke den <b>Number Slider</b> und setze die folgenden Werte: Rounding: Integer Lower Limit: 1 Upper Limit: 5 Value: 3	
67.	Verbinde den Kurve (C) Ausgabeparameter der <b>Interpolate</b> Komponente mit dem Basis(B) Eingabeparameter der <b>Extrude</b> Komponente	
68.	Verbinde den <b>Number Slider</b> Ausgabeparameter mit dem Faktor (F) Eingabeparameter der <b>Unit Y</b> Komponente	
69.	Verbinde den Einheitsvektor (V) Ausgabeparameter der <b>Unit Y</b> Komponente mit dem Richtung(D) Eingabeparameter der <b>Extrude</b> Komponente	



Du solltest nun ein Diagonalraster von Streifen oder Finnen im Rhinoansichtsfenster sehen. Passe den Faktor Schieberegler an, um die Tiefe der Finnen zu veraendern

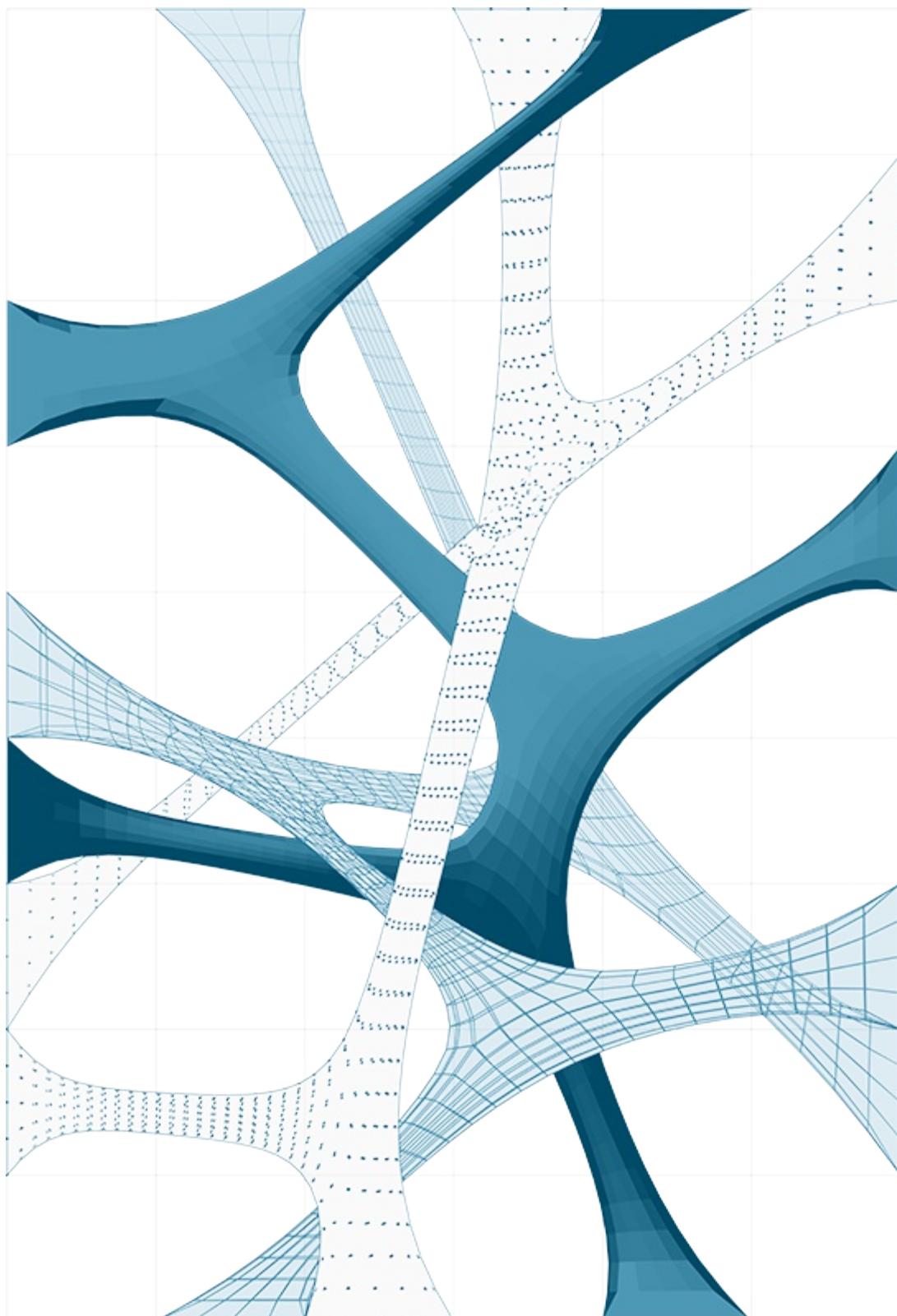


4 / 4

## 1.6. Mit Meshes arbeiten

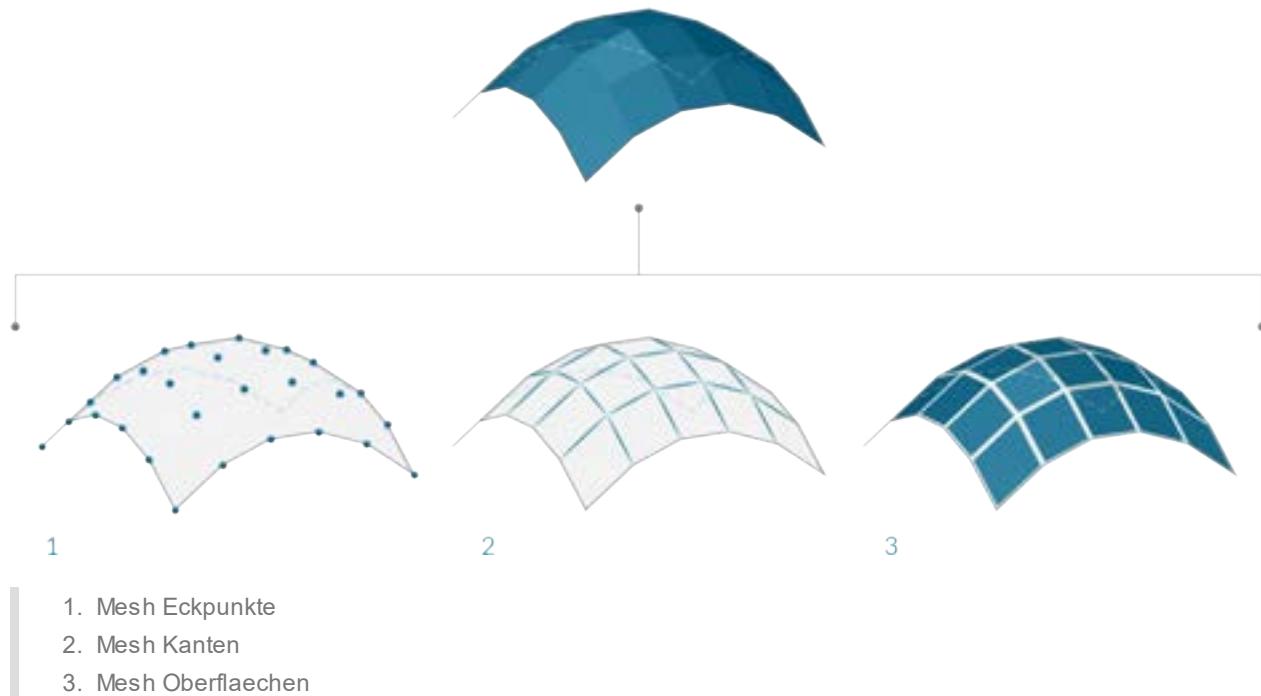
---

**Im Feld der computerbasierten Modellierung, sind meshes die häufigste Form der Darstellung von 3D Geometrien. Meshgeometrien sind eine leichtgewichtige und flexible alternative zur Arbeit mit NURBS und werden überall, vom Rendern und Visualisieren bis zur digitalen Fabrikation und 3D Druck verwendet. Dieses Kapitel wird eine Einführung in die Handhabung von Meshgeometrien in Grasshopper geben.**



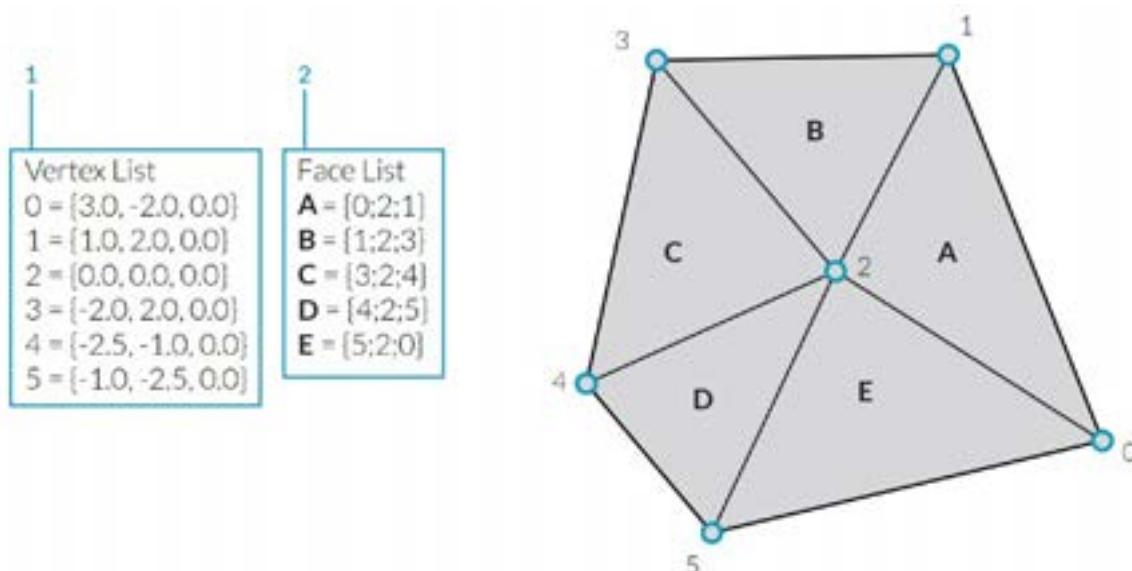
### ### 1.6.1 Was ist ein Polygonnetz?

**Ein Polygonnetz ist eine Sammlung von Vierecken und Dreiecken, die eine Fläche oder einen Körper darstellen. Dieser Abschnitt beschreibt die Struktur von Polygonnetzobjekten, welche aus Eckpunkten, Kanten und Netzflächen, sowie zusätzlichen Polygonnetzeigenschaften, wie Farben und Normalen, bestehen.**



#### 1.6.1.1 Grundsaetzliche Anatomie von Meshes

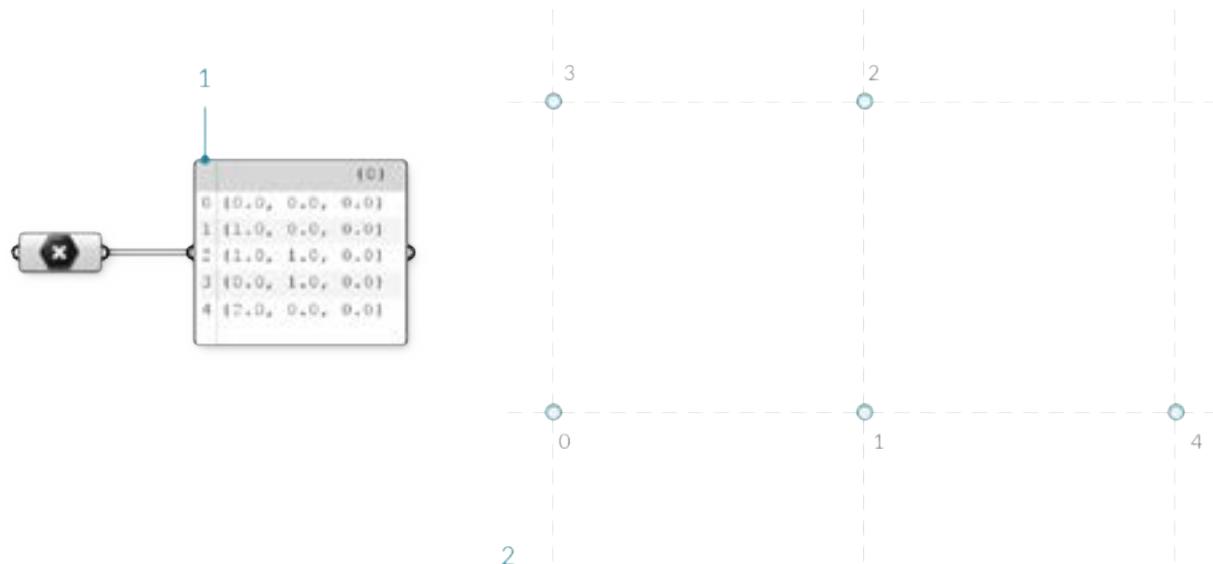
Grasshopper definiert Meshes mit einer Oberflächen-Eckpunkte Datenstruktur. Auf der einfachsten Ebene, ist diese Struktur eine Ansammlung von Punkten, die in Polygone gruppiert sind. Die Punkte des Mesh nennen wir *vertices*, während die Polygone *faces* genannt werden. Um ein Mesh zu erstellen, benötigen wir eine Liste von Eckpunkten und ein System um diese in "Faces" zu gruppieren.



1. Eine Liste von "Vertices"
2. "Faces" als Gruppierung von "Vertices"

#### Vertices

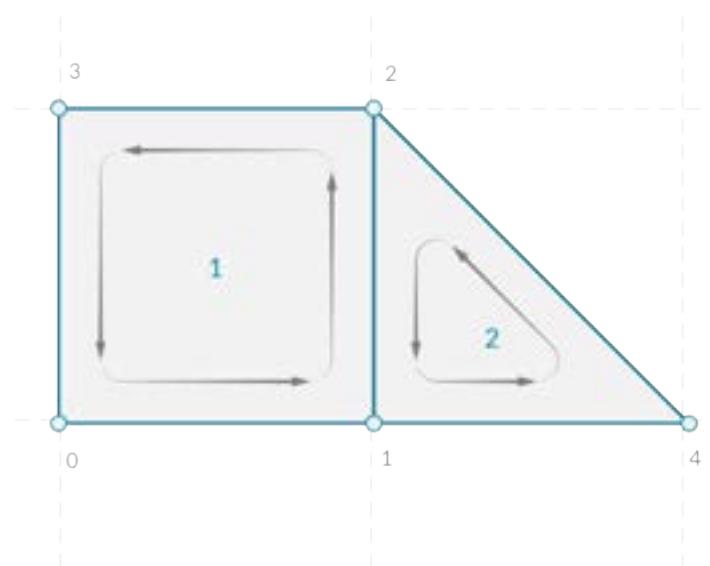
Die Vertices eines Mesh sind einfach eine Liste von Punkten. Erinnere Dich daran, dass eine Liste in Grasshopper eine Sammlung von Objekten darstellt. Jedes Objekt in der Liste hat einen *index*, der die Position von Objekten in einer Liste beschreibt. Der Index von Vertices ist sehr wichtig wenn ein Mesh konstruiert wird, oder wenn Informationen ueber die Struktur des Mesh benoetigt werden.



1. Eine Liste von Punkten. Alle Listen in Grasshopper beginnen mit einem Index von 0
2. Die Menge an Punkten mit ihrem entsprechenden Index benannt

## Faces

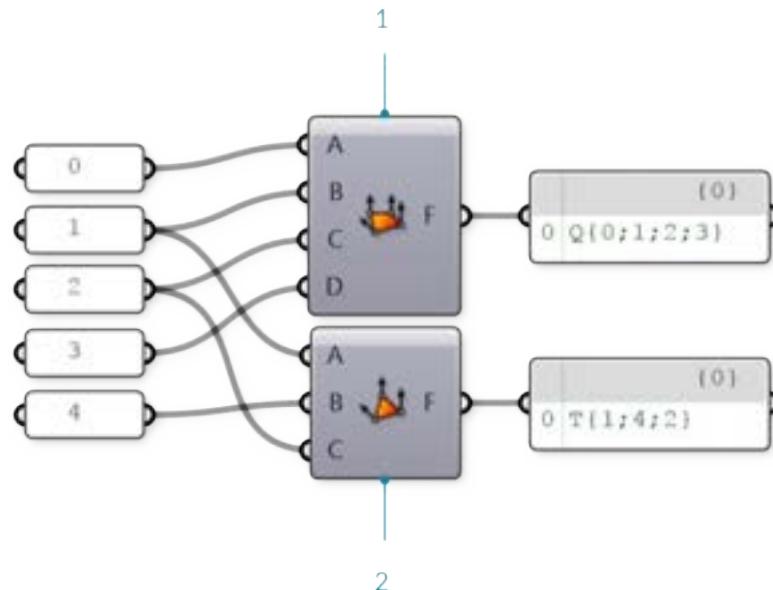
Ein Face ist eine geordnete Liste von drei oder vier Vertices. Die Flaechendarstellung eines Meshface ist deshalb durch die Position der Vertices beschrieben. Wir haben bereits eine Liste von Vertices, die ein Mesh bestimmen, also werden wir einfach die Indices des Vertices heranziehen, anstatt einzelne Punkte zur Beschreibung der Faces bereitzustellen. Dies ermoeglicht es uns die selben Vertices in mehreren Faces zu nutzen.



1. Ein viereckiges Face bestehend aus Indices 0, 1, 2, und 3
2. Ein dreieckiges Face bestehend aus Indices 1, 4, und 2

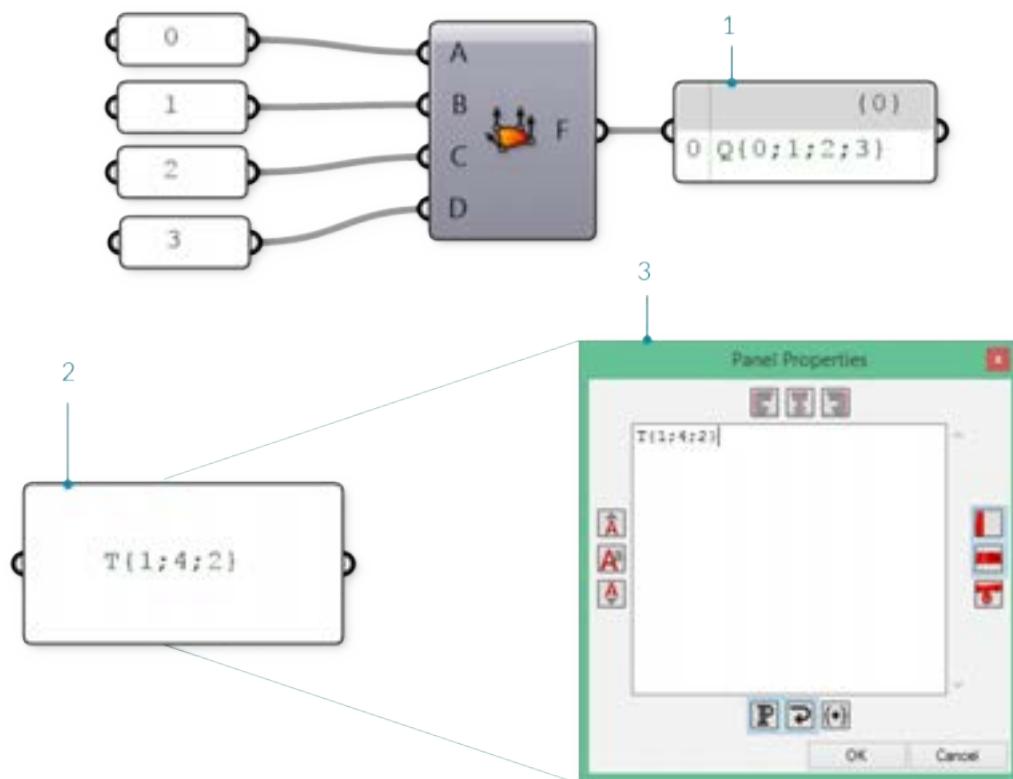
In Grasshopper, werden Faces entweder mit **Mesh Triangle** oder mit **Mesh Quad** Komponenten erzeugt. Der Eingabeparameter fuer diese Komponenten besteht aus Integerlisten, die den Indices der Vertices entsprechen, die wir fuer unser Face nutzen wollen. Indem wir ein **Panel** mit dem Ausgabeparameter dieser Komponente

verbinden, koennen wir sehen, dass dreieckige Faces als  $T\{A;B;C\}$  beschrieben werden, und viereckige Faces als  $Q\{A;B;C;D\}$ . Faces mit mehr als vier Seiten sind nicht zulaessig. Um ein fuenfseitiges Meshelement herzustellen, muss das Mesh in zwei oder mehr Faces zerlegt werden.



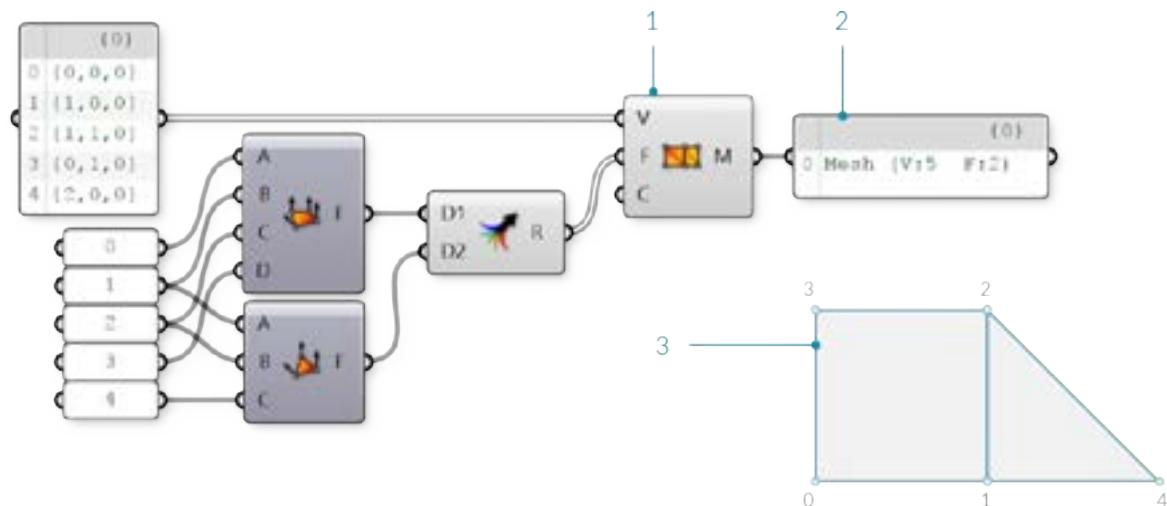
1. Mesh Quad Komponente mit Indices 0, 1, 2, und 3
2. Mesh Triangle Komponente mit Indices 1, 4, und 2

Es ist wichtig sich daran zu erinnern, dass diese Komponenten nicht zur Erzeugung einer Meshgeometrie fuehren, sondern der Ausgabeparameter eine Liste von Indices liefert, die definieren wie ein Mesh erzeugt werden soll. Indem wir dem Format dieser Liste Beachtung schenken, koennen wir auch manuell Faces editieren, indem wir eine **Panel** Komponente nutzen, um ein korrektes Format fuer entweder ein dreieckiges oder ein viereckiges Face anzugeben.



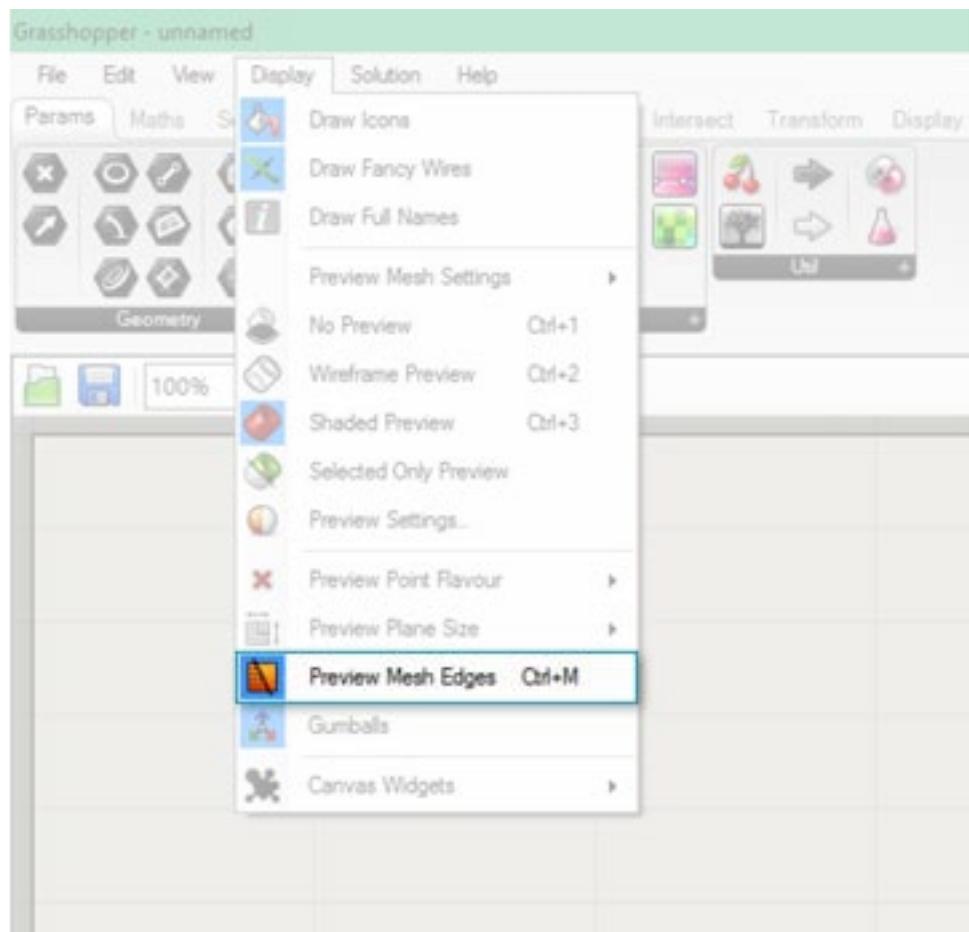
1. Ein Face, das mit einer **Mesh Quad** Komponente erzeugt wurde
2. Ein Face, das mit einem **Panel** erzeugt wird
3. Ein Panel Eigenschaftenfenster wird automatisch geöffnet, wenn ein Panel doppelgeklickt wird oder nach einem Rechtsklick "Edit Notes..." gewählt wird

So weit haben wir eine Liste mit Vertices und ein Set Facedefinitionen, aber noch kein Mesh erzeugt. Damit ein Mesh erzeugt wird, müssen wir die Faces und Vertices miteinander durch eine **Construct Mesh** Komponente verbinden. Wir verbinden unsere Liste mit Vertices mit dem V Eingabeparameter, und eine verschmolzene Liste der Faces mit dem F Eingabeparameter (Die Komponente hat auch einen optionalen Farbeingabeparameter, den wir unten noch behandeln werden.) Wenn wir ein Panel mit dem Ausgabeparameter der **Construct Mesh** Komponente verbinden, werden wir die Informationen zu der Anzahl der Faces und Vertices erhalten.

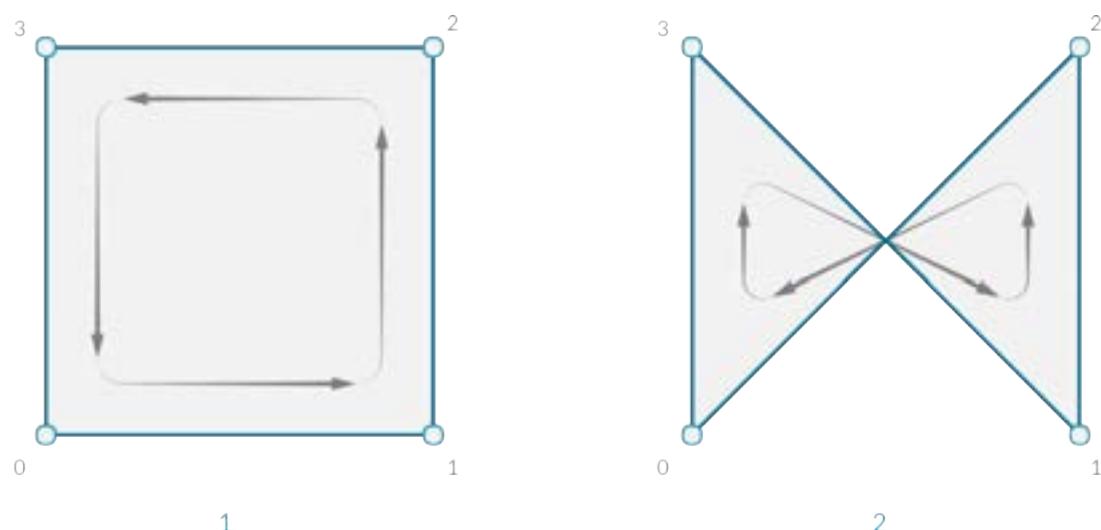


1. Die **Construct Mesh** Komponente nimmt eine Liste von Vertices und Faces als Eingabeparameter. Der Farbeingabeparameter ist optional und bleibt im Moment ungenutzt.
2. Ein Panel zeigt, dass wir ein Mesh mit 5 Vertices und 2 Faces erzeugt haben
3. Das resultierende Mesh (die Vertices wurden mit ihren Indizes benannt)

Als Standard zeigt Grasshopper die Kanten von Meshgeometrien in der Vorschau nicht an. Um die Kanten in der Vorschau zu sehen, kannst Du die Meshkantenvorschau mit dem Tastaturkürzel Strg+M, oder indem Du im Ansichtsmenu die Option 'Preview Mesh Edges' auswählst.



Es ist extrem wichtig der Ordnung der Indices Aufmerksamkeit zu schenken, wenn Du ein Meshface erstellst. Das Face wird hergestellt, indem die Vertices in der angegebenen Abfolge verbunden werden. Bei viereckigen Faces sind  $Q\{0,1,2,3\}$  und  $Q\{1,0,2,3\}$  sehr verschieden, auch wenn sie beide aus den selben Vertices bestehen. Falsche Vertexreihenfolge kann zu Problemen, wie Löchern im Mesh oder nicht-mannigfaltigen Meshgeometrien oder nicht-orientierbaren Flächen führen. Solche Meshgeometrien werden üblicherweise nicht korrekt gereebert und sind nicht möglich 3D gedruckt zu werden. Diese Schwierigkeiten werden im Abschnitt **Understanding Topology** tiefer behandelt.



1. Ein viereckiges Face mit den Indices 0,1,2,3
2. Ein viereckiges Face mit den Indices 0,3,1,2

### 1.6.1.2 Implizite Mesh Daten

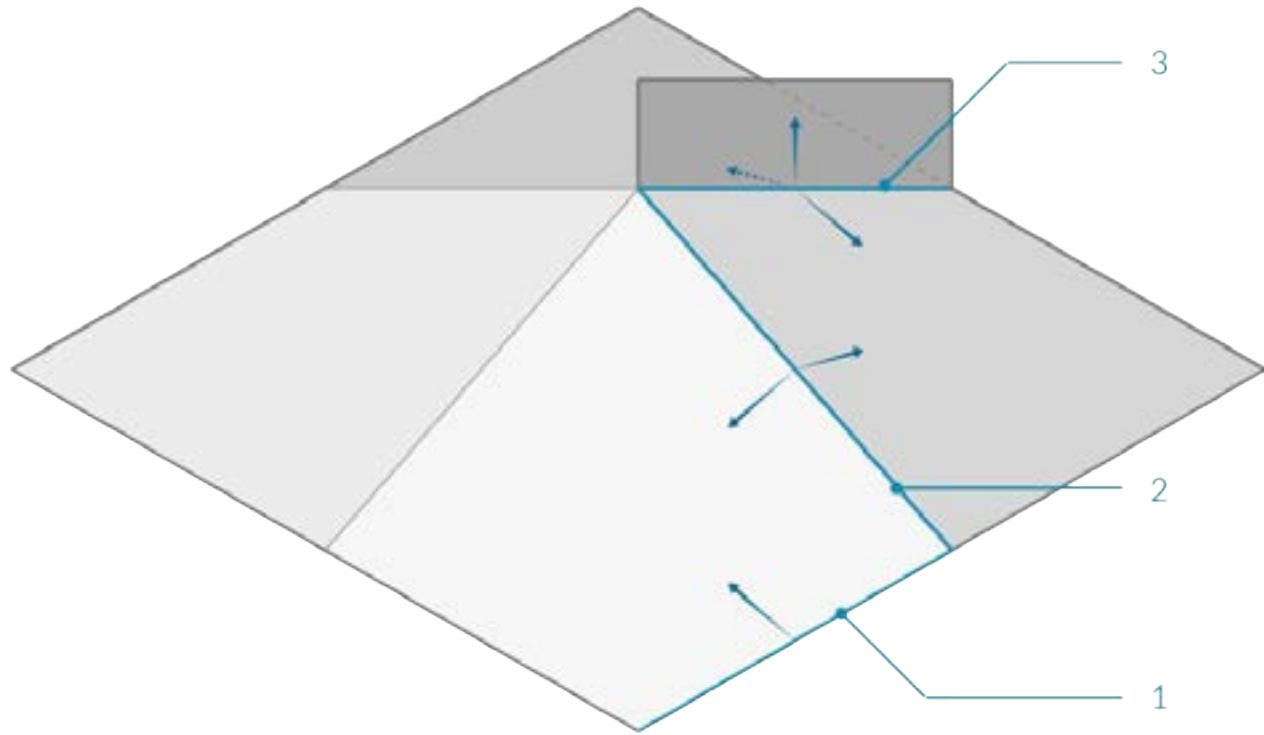
Zusaetlich zu Faces und Vertices, gibt es andere Informationen ueber Meshed, die wir benutzen werden. In Face-Vertex-basierten Meshes, werden Daten wie *edges* und *normals* basierend auf Faces und Vertices implizit kalkuliert. Dieser Abschnitt beschreibt, wie wir diese Informationen erheben.

#### Edges

Die *edges* eines Meshs sind Linien, die zwei aufeinanderfolgende Vertices eines Face miteinander verbinden. Merke, dass einige Kanten zwischen mehreren Oberflaechen geteilt werden, waehrend andere Kanten nur an eine Oberflaeche angrenzen. Die Anzahl an Oberflaechen, die an eine Kante angrenzen, wird *Valenz* der Kante genannt.

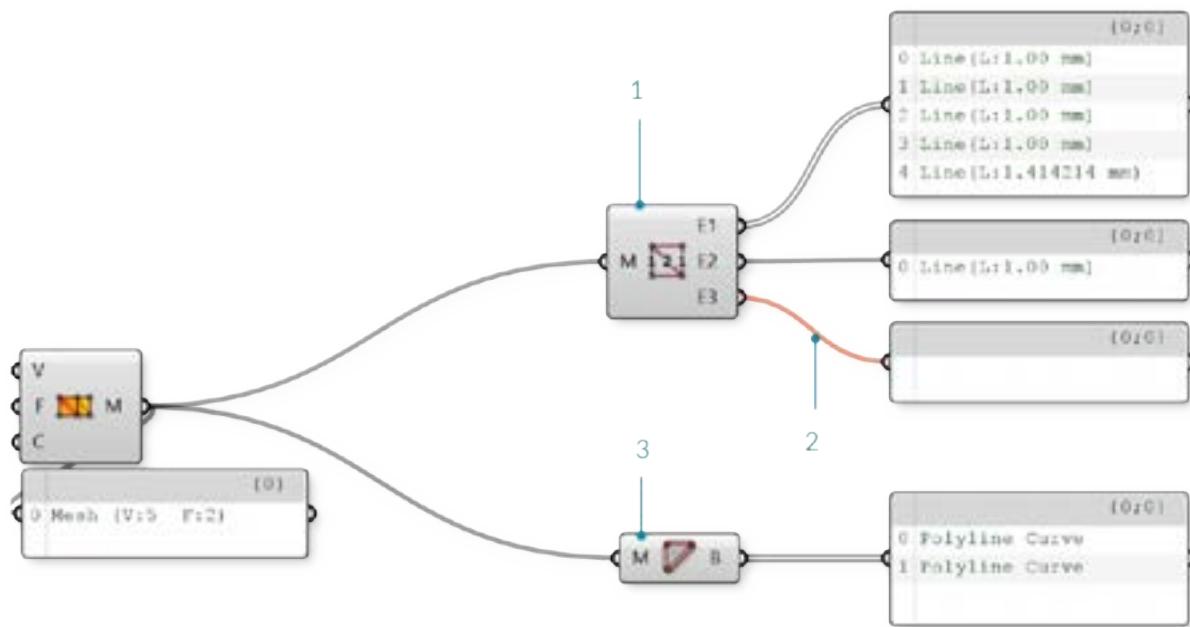
Grasshopper gruppiert Kanten basierend auf ihrer Valenz in drei Kategorien:

1. E1 - 'Naked Edges' haben eine Valenz von 1. Sie bestimmen die auesseren Grenzen des Meshs.
2. E2 - 'Interior Edges' haben eine Valenz von 2.
3. E3 - 'Non-Manifold Edges' haben eine Valenz von 3 oder groesser. Meshs die solche Strukturen enthalten werden "Non-Manifold" genannt und im naechsten Abschnitt besprochen.



- 1. "Naked edge" mit einer Valenz von 1
- 2. "Interior edge" mit einer Valenz von 2
- 3. "Non-manifold edge" mit einer Valenz von 3

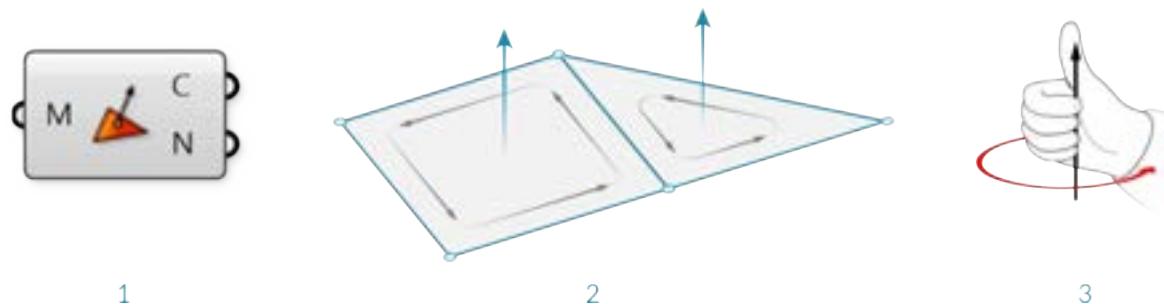
Wir koennen die **Mesh Edges** Komponente nutzen um die Kanten eines Mesh entsprechend ihrer Valenz auszugeben. Dies ermoeglicht es Kanten entlang der Meshgrenze oder nicht-mannigfaltige Kanten zu identifizieren. Manchmal jedoch ist es nuetzlicher die gesamte Grenze jeder Oberflaeche zu erhalten. Dazu nutzen wir die **Face Boundaries** Komponente. Diese wir die Grenze einer jeden Oberflaeche als Polylinie ausgeben.



1. Die **Mesh Edges** Komponente gibt drei Sets von Kanten aus. Dieses Mesh hat 5 "naked edges", 1 "interior edge" und keine "non-manifold edges"
2. Der E3 Ausgabeparameter ist leer, da dieses Mesh keine nicht-mannigfaltigen Kanten hat.
3. Die **Face Boundaries** Komponente gibt eine Polylinie fuer jede Oberflaeche aus

### Face Normals

Ein *Normalenvektor* ist ein Vektor mit einer Magnitude von eins und ist rechtwinklig zu einer Flaeche angeordnet. Im Fall einer dreieckigen Oberflaeche, wissen wir, dass alle drei Punkte auf einer ebenen Flaeche liegen, so dass die Normale immer rechtwinklig zu dieser Ebene liegen. Woher wissen wir nun, in welche Richtung der Normalenvektor deutet ("hoch" oder "runter")? Wieder einmal ist die Reihenfolge der Indices ausschlaggebend. Meshoberflaechen werden in Grasshopper gegen den Uhrzeigersinn definiert, so dass eine Oberflaeche mit Vertices {0,1,2} sich umgekehrt zu einer mit Vertices {1,0,2} verhaelt. Ein anderer Weg um die Richtung zu visualisieren ist die *Rechtehandregel*.



1. Die **Face Normals** Komponente wird eine Liste von Zentren und Normalenvektoren fuer jede Oberflaeche ausgeben
2. Oberflaechennormalen entsprechend zur Abfolge von Vertices
3. "Rechtehandregel" zur Bestimmung der Normalenrichtung

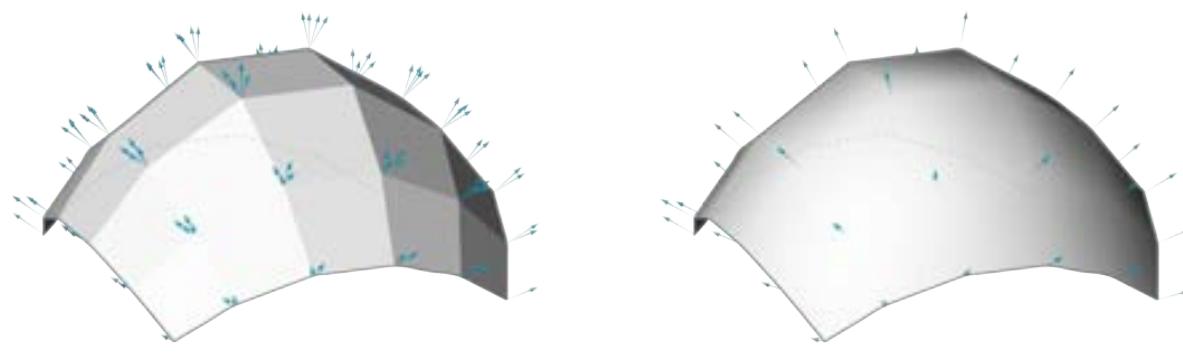
Grasshopper erlaubt auch viereckige Oberflaechen in Meshs, in welchem Fall die vier Punkte nicht auf einer planaren Flaeche liegen. Fuer diese Oberflaechen, wird des Zentrum aus dem Mittelwert der Koordinaten der vier Vertices gebildet (im Fall einer nicht-ebenen viereckigen Oberflaeche liegt dieser Punkt nicht notwendigerweise auf dem Mesh). Um die Normale einer vierseitigen Oberflaeche zu berechnen, muessen wir diese zuerst triangulieren, indem wir sie in zwei Dreiecke teilen. Die Normale der viereckigen Gesamtoberflaeche ist dann der Mittelpunkt der

beiden Normalenvektoren, gewichtet nach der Flaeche der beiden Dreiecke.

### Eckpunktnormalen

Zusaetlich zu den Oberflaechennormalen, ist es auch eine Moeglichkeit die Normalen fuer jeden Eckpunkt des Polygonnetzes zu berechnen. Fuer einen Eckpunkt, der nur in einer einzigen Oberflaechen genutzt wird, ist die Normale klar definiert. Wenn der Eckpunkt an mehrere Oberflaechen grenzt, ist die Normale an diesem Eckpunkt der Mittelwert der beiden Oberflaechennormalen.

Waehrend die Eckpunktnormalen weniger intuitiv berechnet werden, sind sie doch wichtig fuer die geglaettete Darstellung von Polygonnetzen. Du hast vielleicht sogar festgestellt, dass auch wenn ein Polygonnetz aus ebenen Oberflaechen aufgebaut istm kann es trotzdem glatt und gerundet wirken, wenn es in Rhino schattiert wurde. Diese geglaettete Darstellung wird durch die Eckpunktnormalen ermoeglicht.



1

2

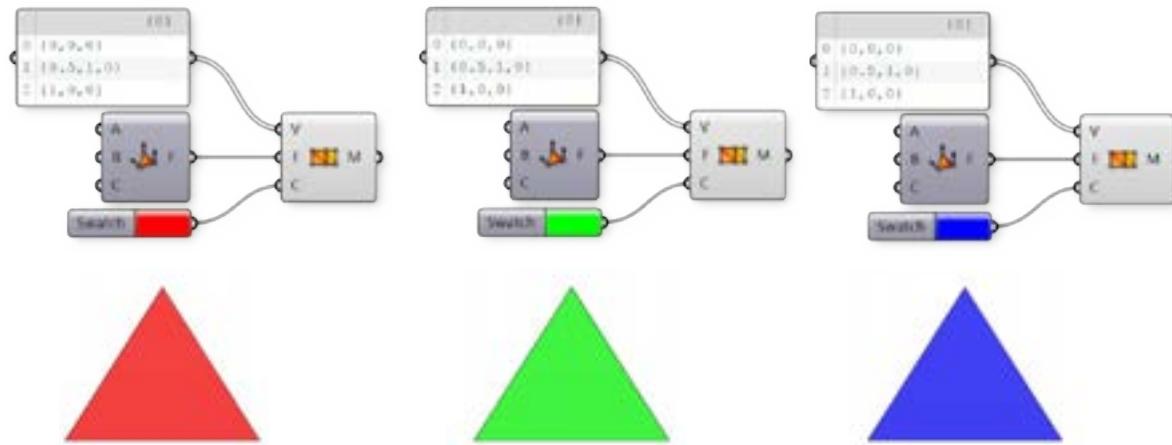
1. Normalen entsprechend der Oberflaechennormalen ergeben eine diskrete, polygonale Schattierung
2. Angrenzende Oberflaechennormales werden gegeneinander ausgemittelt um Eckpunktnormalen zu erzeugen, was in einer geglaetteten Schattierung entlang der Oberflaechen fuehrt

### 1.6.1.3 Polygonnetzeigenschaften

Polygonnetze koennen andere Eigenschaften mit den Eckpunkten oder Oberflaechen verknuepft haben. Die einfachste dieser Eigenschaften ist die Eckpunktfarben welche unterhalb beschrieben wird, aber auch andere Eigenschaften wie UV Koordinaten von Texturen existieren. (Manche Programme erlauben auch Eckpunktnormalen als Eigenschaften, anstatt wie in unserem Fall abgeleitet von Oberflaechen oder Eckpunkten, welche dann noch groessere Flexibilitaet in der Erscheinung der gerenderten Flaeche erlauben).

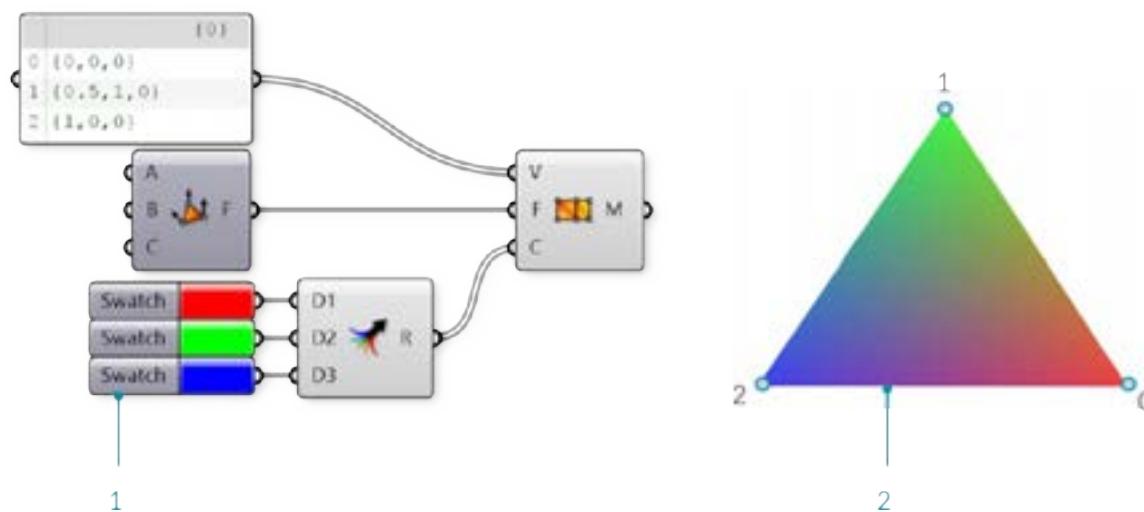
#### Farbe

Wenn Du eine **Construct Mesh** Komponente verwendest, gibt es einen optionalen Eingabeparameter fuer die Eckpunktfarbe. Farben koennen auch einem bestehenden Polygonnetz mit der **Mesh Color** Komponente zugewiesen werden. Ein einzelner Farbton kann so das gesamte Polygonnetz einfärben.



Dreieckige Polygonnetzobjekte mit rot, gruen und blau eingefärbt

Während die oben aufgeführten Beispiele das gesamte Polygonnetz einfarben, werden die Farbdaten eigentlich an den Eckpunkten zugewiesen. Indem wir eine Liste von drei Farben zuweisen, können wir jeden Eckpunkt einzeln einfarben. Diese Farben werden für Visualisierungen verwendet, wobei jede Oberfläche mit einem interpolierten Farbton der Eckpunkte gerendert wird. Zum Beispiel wird im unterhalb angeführten Bild eine dreieckige Oberfläche mit Eckpunktfarben rot, grün und blau gezeigt.

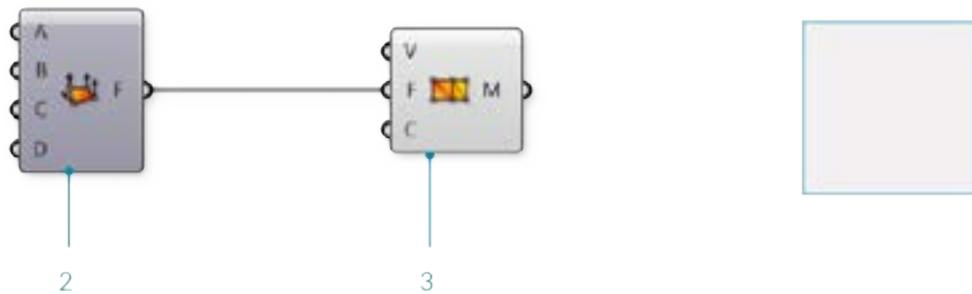


1. Rot, grün und blau werden an den drei Eckpunkten des Polygonnetzes zugewiesen
2. Das resultierende Polygonnetz interpoliert die Farben der Eckpunkte

#### 1.6.1.4 Exercise

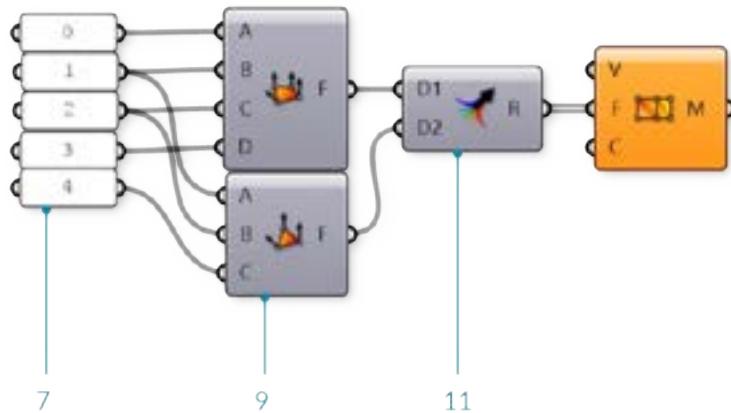
Beispieldateien zu diesem Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

01.	Beginne eine neue Definition, drücke Strg+N (in Grasshopper)	
02.	<b>Mesh/Primitive/Mesh Quad</b> - Ziehe eine <b>Mesh Quad</b> Komponente auf die Leinwand	
03.	<b>Mesh/Primitive/Construct Mesh</b> - Ziehe eine <b>Construct Mesh</b> Komponente auf die Leinwand	
04.	Verbinde einen Oberfläche (F) Ausgabeparameter der <b>Mesh Quad</b> Komponente mit dem Oberfläche (F) Eingabeparameter der <b>Construct Mesh</b> Komponente	



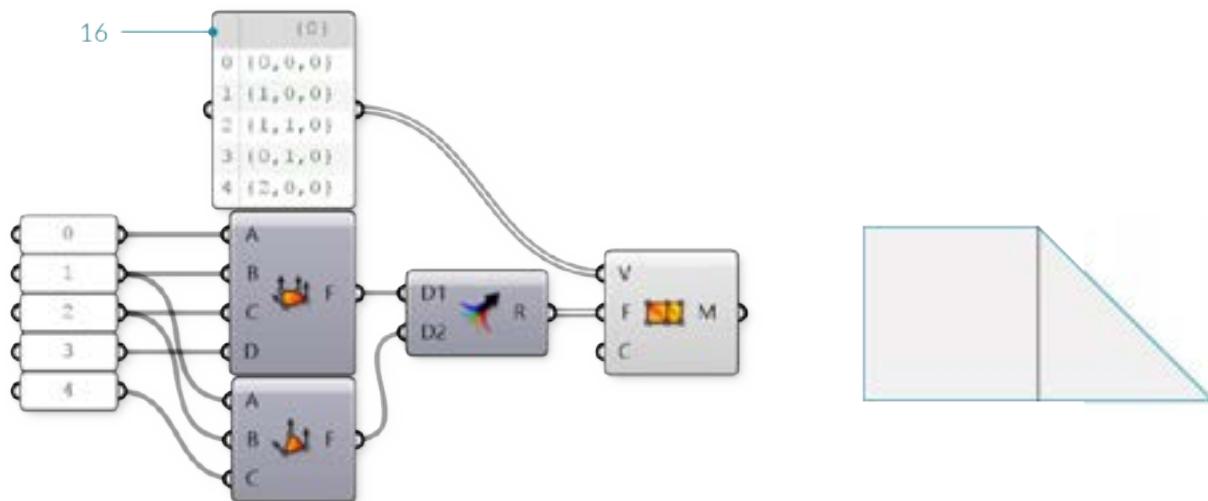
**Mesh Quad** und **Construct Mesh** haben Standardwerte, die einzelne Polygonnetzoberflächen erstellen. Als næchstes werden wir die Standardwerte mit unseren eigenen Eckpunkten und Oberflächen ersetzen.

05.	Params/Input/Panel - Ziehe eine <b>Panel</b> Komponente auf die Leinwand	
06.	Doppelklicke die <b>Panel</b> Komponente und setze den Wert auf '0'	
07.	<b>Params/Input/Panel</b> - Ziehe vier weitere <b>Panel</b> Komponenten auf die Leinwand und setze ihre Werte auf 1,2,3, und 4	
08.	Du kannst auch die urspruengliche **Panel** Komponente kopieren, indem Du sie anklickst und ziehst, waehrend Du die Altaste drueckst, bevor Du den Klick wieder loslaesst	
09.	Verbinde die <b>Panels</b> mit den Eingabeparametern der <b>Mesh Quad</b> Komponente in folgender Reihenfolge: 0 - A 1 - B 2 - C 3 - D	
10.	Verbinde die <b>Panels</b> mit den Eingabeparametern der <b>Mesh Triangle</b> Komponente in der folgenden Reihenfolge: 1 - A 2 - B 4 - C	
11.	<b>Sets/Tree/Merge</b> - Ziehe eine <b>Merge</b> Komponente auf die Leinwand	
12.	Verbinde den Oberflaechen (F) Ausgabeparameter der <b>Mesh Quad</b> Komponente mit dem Daten 1 (D1) Eingabeparameter der <b>Merge</b> Komponente und den Oberflaechen (F) Ausgabeparameter der <b>Mesh Triangle</b> Komponente mit dem Daten 2 (D2) Eingabeparameter der <b>Merge</b> Komponente	
13.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Merge</b> Komponente mit dem Oberflaechen (F) Eingabeparameter der <b>Construct Mesh</b> Komponente	



Die Standard Eckpunkte (V) Liste der **Construct Mesh** Komponente hat vier Punkte, aber die **Mesh Triangle** Komponente nutzt einen Index von 4, was einem fuenften Punkt in einer Liste entsprechen wuerde. Da hier nicht genug Eckpunkte vorliegen, gibt die **Construct Mesh** Komponente eine Fehlerwarnung aus. Um dies zu beheben, werden wir unsere eigene Liste von Punkten eingeben.

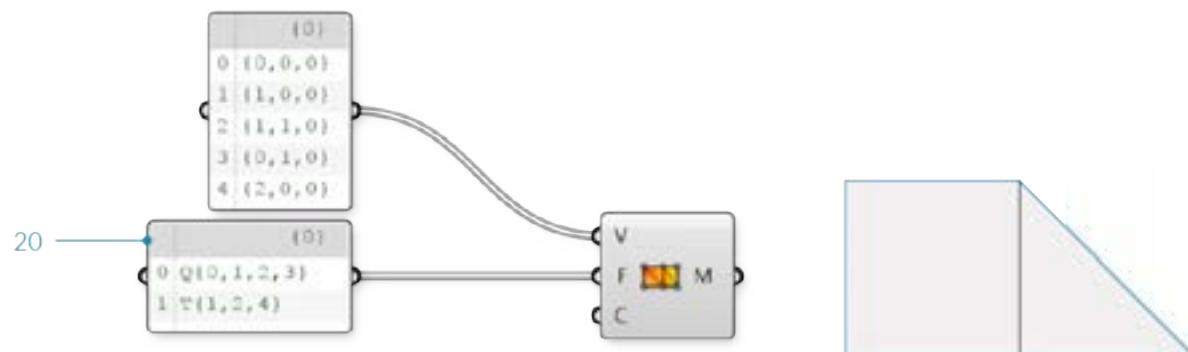
14.	<b>Params/Input/Panel</b> - Ziehe eine <b>Panel</b> Komponente auf die Leinwand	
	Rechtsklicke auf die <b>Panel</b> Komponente und entferne die Auswahl von 'Multiline Data' option	
15.	Als Standard ist die Option 'Multiline Data' im Paneel aktiviert. Indem wir sie deaktivieren, wird jede Zeile des Paneel als separates Element innerhalb der Liste gelesen.	
	Doppelklicke die <b>Panel</b> Komponente um sie zu bearbeiten und gebe folgende Punkte ein: {0,0,0} {1,0,0} {1,1,0} {0,1,0} {2,0,0}	
16.	Versichere Dich, dass Du die richtige Schreibweise verwendest. Um einen Punkt in einem **Panel**, zu beschreiben, musst Du die geschweiften Klammern verwenden: '{' und '}' mit Komma zwischen den x, y, und z Werten	
17.	Verbinde die <b>Panel</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Construct Mesh</b> Komponente	



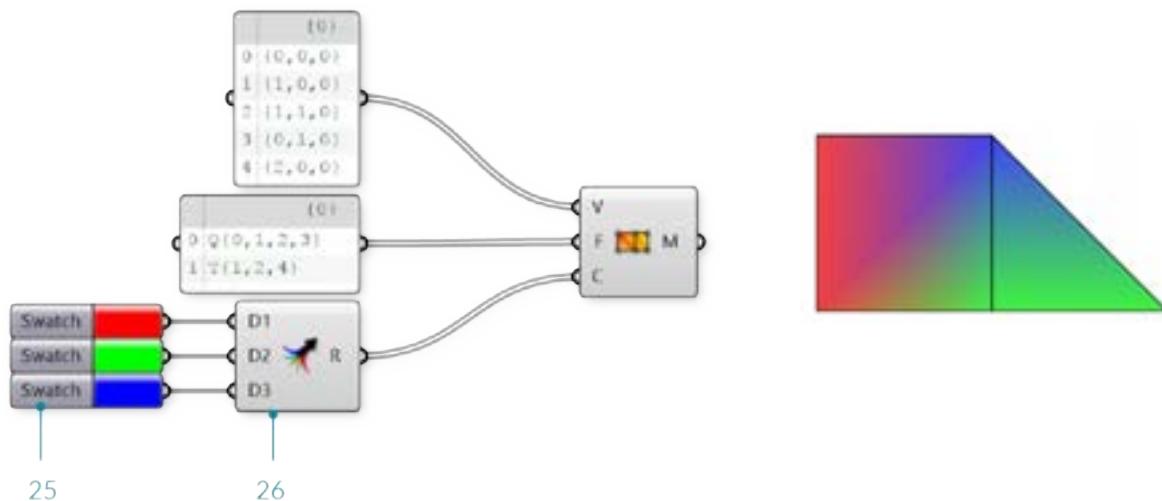
Wir haben nun ein Polygonnetz mit zwei Oberflächen und fuenf Eckpunkten.

Optional koennen wir die **Mesh Quad** und **Mesh Triangle** Komponenten mit einem Panel ersetzen, das die Indices der Oberflächen definiert.

18.	<b>Params/Input/Panel</b> - Ziehe eine <b>Panel</b> Komponente auf die Leinwand	
	Rechtsklicke auf die <b>Panel</b> Komponente und entferne die Auswahl von 'Multiline Data'	
19.	Alternativ kannst Du die bestehende <b>Panel</b> Komponente, die wir fuer die Punkte verwendet haben kopieren, da diese bereits 'Multiline Data' deaktiviert hat	
20.	Doppelklicke auf die <b>Panel</b> Komponente um sie zu bearbeiten und gebe folgende Daten ein: Q{0,1,2,3} T{1,2,4}	
21.	Verbinde die <b>Panel</b> Komponente mit dem Oberflächen (F) Eingabeparameter der <b>Construct Mesh</b> Komponente	



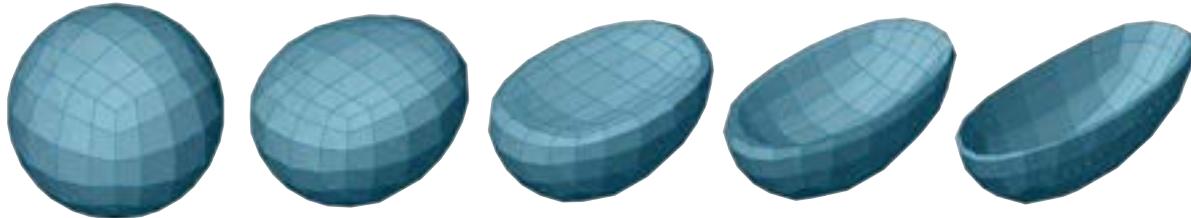
22.	<b>Params/Input/Colour Swatch</b> - Ziehe eine <b>Colour Swatch</b> Komponente auf die Leinwand 
23.	Klicke auf den farbigen Abschnitt der Komponente (der Standard ist weiss) um das Farbauswahlpaneel zu öffnen
24.	Nutze den Schieberegler um die Werte G und B auf null zu stellen. Die angezeigte Farbe sollte nun rot sein
25.	<b>Params/Input/Colour Swatch</b> - Ziehe zwei weitere <b>Colour Swatch</b> Komponenten auf die Leinwand und setze ihre Farben auf blau und grün
26.	<b>Sets/Tree/Merge</b> - Ziehe eine <b>Merge</b> Komponente auf die Leinwand
27.	Verbinde die drei <b>Color Swatch</b> Komponenten mit den D1, D2, und D3 Eingabeparametern der <b>Merge</b> Komponente.
28.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Merge</b> Komponente mit dem Farbe (C) Eingabeparameter der <b>Construct Mesh</b> Komponente



Wir haben fünf Eckpunkte, aber nur drei Farben. Grasshopper wird die Farben in einem sich wiederholenden Muster zuweisen. In diesem Fall werden die Eckpunkte 0 und 3 rot, die Eckpunkte 1 und 4 grün und der Eckpunkt 2 blau dargestellt.

### ### 1.6.2 Topology verstehen

**Während die Eckpunkte eines Polygonnetzes Information über Positionen erhalten sind es wirklich die Verbindungen zwischen den Eckpunkten, die eine Polygonnetzstruktur ihre einzigartige Struktur und Flexibilität geben.**

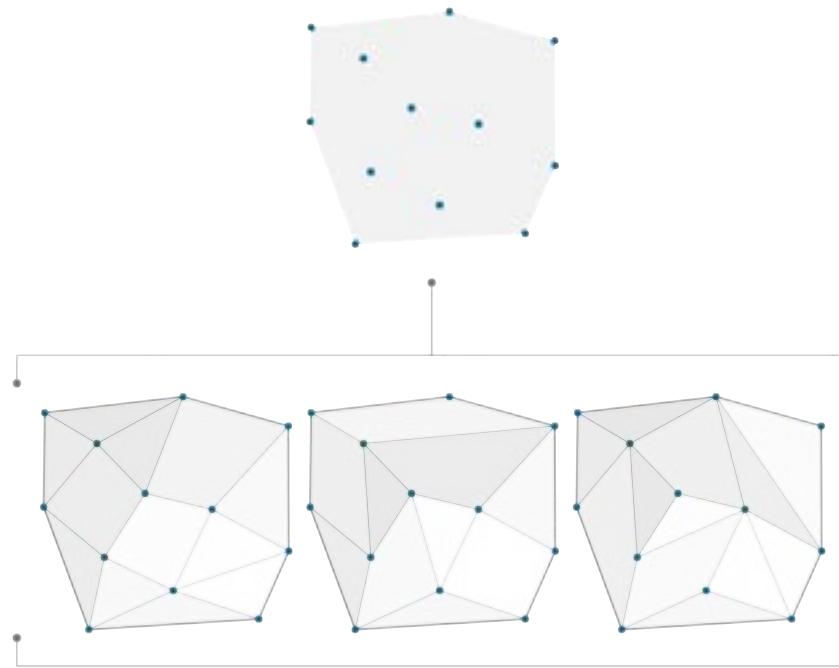


#### 1.6.2.1 Was ist Topology?

Eine Einführung zur Polygonnetzgeometrie wäre unvollständig ohne zumindest eine grundlegende Einführung zur Topologie. Weil Topologie auch mit den Verhältnissen zwischen einer Menge von "Dingen" und deren Eigenschaften eher als mit den "Dingen" selbst beschäftigt, mobilisiert es eine riesige Bandbreite von greifbaren und ungreifbaren Anwendungen. In diesem Primer, sind wir an der grundsätzlichen Anwendung in einem parametrischen Arbeitsablauf interessiert, der uns die Möglichkeit zur Erstellung und Kontrolle von Polygonnetzgeometrien gibt.

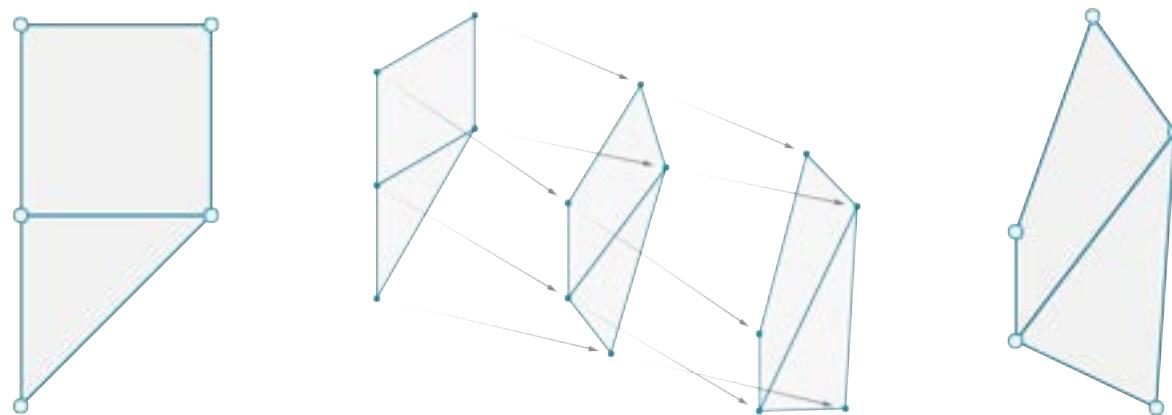
In Grasshopper gibt es zwei grundlegende Typen von Informationen, die ein Polygonnetz definieren und zwar Geometrie und Konnektivität; in anderen Worten, eine Menge an Punkten im Raum (Eckpunkte) und eine Menge von entsprechenden Punktzuordnungen (Oberflächen).

Ohne Verbindungsinformation, ist ein Polygonnetz unstrukturiert und daher immer noch undefiniert. Die Einführung einer Anzahl von Oberflächen ist der Schritt (oder Sprung) der letztendlich ein Polygonnetz erzeugt und seinen Charakter bezüglich Kontinuität, Konvergenz und Verbindungen generiert; dieses strukturelle Netzwerk wird als *topologischer Raum* bezeichnet.



Die selbe Menge an Eckpunkten kann unterschiedliche Verbindungsinformationen haben und ergibt unterschiedliche Topologien.

### Homeomorphismus

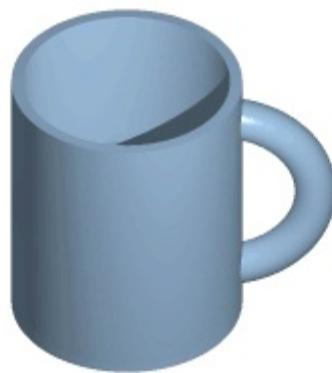


Die Punkte eines Polygonnetzes können bewegt werden ohne die Konnektivitätsinformation zu verändern. Das neue Polygonnetz hat die selbe Topologie wie das ursprüngliche.

Es ist möglich für zwei bestimmte Polygonnetzformen topologisch identisch zu sein. All dies würde bedeuten, dass sie aus der selben Anzahl von Punkten bestehen und dass die Punkte durch den selben Satz Netzelemente strukturiert sind. Früher haben wir festgelegt, dass Netzelemente nur auf Indizes eines Satzes von Punkten beruhen und sich nicht auf die eigentliche Position im Raum beziehen. Deshalb ergibt sich, wenn der einzige Unterschied zwischen zwei Polygonnetzen eine bestimmte dreidimensionale Position von Punkten ist, die durch den Nutzer definiert werden, dann sind die beiden Polygonnetze als "homöomorph" zu betrachten (topologisch äquivalent) und teilen deshalb alle topologischen Eigenschaften.

$\{A, R\}$   $\{B\}$   $\{C, G, I, J, L, M, N, S, U, V, W, Z\}$   
 $\{D, O\}$   $\{E, F, T, Y\}$   $\{H, K\}$   $\{P, Q\}$   $\{X\}$

Ein Beispiel von Homoeomorphismus sind Buchstaben (merke dass einige der homoeomorphen Gruppen oberhalb sich dadurch unterscheiden, welche Schriftart sie nutzen).

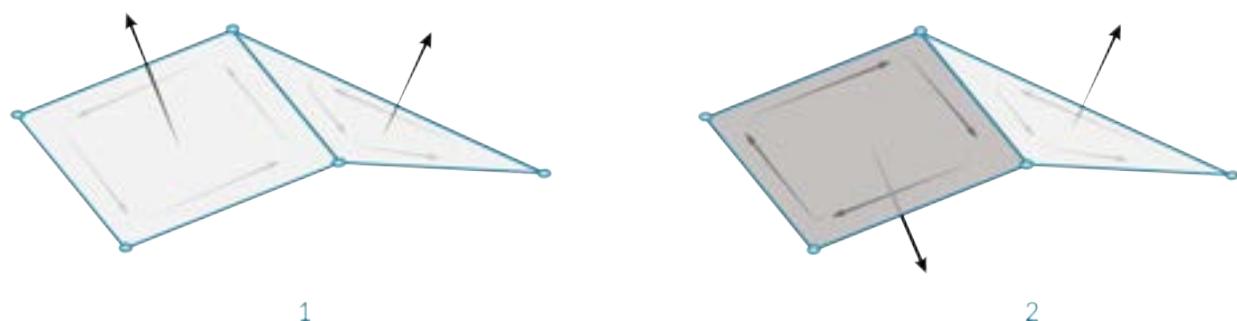


Topologisch äquivalente Schale und Donut

### 1.6.2.2 Polygonnetzcharakteristika

#### Orientierbar

Ein Polygonnetz wird als *orientierbar* betrachtet, wenn es wohldefinierte Seite aufweist. Ein einfaches Beispiel einer nicht orientierbaren Polygonnetze ergibt sich, wenn nebeneinander liegende Normale in entgegengesetzte Richtungen zeigen. Diese "umgekehrten Netzelementen" können Probleme in der Visualisierung und beim Rendern darstellen, sowie bei der Fertigung von 3D-Drucken bereiten.



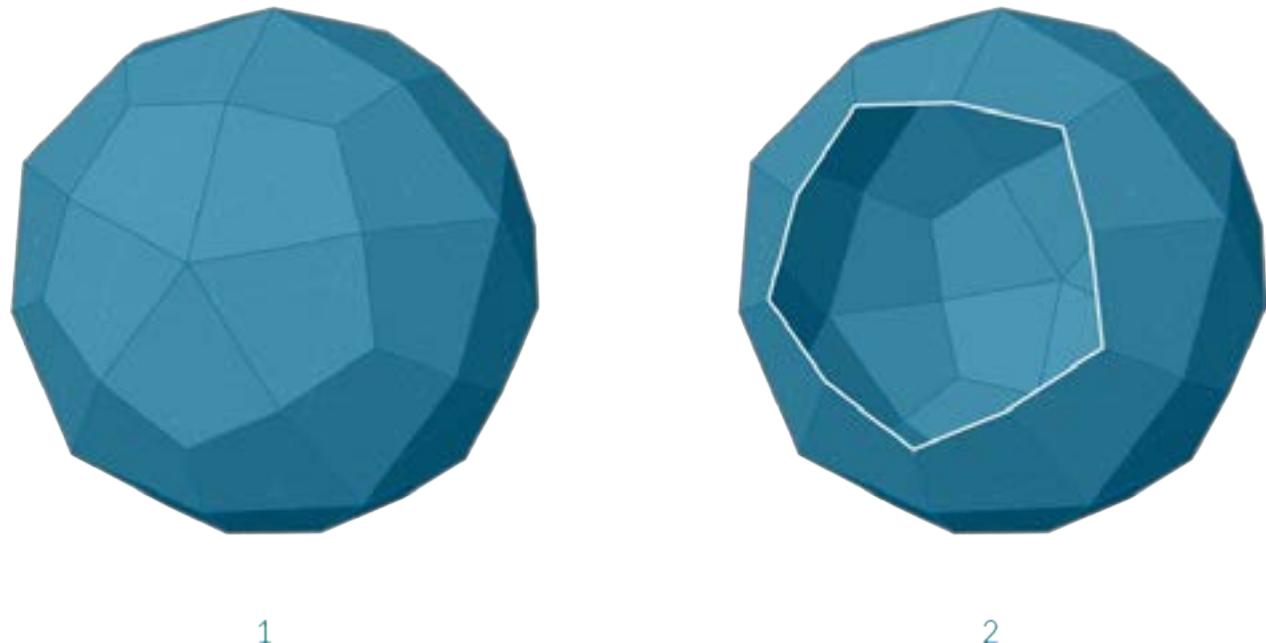
1. Eine orientierbare Fläche mit Netzelementen, die in die gleiche Richtung deuten
2. Eine nicht orientierbare Fläche hat angrenzende Normalen, die in verschiedene Richtungen deuten.

#### Open vs Closed

Es ist oftmals notwendig zu wissen, ob ein Polygonnetz ein *geschlossenes* Polygonnetz ist, das einen Körper beschreibt, oder ein *offenes* Polygonnetz, das lediglich eine 2D Fläche darstellt. Der Unterschied kann im Prinzip bezüglich der Herstellbarkeit sein. Du kannst keine einzelne Fläche ohne Dicke 3D drucken, sondern musst dem Polygonnetz zusätzlich eine Dicke geben, so dass es ein Körper wird. Polygonnetzkörper sind auch Voraussetzung für boolesche Operationen (die im nachfolgenden Abschnitt behandelt werden).

Die **Mesh Edges** Komponente kann genutzt werden um dies zu bestimmen. Wenn keine der Kanten des Polygonnetz eine Valenz von 1 hat (wenn der E1 Ausgabeparameter *null* ausgibt), dann wissen wir dass alle Kanten 'innenliegende Kanten' sind und das Polygonnetz keine auessere Begrenzungskante hat, also ein geschlossenes Polygonnetz ist.

Auf der anderen Seite, wenn es 'offene Kanten' gibt, muss es sich um die Begrenzung des Polygonnetzes handeln und das Polygonnetz ist nicht geschlossen.



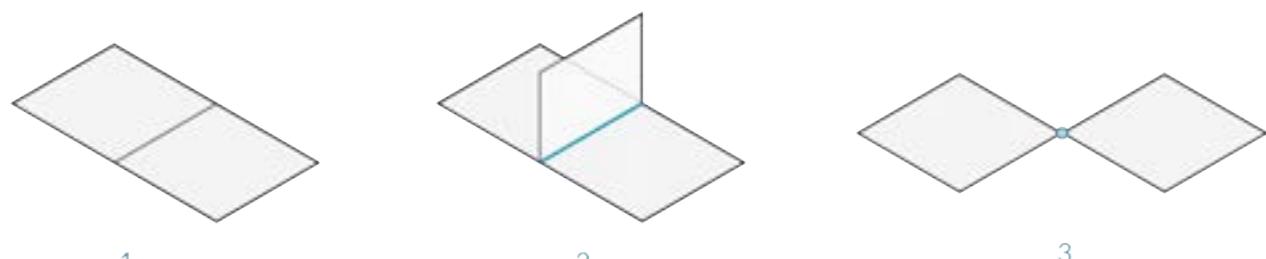
1

2

1. Ein geschlossenes Polygonnetz. Alle Kanten sind mit genau zwei Netzflächen benachbart.
2. Ein offenes Polygonnetz. Die weiße Kanten grenzen nur an eine Netzfläche an.

### Mannigfaltig oder Nicht-Mannigfaltig

Nicht-mannigfaltige Geometrien sind essentiell Geometrien, die nicht in der "realen Welt" existieren können. Das bedeutet nicht notwendigerweise, dass sie "schlechte Geometrie" erzeugen, aber dass man darauf achten muss, da es Schwierigkeiten für Werkzeuge und Operationen bedeuten kann (z.B.: das Rendern von lichtbrechenden Effekten, Fluidsimulationen, boolsche Operationen, 3d Druck). Häufige Bedingungen, die in nicht-mannigfaltigen Polygonnetzen resultieren beinhalten: Selbstverschneidung, offene Kanten (von Löchern oder innenliegenden Netzflächen), unverbundene Topologie und überlappende/duplizierte Flächen. Ein Polygonnetz kann auch als *mannigfaltig* betrachtet werden, wenn es Eckpunkte beinhaltet, die von Netzflächen geteilt werden, die keine Janten miteinander teilen oder deren Valenz größer als 2 ist und somit Verbindung mit mindestens drei Netzflächen herstellen.



1

2

3

1. Ein einfaches mannigfaltiges Polygonnetz
2. Drei Netzflächen treffen sich in einer einzelnen Kante und sind nicht-mannigfaltig, stellen also eine T-Verbindung her

- 3. Zwei Netzflächen treffen sich in einem Punkt aber teilen keine Kante und sind deshalb nicht-mannigfaltig

### 1.6.2.3 Polygonnetze oder NURBS

Inwiefern sind Polygonnetzgeometrien unterschiedlich von NURBS Geometrien? Wann ist es besser die eine oder die andere zu nutzen?

#### Parametrisierung

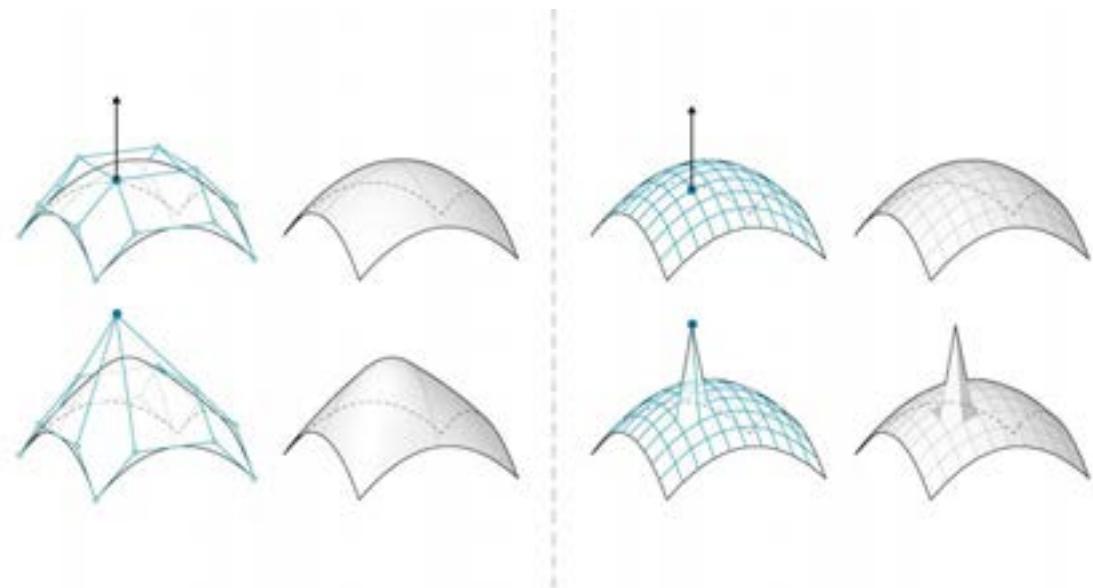
Im einem früheren Kapitel, haben wir gesehen dass NURBS Flächen durch eine Serie von NURBS Kurven in beiden Richtungen definiert sind. Diese Richtungen sind mit U und V benannt und ermöglichen eine NURBS Fläche entsprechend einer zweidimensionalen Domäne zu parametrisieren. Die Kurven selbst werden als Gleichungen im Computer gespeichert, wodurch die resultierende Fläche mit beliebiger Präzision berechnet werden kann. Das verbinden zweier NURBS Flächen wird in einer Polyfläche resultieren, in der verschiedene Schnitte der Geometrie verschiedene UV Parameter und Kurvendefinitionen haben werden.

Polygonnetze auf der anderen Seite, werden durch eine konkrete Anzahl von genau definierten Eckpunkten und Netzflächen bestimmt. Das Netzwerk von Eckpunkten wird generell nicht mit einfachen UV Koordinaten beschrieben werden können. Die Darstellungspräzision kann nun nur durch Verfeinerung des Polygonnetzes oder das hinzufügen weiterer Netzflächen verbessert werden, weil die Netzflächen die Darstellungspräzision durch ihre Darstellung klar definieren. Das Fehlen von UV Koordinaten jedoch ermöglicht es Polygonnetze komplizierterer Geometrie flexibel in einem Polygonnetz zu handhaben, statt wie beim Fall der NURBS Darstellung auf eine Polyfläche zurückgreifen zu müssen.

Anmerkung - Während ein Polygonnetz keine implizite UV Parametrisierung hat, ist es manchmal sinnvoll eine solche Parametrisierung anzuwenden, um eine Textur oder ein Bild auf eine Polygonnetzgeometrie anzuwenden um sie zu rendern. Manche Modellierungsprogramme haben deshalb die Möglichkeit UV Koordinaten für Polygonnetze als *Attribute* (ähnlich wie Eckpunktfarben) den Eckpunkten zuzuweisen, die dann bearbeitet und verändert werden können. Diese sind normalerweise zugewiesen und werden nicht durch das Polygonnetz selbst bestimmt.

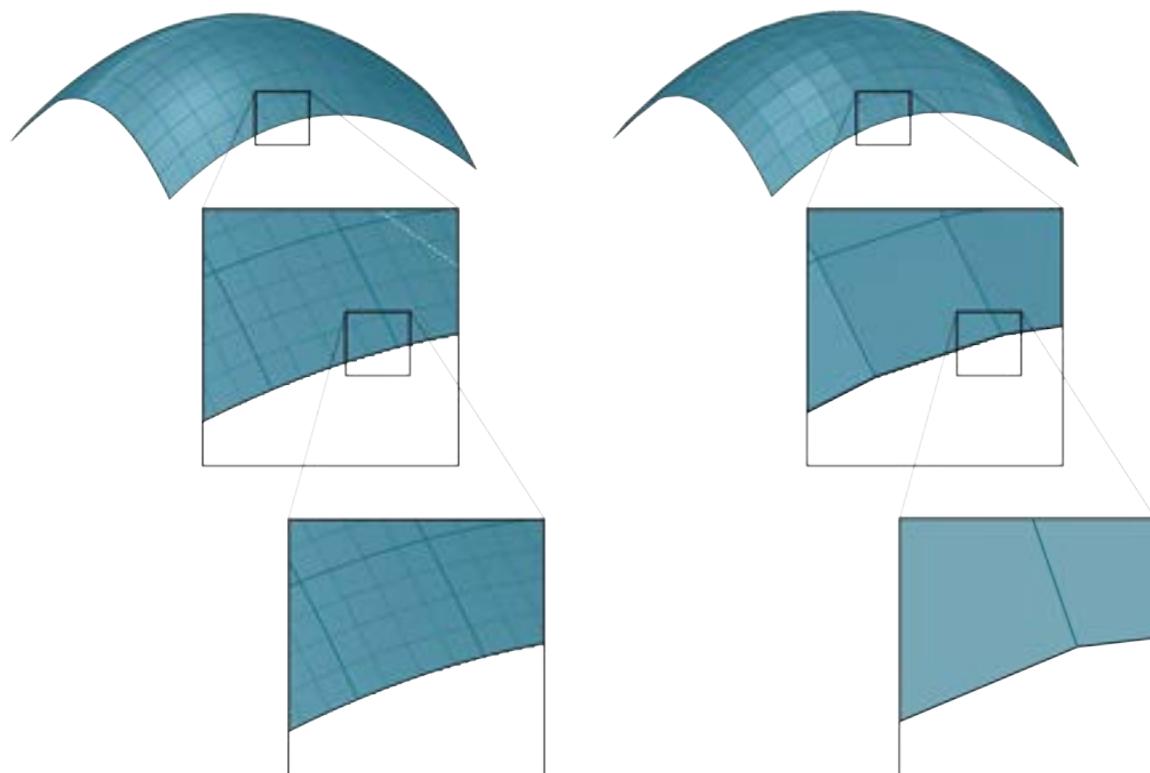
#### Lokale und Globale Einflüsse

Ein wichtiger Unterschied ist der Umfang, in welchem lokale Veränderungen in einem Polygonnetz oder einer NURBS Geometrie die Gesamtform beeinflussen. Polygonnetzgeometrien sind komplett lokal. Einen Eckpunkt zu bewegen beeinflusst lediglich angrenzende Netzflächen. Bei NURBS Flächen, wird der Umfang des Einflusses einer Veränderung komplizierter und hängt vom Graph der Fläche und den Gewichtungen und Knoten der Kontrollpunkte ab. Generell jedoch wird die Bewegung eines einzelnen Kontrollpunktes einer NURBS Fläche eine globale Veränderung der Geometrie bedeuten.



1. NURBS Flächen - die Veränderung eines Kontrollpunktes hat globale Wirkung
2. Polygonnetzgeometrie - die Bewegung eines Eckpunktes hat lokalen Einfluss

Eine Analogie, die hier hilfreich ist, ist der Vergleich einer Vektorgrafik (bestehend aus Linien und Kurven) mit einem Rasterbild (bestehend aus individuellen Punkten). Wenn Du auf ein Vektorbild zoomst, werden die Kurven scharf und klar bleiben, während die Vergrößerung eines Rasterbildes in der Ansicht individueller Pixel enden wird. In dieser Analogie kann eine Vektorgraphik mit einer NURBS Fläche verglichen werden, während sich ein Polygonnetz wie ein Rasterbild verhält.



Vergroesserung einer NURBS Flaeche erhaelt die geglaettete Kurve, waehrend das Polygonnetzelement eine feste Aufloesung hat

Es ist interessant festzustellen, dass waehrend NURBS Flaechen als mathematische Gleichungen gespeichert werden, die letztendliche Darstellung der Flaechen Polygonnetze benoetigt. Es ist nicht moeglich in einem Computer eine kontinuierliche Gleichung darzustellen. Statt dessen muss er die Gleichungen in kleinere Teile herunterbrechen, was darin resultiert, dass fuer jede Render- und Darstellungsaktion ein Polygonnetz der entsprechenden NURBS Flaeche berechnet werden muss. In unserer Analogie koennen wir sehen, dass der Computer, auch wenn er die Gleichung einer Kurve speichern kann, diese in eine Serie von diskreten Pixeln auf dem Bildschirm umwandeln muss, um sie darzustellen.

### 1.6.2.4 Vor und Nachteile von Polygonnetzen

Wenn wir uns fragen "Was sind die Vor- und Nachteile der Modellierung mit Polygonnetzen?" muessen wir uns eigentlich Fragen "Was sind die Vor- und Nachteile einer Modellierung mit Formen die nur durch eine Menge von Eckpunkten und dem entsprechenden topologischen Netzwerk bestimmt werden?". Durch diese Methode der Rahmung der Frage ist es einfacher zu sehen wie die "einfache" Natur von Polygonnetzen der ausschlaggebende Aspekt fuer die Bevorzugung einer Modellierung in dieser Technik ist, abhaengig vom Kontext der entsprechenden Anwendung.

Polygonnetz koennen in einer Situation von Vorteil sein, in der:

- Das dynamische Rendering einer Formveraenderung ohne Veraenderung der Topologie angestrebt wird
- Eine diskrete Annaeherung einer gerundeten Geometrie ausreichend ist
- Eine Geometrie niedrigerer Auflösung systematisch geglättert (oder artikuliert) wird, indem computerbasierte Methoden angewendet werden um eine höhere Auflösung zu erzielen
- Wenn das niedrig aufgelöste Modell gleichzeitig hohe Auflösung von Detail unterstützen muss

Polygonnetze koennen in folgenden Situationen von Nachteil sein:

- Wenn Krümmung und Glättung mit einem hohen Level an Präzision dargestellt werden müssen
- Wahre Ableitungen berechnet werden müssen
- Die Geometrie in einen Körper für industrielle Produktion umgewandelt werden muss
- Die finale Form einfache Veränderungen zulassen soll

### ### 1.6.3 Polygonnetze erstellen

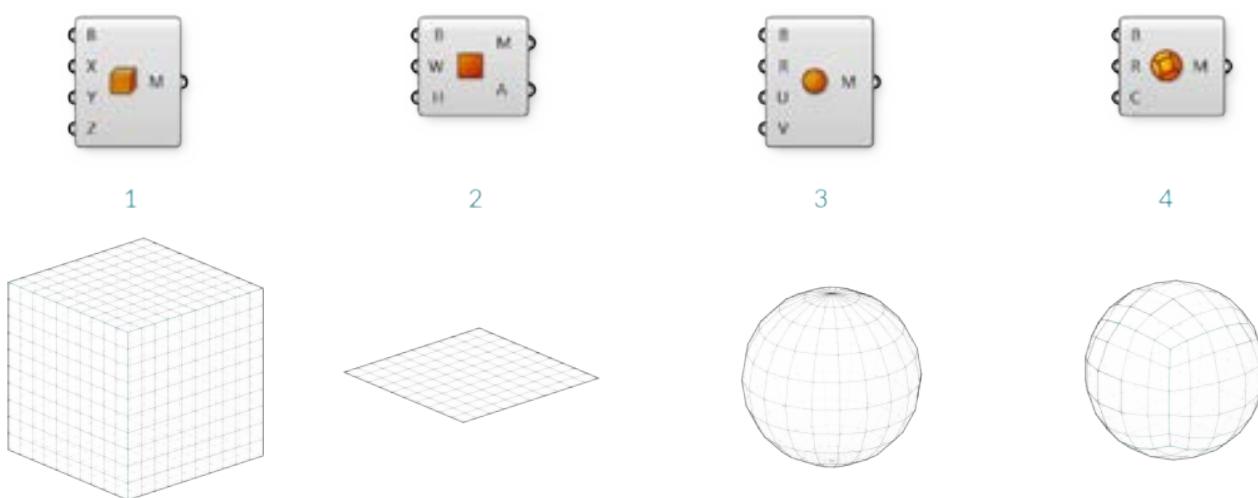
Im letzten Abschnitt, haben wir uns die grundlegende Struktur von Polygonnetzen angesehen. In diesem Schnitt, werden wir eine kurze Einfuehrung in die verschiedenen Wege geben eine Polygonnetzgeometrie zu erzeugen.

Es gibt drei fundamentale Wege um ein Polygonnetz in Grasshopper zu erzeugen:

1. Mit einem primitiven Polygonnetzkoerper beginnen
2. Haendisch ein Polygonnetz von Netzflaechen und Eckpunkten erzeugen
3. Umwandeln von NURBS Geometrien in Polygonnetze

#### 1.6.3.1 Primitive Koerper

Grasshopper kommt mit ein paar einfachen primitiven Polygonnetzkomponenten:



1. **Mesh Box** - Dieser primitive Koerper benoetigt ein Kistenobjekt als Eingabeparameter, der die Groesse und Positon, sowie die X, Y und Z Werte, welche die Anzahl von Netzflaechen bestimmen, uebergibt. Die sechs Seiten der Polygonnetzkiste sind "unverschweisst" um Knicke zu erlauben. (Im nachfolgenden Abschnitt werden wir mehr Informationen zu verschweissten Polygonnetzen liefern.)
2. **Mesh Plane** - Dieser primitive Koerper benoetigt ein Rechteck als Eingabeparameter um die Groesse und Pasion der Ebene zu bestimmen, sowie die W und H Werte um die Anzahl der Netzflaechen zu definieren.
3. **Mesh Sphere** - Dieser primitive Koerper benoetigt eine Basisebene um das Zentrum und die Orientierung der Kugel zu definieren, den Radius fuer die Groesse und U/V Werte um die Anzahl der Netzflaechen festzulegen.
4. **Mesh Sphere Ex** - Auch als "Quadball" bekannt, erzeugt dieser primitive Koerper eine Kugel, die aus sechs Abschnitten besteht, die entsprechend dem C Eingabeparameter unterteilt werden. Ein Quadball ist topologisch gleich einem Wuerfel, auch wenn er geometrisch eine Kugel ist.

#### 1.6.3.2 Polygonnetze konstruieren



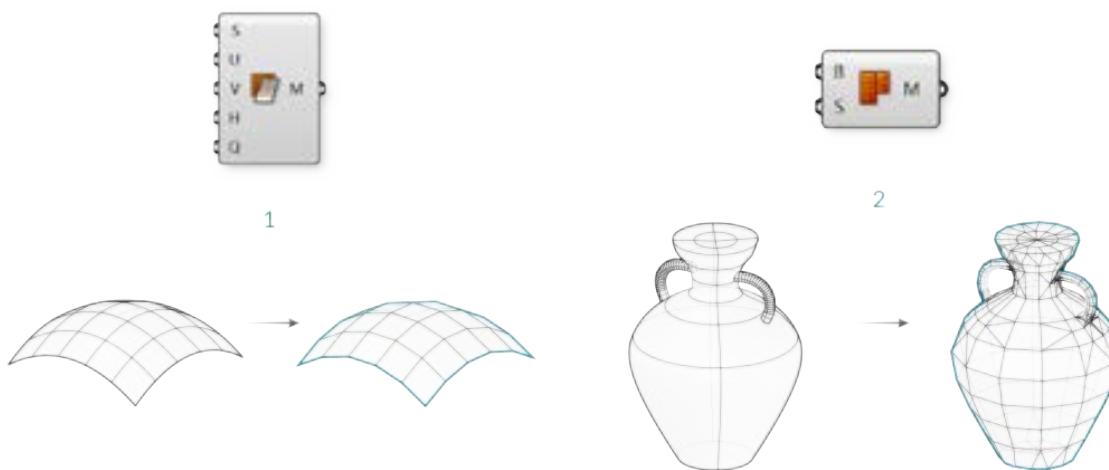
Wir haben im vorangegangenen Abschnitt gesehen, dass die **Construct Mesh** Komponente genutzt reden kann,

um ein Polygonnetz aus einer Liste von Eckpunkten und einer Liste von Netzflächen (und einer optionalen Liste von Farben) zu erzeugen. Ein Polygonnetz handisch zu konstruieren kann sehr mühselig sein, weshalb diese Komponente oft genutzt wird um eine bestehende Liste von Netzflächen und Eckpunkten bearbeiten, die mit der **Deconstruct Mesh** Komponente aus einem bestehenden Polygonnetz extrahiert wurde.

### 1.6.3.3 NURBS zu Polygonnetz

Vielleicht die häufigst genutzte Methode zur Erstellung eines komplexen Polygonnetzes ist die Erzeugung von einer NURBS Geometrie. Individuelle NURBS Flächen können mit der **Mesh Surface** Komponente in Polygonnetze umgewandelt werden, die einfach Flächen entlang ihrer UV Koordinaten unterteilt und viereckige Netzfläche erzeugt. Diese Komponente ermöglicht es eine Zahl für die Unterteilung in U und V Richtung für das resultierende Polygonnetz einzugeben.

Komplexere Polyflächen können nicht mit der **Mesh Brep** Komponente in ein einzelnes Polygonnetz umgewandelt werden. Diese Komponente hat einen Eingabeparameter für optionale Einstellungen, die mit durch die eingebauten Optionen für *Geschwindigkeit* und *Qualität* oder einer *Custom Settings* Komponente definiert werden können. Eine andere Möglichkeit ist es mit einem Rechtsklick auf den S Eingabeparameter die option "Set Mesh Options" zu nutzen. Für die effiziente Nutzung von Polygonnetzen, ist es oftmals notwendig diese zu verfeinern, indem Strategien, wie wiederholen, glätten und unterteilen angewendet werden. Einige dieser Techniken werden wir später in diesem Primer beschreiben.



1. **Mesh Surface** wandelt eine NURBS Fläche in ein Polygonnetz um
2. **Mesh Brep** kann Polyflächen und kompliziertere Geometrien in ein einzelnes Polygonnetz umwandeln.  
Die Anpassung der Einstellungen ermöglicht es mit mehr oder weniger Netzflächen ein feineres oder gröberes Polygonnetz darzustellen.

**MERKE:** Es ist generell viel leichter eine NURBS Geometrie in ein Polygonnetz umzuwandeln, als anders herum. Während die UV Koordinaten einer NURBS Fläche eine einfache Umwandlung in viereckige Netzflächen erlaubt, ist die entgegengesetzte Abbildung nicht immer eindeutig möglich, da ein Polygonnetz aus dreieckigen und viereckigen Netzflächen bestehen kann, die nicht die einfache Extraktion von UV Koordinaten erlauben.

### 1.6.3.4 Übung

In dieser Übung werden wir einfache primitive Polygonkörper nutzen, um eine Transformation der Eckpunkte durchzuführen und dann eine basierte Annäherung der Normalenvektoren in einem Renderingprozess darzustellen.

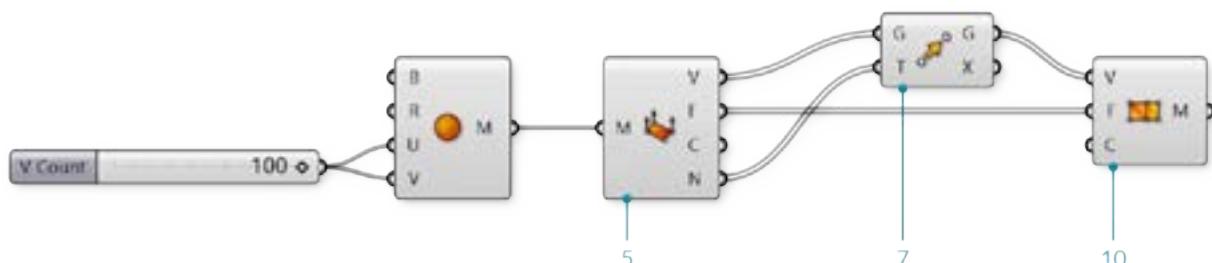
Beispieldateien für diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

01.	Beginne eine neue Definition, druecke Strg-N (in Grasshopper)	
02.	<b>Mesh/Primitive/Mesh Sphere</b> - Ziehe eine <b>Mesh Sphere</b> Komponente auf die Leinwand	
03.	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand und setze die folgenden Werte: Rounding: Integer Lower Limit:0 Upper Limit: 100 Value: 10	
04.	Verbinde den <b>Number Slider</b> mit dem Anzahl U (U) und dem Anzahl V (V) Eingabeparameter der <b>Mesh Sphere</b> Komponente	



Passe den Slider an und beobachte die Veraenderung der Aufolesung der Kugel im Rhinoansichtsfenster.  
Hoehere Werte resultieren in einer glatteren Kugel, aber erzeugen auch groessere Datensaetze, die dann hoehere Rechenzeiten zur Folge haben.

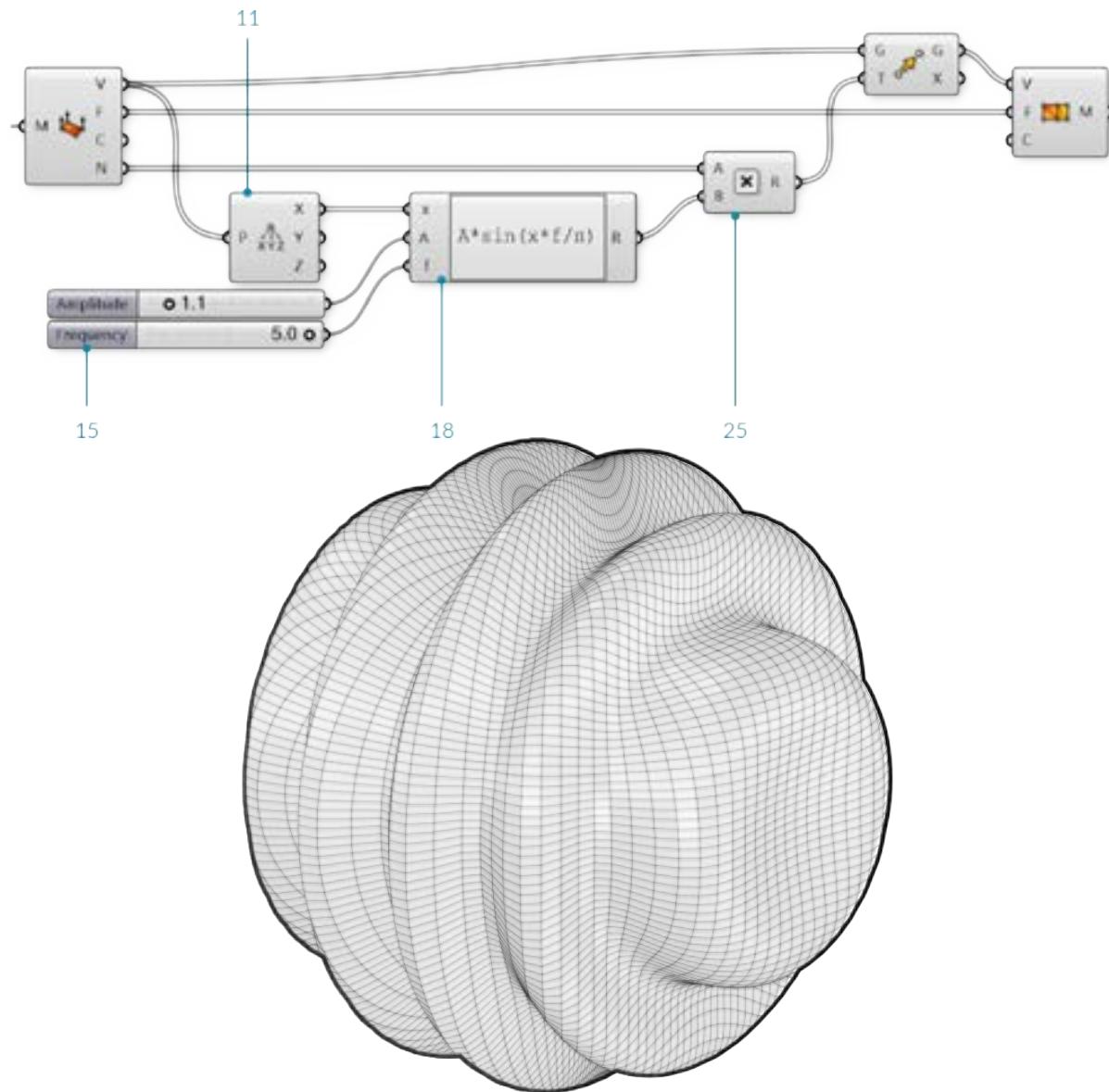
05.	<b>Mesh/Analysis/Deconstruct Mesh</b> - Ziehe eine <b>Deconstruct Mesh</b> Komponente auf die Leinwand	
06.	Verbinde den Polygonnetz (M) Ausgabeparameter der <b>Mesh Sphere</b> Komponente mit dem Polygonnetz (M) Eingabeparameter der <b>Deconstruct Mesh</b> Komponente	
07.	<b>Transform/Euclidean/Move</b> - Ziehe eine <b>Move</b> Komponente auf die Leinwand	
08.	Verbinde den Eckpunkte (V) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem Geometrie (G) Eingabeparameter der <b>Move</b> Komponente	
09.	Verbinde den Normalen (N) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem Bewegungsvektor (T) Eingabeparameter der <b>Move</b> Komponente	
10.	<b>Mesh/Analysis/Construct Mesh</b> - Ziehe eine <b>Construct Mesh</b> Komponente auf die Leinwand	
11.	Verbinde den Geometrie (G) Ausgabeparameter der <b>Move</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Construct Mesh</b> Komponente	
12.	Verbinde den Netzflaechen (F) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem Netzflaeche Eingabeparameter (F) der <b>Construct Mesh</b> Komponente	



Wir dekonstruieren ein Polygonnetz um seine Eckpunkte, Netzaeichen und Normalen zu erhalten. Wir bewegen dann einfach jeden Eckpunkt entsprechend seines Normalenvektors. Da wir nicht die Topologie der Kugel veraendert haben, koennen wir die Liste der Netzaeichen wieder verwenden um das neue Polygonnetz zu erzeugen. Normalenvektoren haben eine Laenge von eins. Somit haben wir eine neue Kugel mit einem um eins groesseren Radius als die urspruegliche Kugel erzeugt.

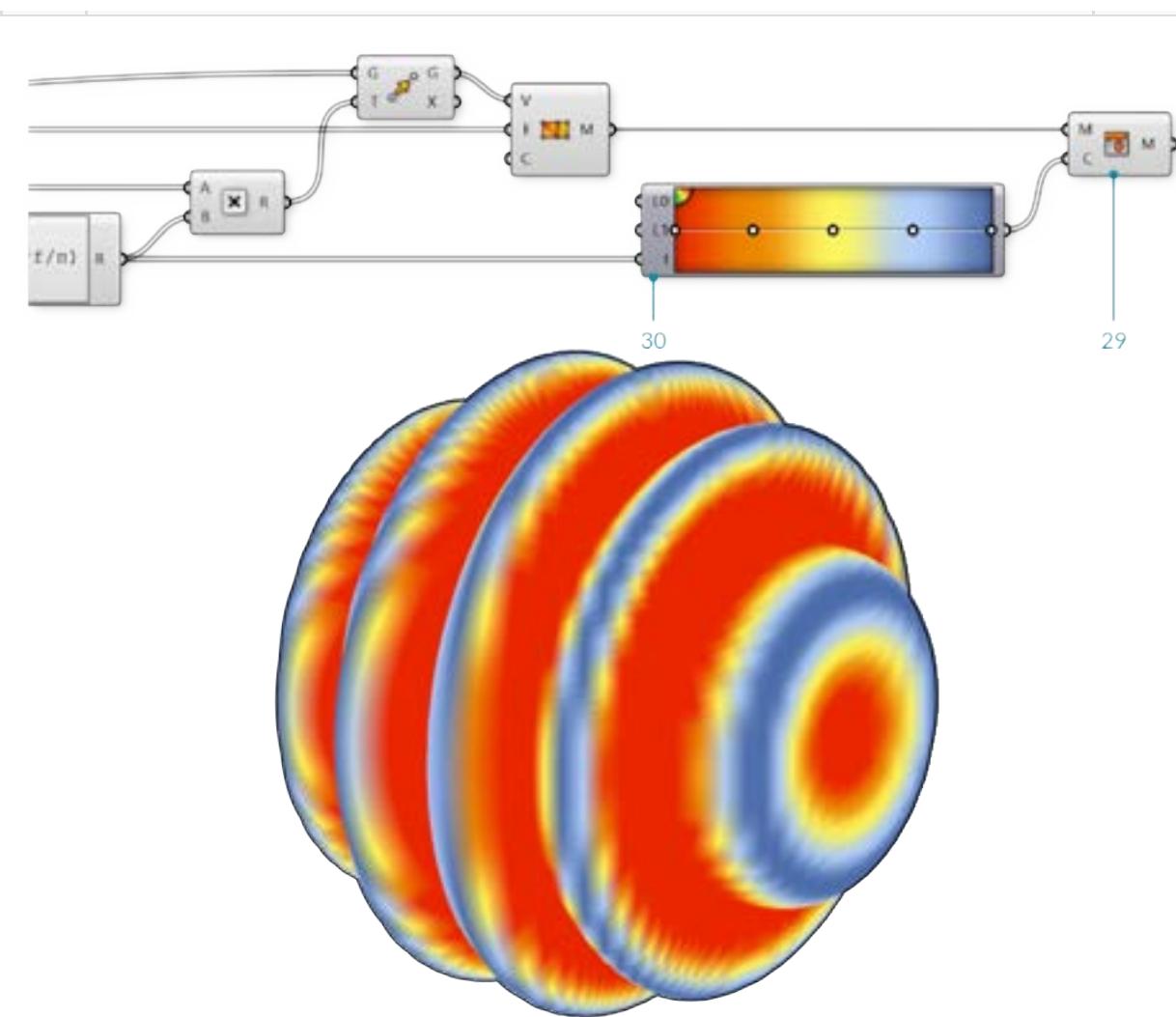
Als naechstes werden wir eine Sinusfunktion nutzen um eine Kugel auf eine komplexere Art zu manipulieren.

13.	<b>Vector/Point/Deconstruct</b> - Ziehe eine <b>Deconstruct</b> Komponente auf die Leinwand	
14.	Verbinde den Eckpunkte (V) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem Punkt (P) Eingabeparameter der <b>Deconstruct</b> Komponente	
15.	<b>Params/Input/Number Slider</b> - Ziehe zwei <b>Number Slider</b> Komponenten auf die Leinwand	
16.	Setze die Werte der ersten <b>Number Slider</b> auf: Name: Amplitude Rounding: Float Lower Limit: 0 Upper Limit: 10	
17.	Setze die Werte des zweiten <b>Number Slider</b> auf: Name: Frequency Rounding: Float Lower Limit: 0 Upper Limit: 5	
18.	<b>Maths/Script/Expression</b> - Ziehe eine <b>Expression</b> Komponente auf die Leinwand	
19.	Zoom auf die <b>Expression</b> Komponente, bis Du die Optionen zum Hinzufuegen und Entfernen von Eingabeveriablen sehen kannst und druecke auf '+' um eine 'z' Variable hinzuzufuegen	
20.	Rechtsklicke auf den 'y' Eingabeparameter der <b>Expression</b> Komponente und aendere den Text auf 'A'	
21.	Rechtsklicke auf den 'z' Eingabeparameter der <b>Expression</b> Komponente und aendere den Text auf 'f'	
22.	Doppelklicke die <b>Expression</b> Komponente und passe den Ausdruck an, indem Du folgenden folgende Gleichung eingibst: $A1 * \sin(x * f / \pi)$	
23.	Verbinde den X Ausgabeparameter der <b>Deconstruct</b> Komponente mit dem 'x' Eingabeparameter der <b>Expression</b> Komponente	
24.	Verbinde den Amplitude <b>Number Slider</b> mit dem AEingabeparameter und den Frequenz <b>Number Slider</b> mit dem 'f' Eingabeparameter der <b>Expression</b> Komponente	
25.	<b>Maths/Operators/Multiplication</b> - Ziehe eine <b>Multiplication</b> Komponente auf die Leinwand	
26.	Verbinde den Normalen (N) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem AEingabeparameter der <b>Multiplication</b> Komponente	
27.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Expression</b> Komponente mit dem B Eingabeparameter der <b>Multiplication</b> Komponente	
28.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Multiplication</b> Komponente mit dem Bewegungsvektor (T) Eingabeparameter der <b>Move</b> Komponente	



Passe die Amplitude des Frequenz Schiebereglers um zu sehen, wie sich das neu konstruierte Polygontetz verändert.

29.	<b>Mesh/Primitive/Mesh Colours</b> - Drag and drop a <b>Mesh Colours</b> component onto the canvas	
30.	<b>Params/Input/Gradient</b> - Drag and drop a <b>Gradient</b> component onto the canvas	
31.	You can right-click the gradient component and select "Presets" to change the color gradient. In this example, we used the Red-Yellow-Blue gradient	
32.	Connect the Result (R) output of the <b>Expression</b> component to the Parameter (t) input of the <b>Gradient</b> component	
33.	Connect the output of the <b>Gradient</b> component to the Colours (C) input of the <b>Mesh Colours</b> component	
	Connect the Mesh (M) output of the <b>Construct Mesh</b> component to the Mesh (M) input of the <b>Mesh Colours</b> component	
33.	In this step, we could achieve the same result by connecting the gradient directly to the Colours (C) input of the **Construct Mesh** component	

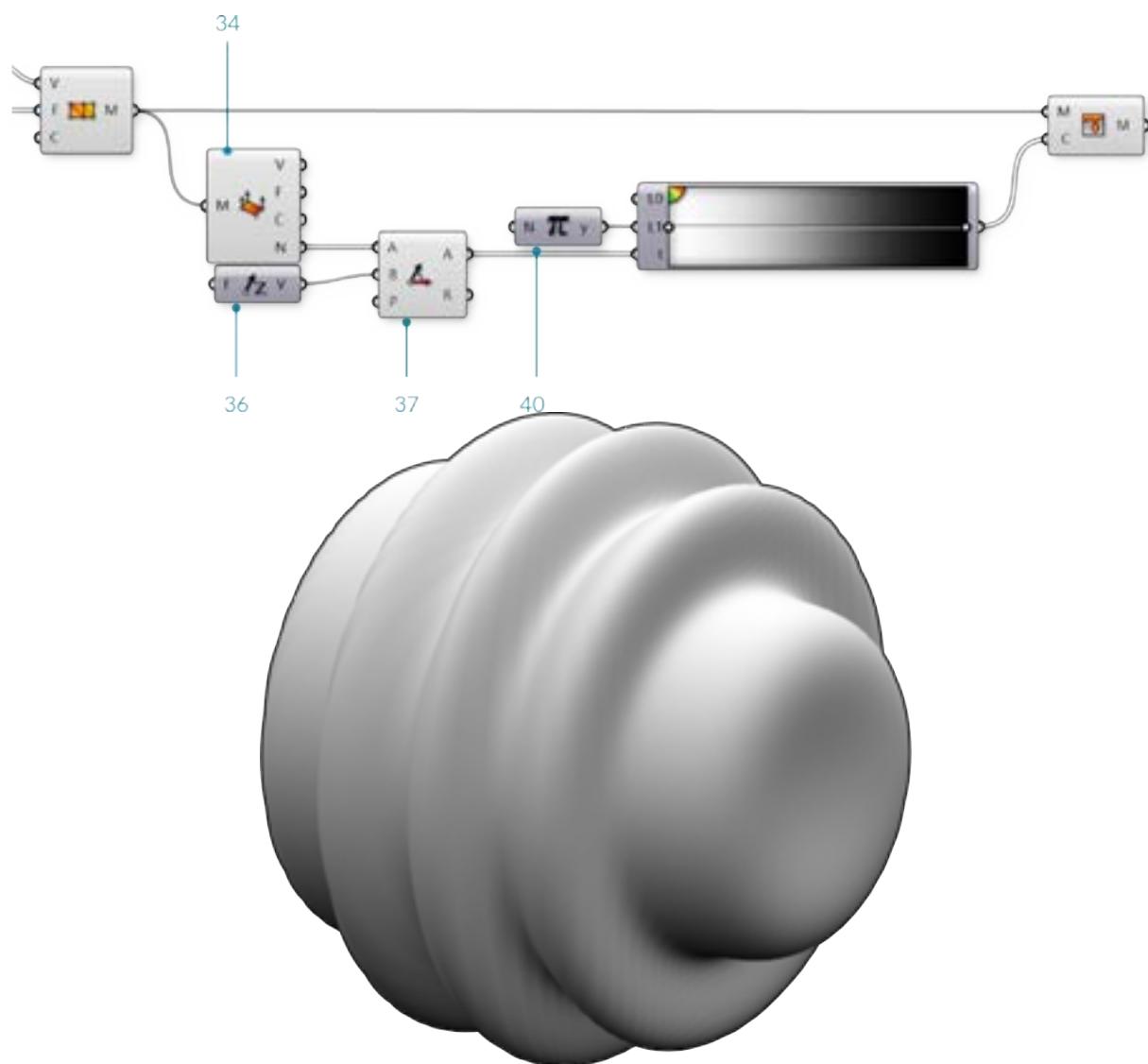


Wir haben die Ergebnisse der Gleichung benutzt, um die Bewegung der Eckpunkte und die Farbe des Polygonnetzes zu veraendern, so dass der Gradient in diesem Fall mit dem Umfang der Bewegung der Eckpunkte uebereinstimmt.

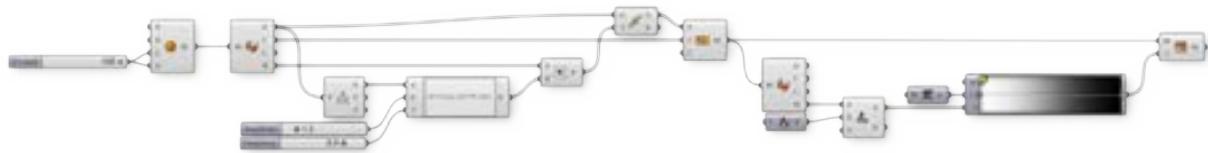
Fuer den letzten Abschnitt dieser Uebung werden wir statt dessen die Richtung der Normalen relativ zum Vektor einer Lichtquelle heranziehen um den grundlegende Prozess des Renderings eines Polygonnetzes darzustellen.

34.	<b>Mesh/Analysis/Deconstruct Mesh</b> - Ziehe eine <b>Deconstruct Mesh</b> Komponente auf die Leinwand	
	Verbinde den Polygonnetz (M) Ausgabeparameter der <b>Construct Mesh</b> Komponente mit dem Polygonnetz (M) Eingabeparameter der <b>Deconstruct Mesh</b> Komponente	
35.	<p>Während die Topologie des ursprünglichen Polygonnetzes nicht veraendert wurde, werden sich die Normalenvektoren veraendern, weshalb wir eine neue <b>Deconstruct Mesh</b> Komponente benoetigen um die neuen Normalen zu ermitteln.</p>	
36.	<b>Vector/Vector/Unit Z</b> - Ziehe eine <b>Unit X</b> Komponente auf die Leinwand <p>Wir werden diesen Vektor als Richtung der Lichtquelle nutzen. Du kannst andere Vektoren nutzen oder eine Linie von Rhino referenzieren um die Uebung dynamischer zu gestalten</p>	

37.	<b>Vector/Vector/Angle</b> - Ziehe eine <b>Angle</b> Komponente auf die Leinwand	
38.	Verbinde den Normalen (N) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem A Eingabeparameter der <b>Angle</b> Komponente	
39.	Verbinde den Ausgabeparameter der <b>Unit Z</b> Komponente mit dem B Eingabeparameter der <b>Angle</b> Komponente	
40.	<b>Maths/Util/Pi</b> - Ziehe eine <b>Pi</b> Komponente auf die Leinwand	
41.	Verbinde die <b>Pi</b> Komponente mit dem Obergrenze (L1) Eingabeparameter der <b>Gradient</b> Komponente	
42.	Verbinde den Winkel (A) Ausgabeparameter der <b>Angle</b> Komponente mit dem Parameter (t) Eingabeparameter der <b>Gradient</b> Komponente	



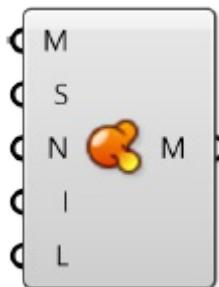
Wir haben eine weiss-nach-schwarz Voreinstellung fuer unseren Gradienten. Dieser setzt die Meshfarbe entsprechend dem Winkel zwischen der Normalen und der Lichtquelle mit Normalen die direkt auf die Lichtquelle ausgerichtet sind auf schwarz und Normalen, die von der Lichtquelle wegzeigen in weiss (um etwas genauer zu sein, koennen wir den Gradienten umkehren, indem wir die Griffe anpassen). Der eigentliche Prozess ein Mesh zu rendern ist viel komplizierter, aber dies ist der grundlegende Prozess um Licht und Schatten auf gerenderten Objekten zu erzeugen.



## 1.6.4 Polygonnetzoperationen

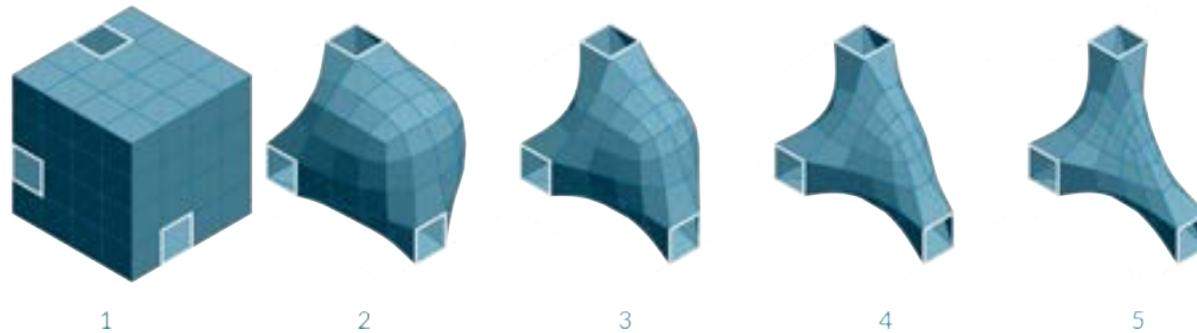
Im letzten Abschnitt haben wir die grundlegende Struktur von Polygonnetzen angeschaut. In diesem Abschnitt werden wir Wege ansehen um Polygonnetzgeometrien zu manipulieren.

### 1.6.4.1 Glaettung



Glaettere Polgonnetze koennen erreicht werden, indem die Anzahl der Netzaechen erhoeht wird. Dieser Prozess wird *Unterteilung* genannt. Dies kann oft zu sehr grossen Datensaetzen fuhren, welche viel Rechenzeit und zusätzliche Erweiterungen fuer Grasshopper benoetigen. In dieser Situation kann die **Smooth** Komponente als Alternative genutzt werden, um weniger gacettierte Meshes zu erzeugen ohne die Anzahl der Eckpunkte und Netzaechen zu erhöhen oder die Topologie zu veraendern. Die **Stärke**, **Anzahl der Iterationen** und **Versatzbegrenzung** koennen genutzt werden um die Art zu beeinflussen, in der die Glaettung passiert.

Verbindung eines boolschen Wertes mit einem Eingabeparameter N gibt uns die Option freie Eckpunkte zu ueberspringen. Exkpunkte sind frei, wenn sie mit einer offenen Kante verbunden sind, mit der Bedeutung dass er sich in der aeusseren Begrenzung des Polygonnetzes befindet. Indem Du diese Option einschaltet, kannst Du die aeusseren Begrenzungen eines Polygonnetzes erhalten, während Du die inneren Kanten des Polygonnetzes glaettst.

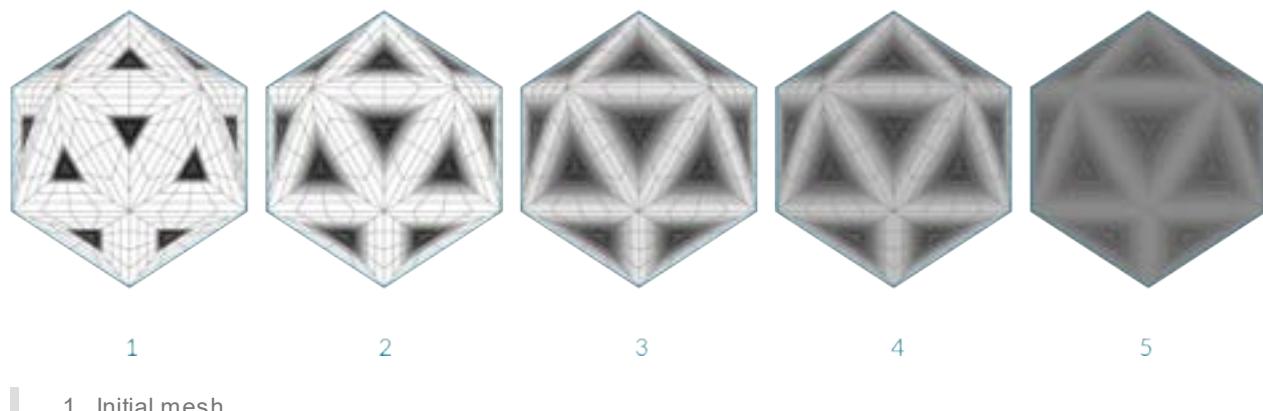


1. Urspraeliches Polygnnetz einer Kiste mit drei Netzaechen entfernt
2. Glaettung nach 2 Iterationen
3. 6 Iterationen
4. 25 Iterationen
5. 50 Iterationen

### 1.6.4.2 Unschaerfe

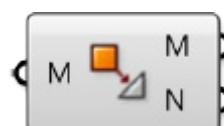


Die **Blur** Komponente verhält sich ähnlich wie die "Smooth" Komponente, abgesehen von den Eckpunktfarben. Es kann auch benutzt werden um die gezackte Erscheinung von Farbpolygonnetzen, auch wenn der Effekt geringer ist, da die Geometrie nicht verändert wird.

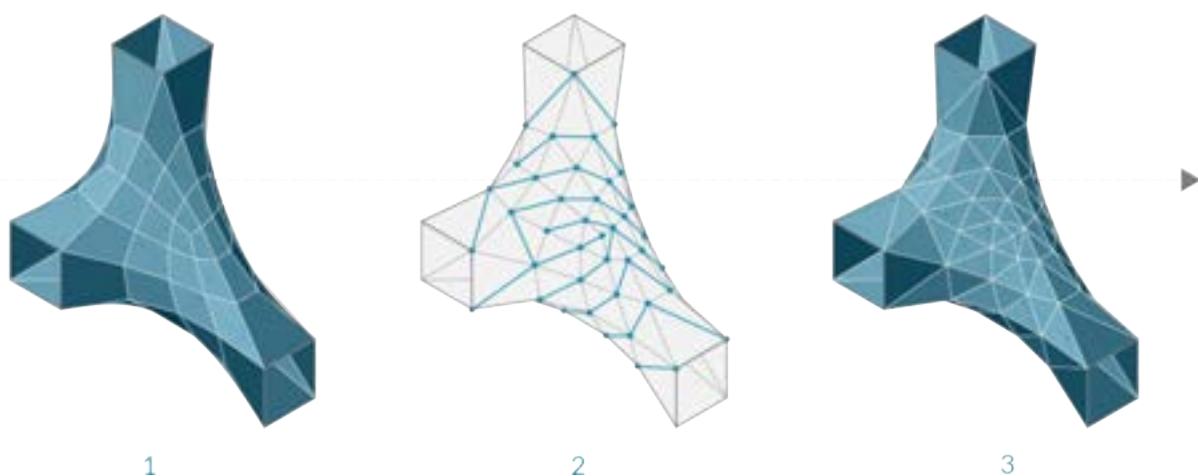


1. Initial mesh
2. Blur after 1 iterations
3. 6 iterations
4. 12 iterations
5. 20 iterations

#### 1.6.4.3 Triangulierung

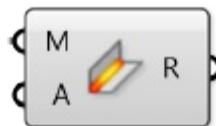


Um die Planarität einer jeden Netzfäche zu sichern oder um das Polygonnetz zu anderen Programmen zu exportieren, die keine viereckigen Netzfächen erlauben, ist es manchmal notwendig ein Polygonnetz zu triangulieren. Mit der Nutzung der **Triangulate** Komponente, wird jede viereckige Netzfäche durch zwei dreieckige ersetzt. Grasshopper nimmt immer die kürzeste Diagonale der Netzfäche um eine Netzfäche mit einer neuen Kante zu unterteilen.

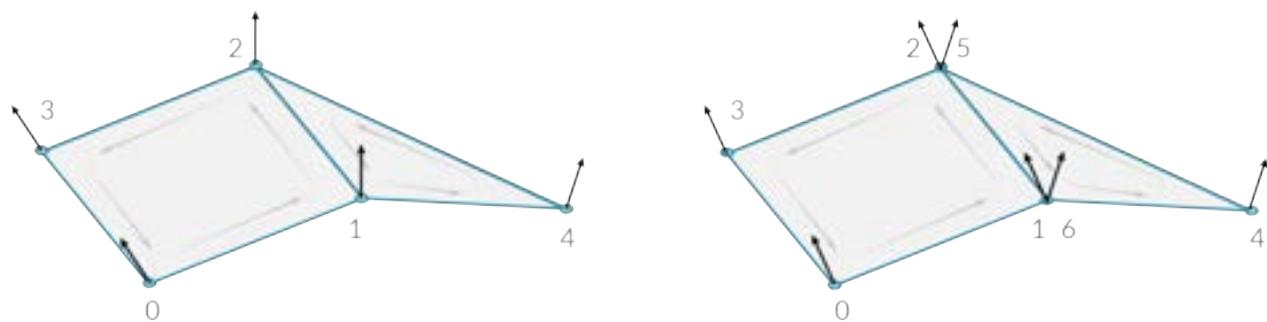


1. Ursprüngliches Polygonnetz mit viereckigen Netzfächen
2. Hinzugefügte Kanten in Übereinstimmung mit der kürzesten Distanz zwischen viereckigen Netzfächen
3. Trianguliertes Polygonnetz als Ergebnis

#### 1.6.4.4 Schweißen



Im letzten Abschnitt, haben wir festgestellt, dass ein einzelner Eckpunkt von benachbarten Netzelementen geteilt werden kann und dass die Normale für diesen Eckpunkt wird aus dem Durchschnitt der Normalen der angrenzenden Netzelemente berechnet, was eine glattere Darstellung gibt. Manchmal ist es jedoch gewünscht, dass eine scharfe Kante oder ein Saum erzeugt werden, an welchen ein glatter Übergang zwischen einer Netzelemente und der benachbarten nicht stattfinden soll. In dieser Situation ist es notwendig für jede Netzelemente an einem Schnittpunkt eine eigene Eckpunktnormale zu definieren. Die Liste von Eckpunkten würde mindestens zwei Punkte enthalten die eine Koordinate teilen aber verschiedene Indizes besitzen.

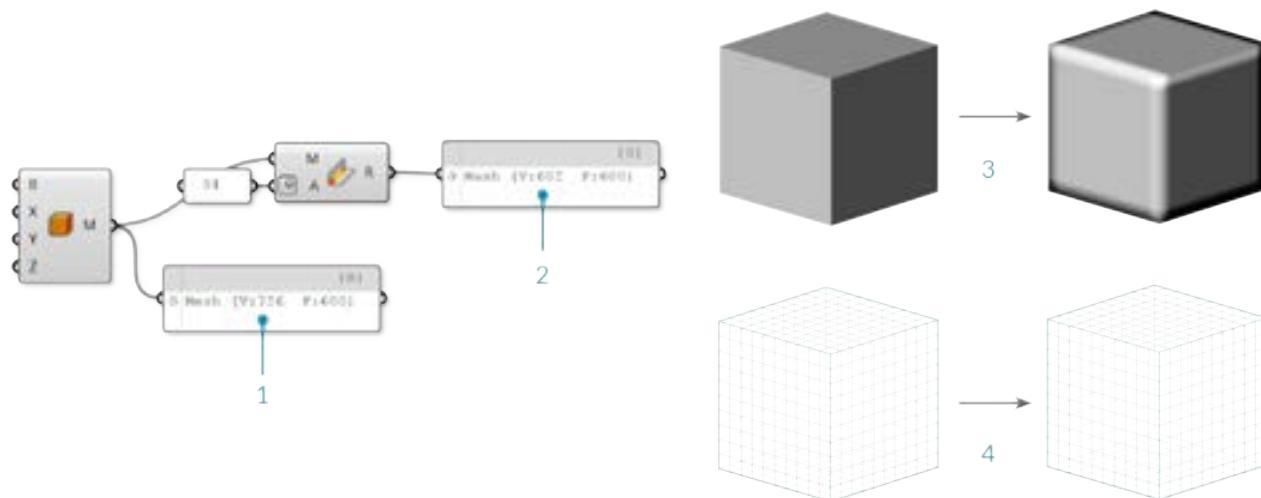


1	Vertex List	Face List
0 = {0.0, 0.0, 0.0}	Q {0, 1, 2, 3}	
1 = {1.0, 0.0, 1.0}	T {1, 4, 2}	
2 = {1.0, 1.0, 1.0}		
3 = {0.0, 1.0, 0.0}		
4 = {2.0, 0.0, -1.0}		

2	Vertex List	Face List
0 = {0.0, 0.0, 0.0}	Q {0, 1, 2, 3}	
1 = {1.0, 0.0, 1.0}	T {4, 5, 6}	
2 = {1.0, 1.0, 1.0}		
3 = {0.0, 1.0, 0.0}		
4 = {2.0, 0.0, -1.0}		
5 = {1.0, 1.0, 1.0}		
6 = {1.0, 0.0, 1.0}		

1. Verschweißte Netzelemente - Beide Netzelemente teilen die Vertices 1 und 2. Die Eckpunktnormalen an diesen Eckpunkten entsprechen dem Durchschnitt der Netzelementennormalen der angrenzenden Netzelemente.
2. Unverschweißte Netzelemente - Duplizierte Eckpunkte wurden der Liste hinzugefügt. Eckpunkte 1 und 6 und Eckpunkte 2 und 5 haben identische Koordinaten, aber unterschiedliche Eckpunkte. Sie haben jeweils ihre eigenen Eckpunktnormalen. Der Prozess zwei Eckpunkte in der selben Position zu nehmen und sie zu einem gemeinsamen Eckpunkt zusammenzuführen nennen wir *schweißen*, wobei *entschweißen* einen einzelnen Eckpunkt und ihn in mehrere Eckpunkte zu teilen.

Die **Weld** Komponente nutzt einen Grenzwinkel als Eingabeparameter. Alle benachbarten Netzelemente mit einem eingeschlossenen Winkel kleiner dem Grenzwinkel werden miteinander verschweißt, wodurch gemeinsame Eckpunkte mit geteilten Eckpunktnormalen entstehen, die die Werte der angrenzenden Flächennormalen mitteln. **Unweld** funktioniert auf die entgegengesetzte Art und Weise, in der angrenzende Netzelemente mit einem Winkel größer dem Grenzwinkel entschweißt werden und die bisher geteilten Eckpunkte dupliziert werden.



1. Die Standardpolygonnetzkiste hat 726 Eckpunkte. Das Polygonnetz ist an den Ecken der Kiste gekantet und die Eckpunkte an diesen Ecken sind dupliziert.
2. Wenn das Polygonnetz mit einem Winkel groesser als 90 Grad verschweisst wird, werden die Netzflächen alle verschweisst und die Anzahl der Endpunkte wurde auf 602 reduziert, während die Anzahl der Netzflächen die selbe bleibt.
3. Wenn wir uns die Vorschau geometrie ansehen, können wir auch feststellen, dass das gerenderte verschweißte Polygonnetz geglättete Ecken hat.
4. Entgegen der "Smooth" Komponente, welche die Polygonnetzgeometrie verändert, wirkt dieses Polygonnetz nur geglättet, weil die Eckpunktnormalen eine Rolle in Rendering und Schattierung spielen. die eigentliche Position der Eckpunkte bleibt unverändert.

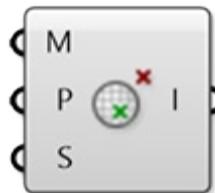
Im oben gezeigten Bild haben wir einen Winkel von 91 Grad angenommen, weil wir wissen, dass die Seiten eines Quadrats 90 Grad Winkel enthält. Um ein Polygonnetz vollständig zu verschweißen solltest Du einen Winkel von 180 Grad angeben.

## 1.6.5 Wechselwirkungen zwischen Polygonnetzen

Dieser Abschnitt betrachtet Wege, in denen Polygonnetzobjekte miteinander in Wechselwirkung treten koennen, wie beispielsweise die Auswertung des naehesten Punktes oder die Kombination mehrerer Polygonnetze miteinander.

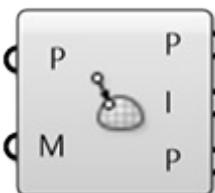
### 1.6.5.1 Polygonnetze und Punkte

#### Inclusion

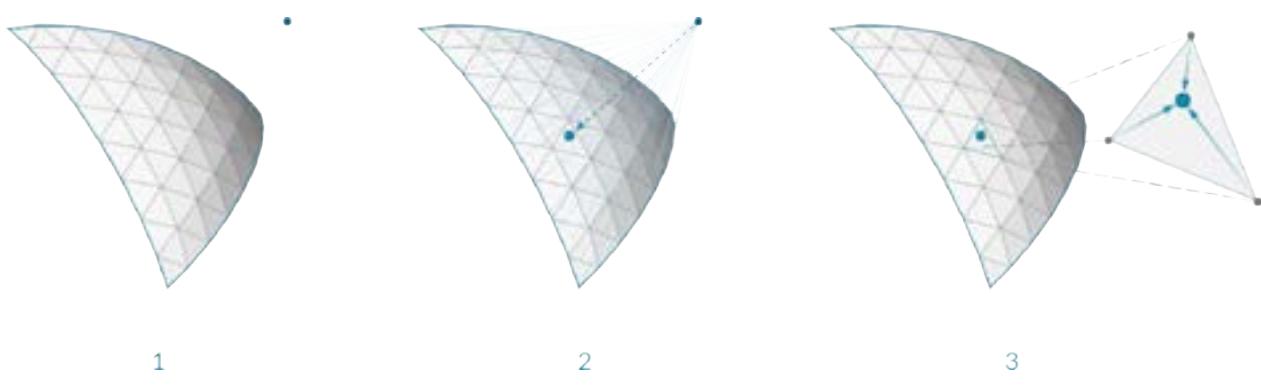


Diese Komponente testet ob ein bestimmter Punkt innerhalb eines Polygonnetzkoerpers liegt oder nicht. Dies funktioniert nur mit geschlossenen Polygonnetzen.

#### Mesh Closest Point



Diese Komponente berechnet die Position auf dem Polygonnetz, welche einem angegebenen Punkt am naehesten liegt. Diese Komponente gibt drei verschiedene Daten aus: die Koordinaten des berechneten Punktes auf dem Polygonnetz, die Indizes der Netzaeche die den Punkt enthaelt und die Polygonnetzparameter. Diese Parameter ist sehr wichtig in Verbindung mit der **Mesh Eval** Komponente, die unterhalb behandelt wird.

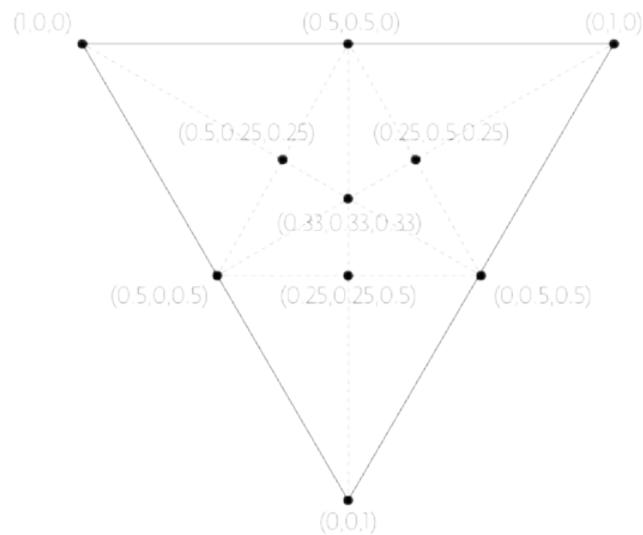


1. Wir wollen basiered auf einem gegebenen Punkt im Raum den naehesten Punkt auf einem Polygonnetz finden
2. Die Netzaeche, welche den naehesten Punkt enthaelt werden bestimmt
3. Die Parameter des naehesten Punktes auf der Flaeche werden berechnet

Nutzer, die sich detaillierter mit der Parametrisierung beschaeftigen wollen koennen sich die Struktur der Polygonnetzparameter genauer ansehen. Du kannst sehen, wenn Du ein Paneel an den entsprechenden Ausgabeparameter der **Mesh Closest Point** Komponente anschliesst. Der Polygonnetzparameter hat die folgende Struktur: N[A,B,C,D]. Die erste Zahl, N, ist der Index der Netzaeche, welche den berechneten Punkt

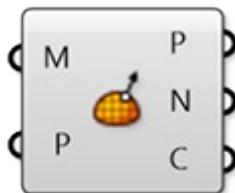
enthaltet.

Die folgenden vier Zahlen definieren die *baryzentrischen* Koordinaten des Punktes innerhalb der Netzfäche. Die Koordinaten des referenzierten Punktes können gefunden werden, indem jeder Eckpunkt des Polygonnetzes mit diesen Zahlen multipliziert wird und die Ergebnisse addiert. (Natürlich wird das schon für uns erledigt und wir können den Punktausgabeparameter nutzen.) Merke Dir, dass baryzentrische Koordinaten nur für dreieckige Netzfächen eindeutig sind, was bedeutet, dass für eine viereckige Netzfäche ein bestimmter Punkt mehrere Parametrisierungen aufweisen kann. Grasshopper vermeidet dieses Problem, indem es viereckige Netzfächen intern trianguliert, wenn es die Netzparameter berechnet, was dazu führt, dass dass von den vier Zahlen des Netzparameters immer mindestens einer 0 ist.



Baryzentrische Koordinaten

### Mesh Eval



Die **Mesh Eval** Komponente nutzt einen Netzparameter als Eingabeparameter und gibt den referenzierten Punkt, seine Normale und Farbe aus. Die Farbe und Normale werden als Interpolation der Eckpunktfarben und -normalen berechnet, indem die selben baryzentrischen Koordinaten wie im Netzparameter benutzt werden.

### 1.6.5.2 Kombination von Polygonnetzgeometrien

#### Mesh Join

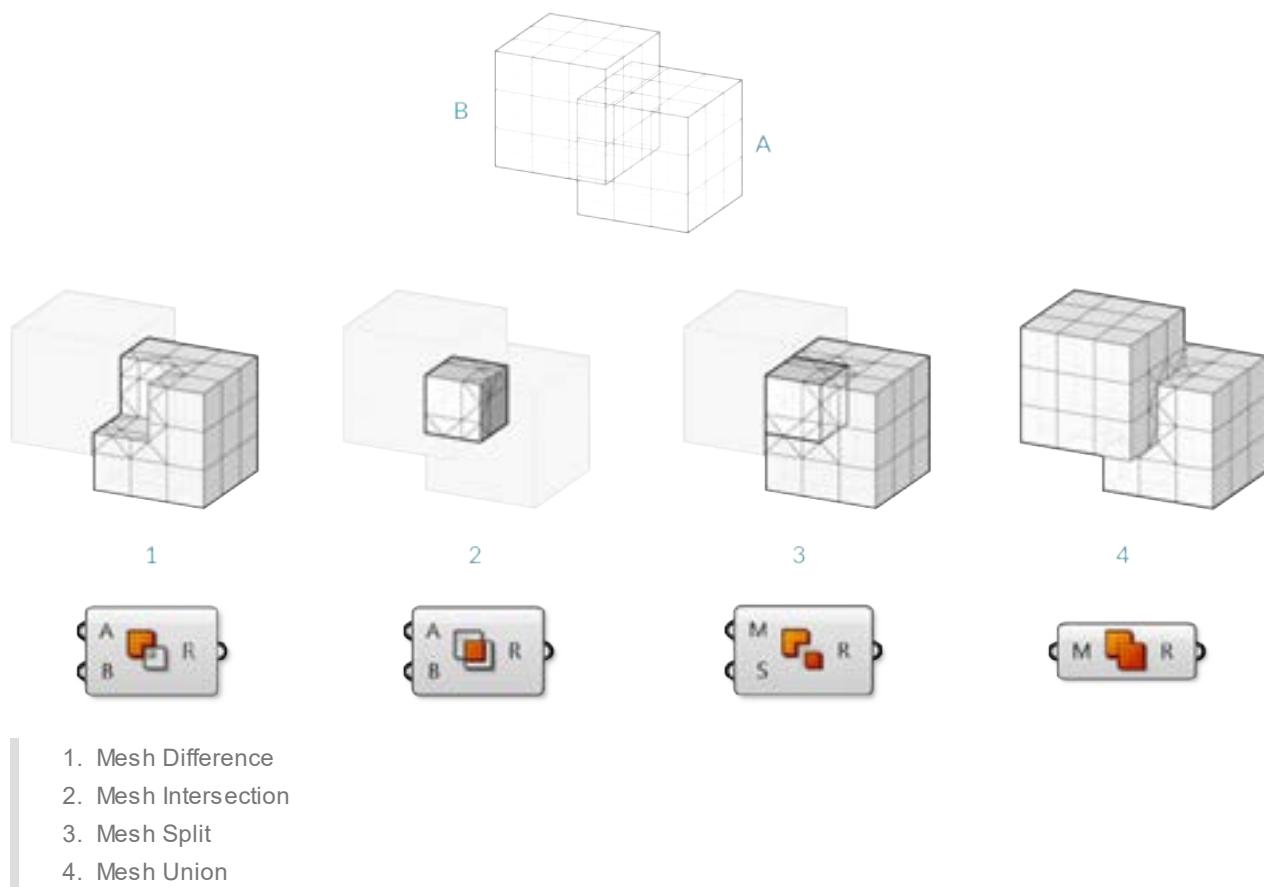


Abweichend von der Verbindung von NURBS Kurven und Oberflächen, die Berührungsstellen benötigen, können beliebige Polygonnetze miteinander verbunden werden - auch wenn die Polygonnetze sich nicht berühren. Es ist keine Voraussetzung, dass die Netzebenen miteinander verbunden sein müssen (obwohl in den meisten Anwendungen ein solches Polygonnetz nicht erstrebenswert ist !!!)

Diese Komponente verschweißt nicht die Eckpunkte und oft ist es sinnvoll sie mit der **Weld** Komponente zu kombinieren.

### Mesh Boolean

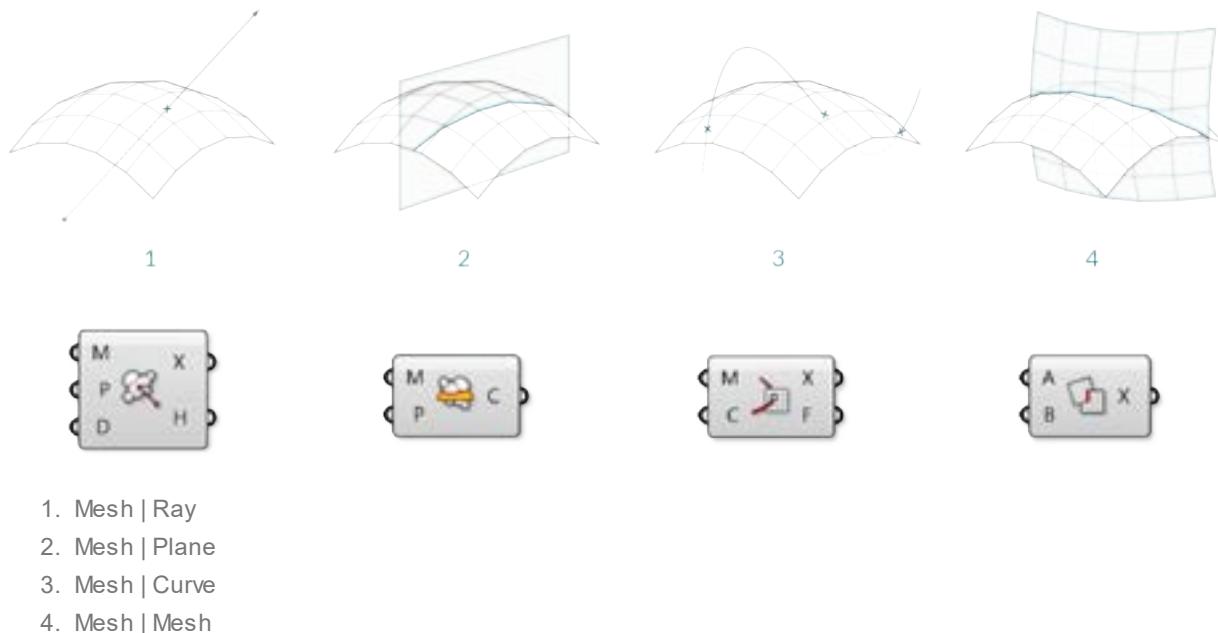
Polygonnetze in Grasshopper haben eine Anzahl von booleschen Operationen, ähnlich den booleschen Operationen für NURBS Körper. Boolesche Operationen sind abhängig von der Eingabereihenfolge, was bedeutet, dass wenn wir die Reihenfolge der Eingabepolygonnetze zwischen A und B umkehren, verschiedene Ergebnisse erzielt werden.



### 1.6.5.3 Verschneidung und Verschattung

#### Intersect

Verschneidungen können zwischen Polygonnetzen und anderen Objekten berechnet werden: Strahlen, Ebenen, Kurven und andere Polygonnetze

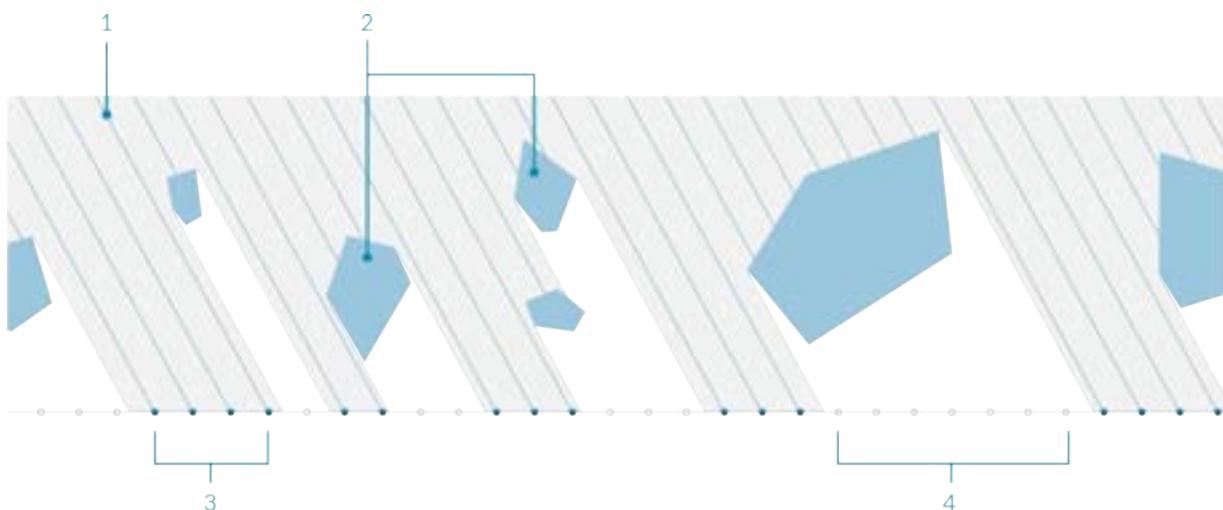


## Occlusion



Wie wir schon besprochen haben, ist eine der vielen Anwendungen von Polygonnetzgeometrien die Visualisierung und die Erstellung von Renderings basiert auf Flächennormalen. Wenn wir rendern ist es auch wichtig zu wissen, ob ein Objekt im Schatten hinter einem anderen Objekt liegt. Die **Occlusion** Komponente in Grasshopper erlaubt es uns einen Satz von Punkten als Stichprobe, das verschattende Polygonnetz und einen *Sichtstrahl* (einen Vektor, der die Lichtrichtung definiert) einzugeben.

Solch ein Prozess kann verwendet werden um Schatten in Renderings zu erzeugen oder um zu bestimmen ob Objekte von einem bestimmten Kamerawinkel aus verdeckt werden.



1. Sichtstrahl um auf Verschattung zu testen
2. Verschattendes Polygonnetz
3. 'Getroffene' Stichprobenpunkte
4. 'Verschattete' Stichprobenpunkte

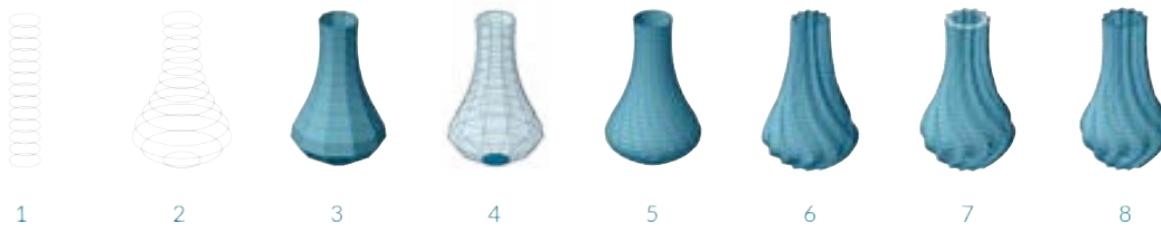


## 1.6.6 Arbeiten mit Polygonnetzgeometrien

In diesem Abschnitt werden wir uns durch eine Uebung arbeiten, die einen kompletten Polygonnetzkoerper erstellt. Am Ende der Uebung werden wir eine dynamische Definition fuer eine Vase entwickelt haben, die 3D gedruckt werden kann.

Beispieldateien fuer diesen Abschnitt: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

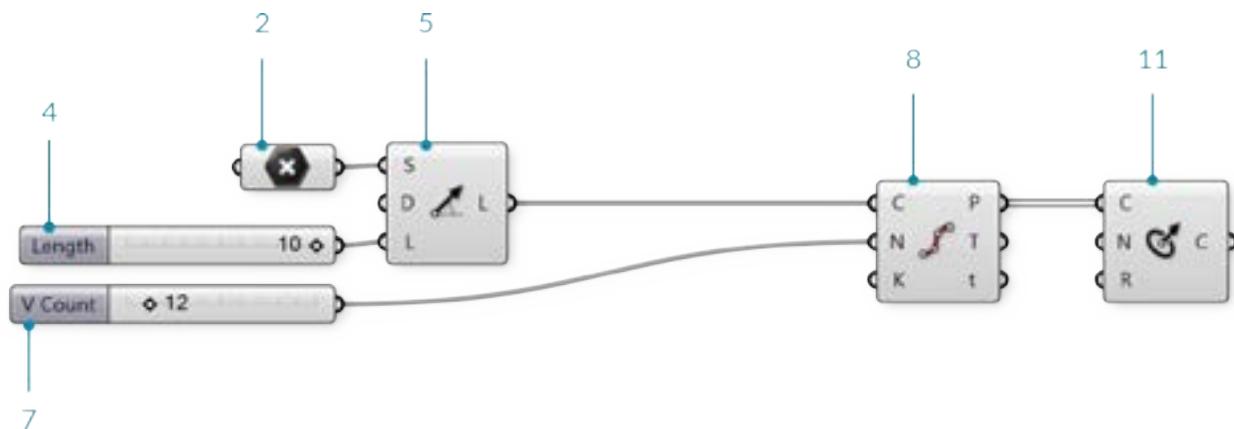
Da diese Definition etwas laenger ist als die vorausgegangenen Beispiele in diesem Primer, werden wir zuerst durch die grundlegenden Schritte gehen, die wir gehen werden:



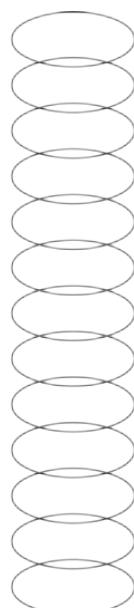
1. Erstelle eine Serie von Kreisen um einen Zylinder als Basisgeometrie zu erzeugen
2. Nutze die "Graph Mapper" Komponente um das Profil der Vase zu definieren
3. Konstruiere die Topology der Netzflaechen um ein einzelnes Polygonnetz zu erzeugen
4. Verschliesse den unteren Teil des Polygonnetzes mit einer Kappe
5. Fuehre eine Verdrehung entlang der Vertikalen ein um eine dynamischere Form zu erhalten
6. Fuehre Furchen als Textur der Vase ein
7. Versetze die Polygonnetzflaeche um der Vase Wandstaerke zu geben.
8. Verschliesse den oberen Spalt zwischen den beiden Polygonnetzen um einen geschlossenen Koerper zu erstellen

01.	Beginne eine neue Definition, druecke Strg+N (in Grasshopper)	
02.	<b>Params/Geometry/Point</b> - Ziehe einen <b>Point</b> Parameter auf die Leinwand	
03.	Referenziere einen Punkt in Rhino, indem Du auf die <b>Point</b> Komponente rechtsklickst und "Set one point" waehlst. Dies wird als Ursprungspunkt fuer die Vase dienen.	
04.	Du kannst einen Punkt manuell in Grasshopper erzeugen, indem Du auf die Leinwand doppelklickst um das Suchfenster zu oeffnen und dann die Koordinaten getrennt durch Kommas eingibst: '0,0,0' (ohne die Anfuehrungszeichen)	
05.	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand und setze folgende Werte: Name: Length Lower Limit: 1 Upper Limit: 10	
06.	<b>Curve/Primitive/Line SDL</b> - Ziehe eine <b>Line SDL</b> Komponente auf die Leinwand	
	Verbinde die <b>Point</b> Komponente mit dem Start (S) Eingabeparameter der <b>Line SDL</b> Komponente und verbinde den <b>Number Slider</b> mit dem Laenge Eingabeparameter.	
	Der Standardwert fuer die Richtung (D) der **Line SDL** Komponente iser der Z Einheitsvektor, den wir in diesem Beispiel nutzen werden.	

	<b>Params/Input/Number Slider</b> - Ziehe einen <b>Number Slider</b> auf die Leinwand und gebe die folgenden Werte ein: Name: VCount Rounding: Integer Lower Limit: 1 Upper Limit: 100	
07.		
08.	<b>Curve/Division/Divide Curve</b> - Ziehe eine <b>Divide Curve</b> Komponente auf die Leinwand	
09.	Verbinde den Linie (L) Ausgabeparameter der <b>Line SDL</b> Komponente mit dem Kurve (C) Eingabeparameter der <b>Divide Curve</b> Komponente	
10.	Verbinde den <b>V Count</b> Schieberegler mit dem Anzahl (N) Eingabeparameter der <b>Divide Curve</b> Komponente	
11.	<b>Curve/Primitive/Circle CNR</b> - Ziehe eine <b>Circle CNR</b> Komponente auf die Leinwand	
12.	Verbinde den Punkte (P) Ausgabeparameter der <b>Divide Curve</b> Komponente mit dem Zentrum (C) Eingabeparameter der <b>Circle CNR</b> Komponente	



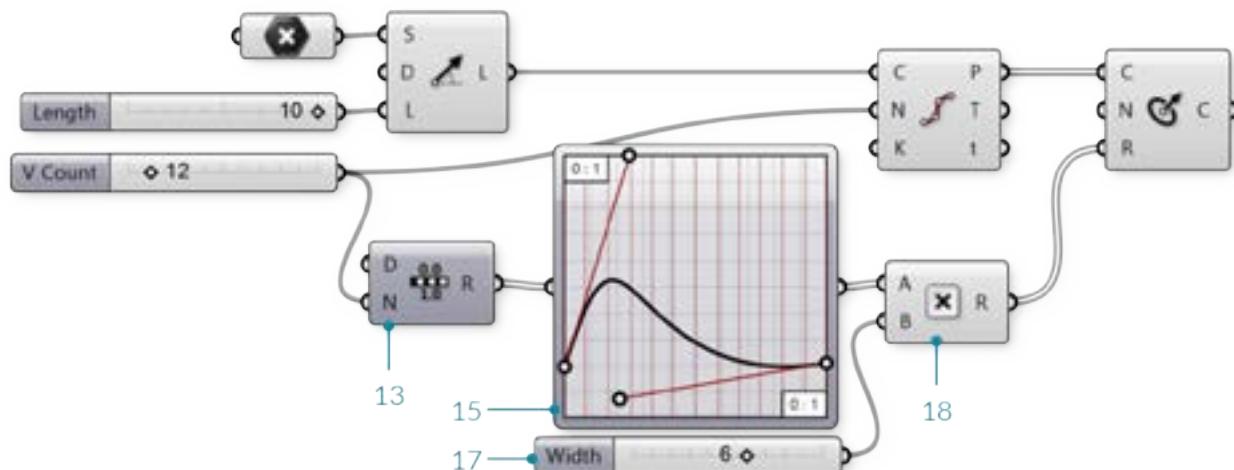
Wir haben eine Serie von Kreisen vertikal gestapelt. Wir werden diese als Profil fuer unsere Vase nutzen.



Als naechstes Werden wir die Radien der Kreise mit dem "Graph Mapper" steuern.

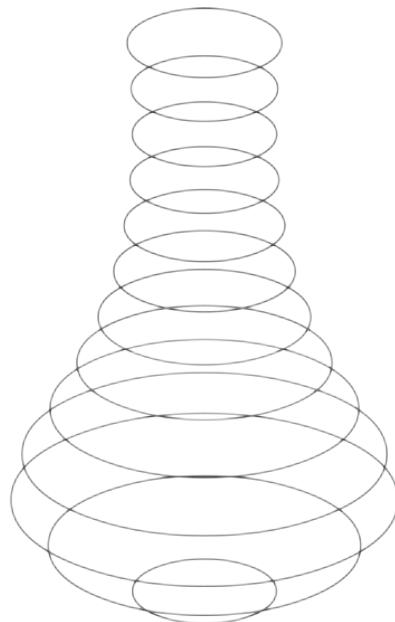
--	--	--

13.	Sets/Sequence/Range - Ziehe eine <b>Range</b> Komponente auf die Leinwand	
14.	Verbinde den <b>V Count</b> Schieberegler mit dem Schritte (N) Eingabeparameter der <b>Range</b> Komponente	
15.	<b>Params/Input/Graph Mapper</b> - Ziehe eine <b>Graph Mapper</b> Komponente auf die Leinwand	
16.	Rechtsklicke auf den <b>Graph Mapper</b> , klicke 'Graph Types' im Menu und waehle 'Bezier'	
17.	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand und setze die folgenden Werte: Name: Width Lower Limit: 0 Upper Limit: 10	
18.	<b>Maths/Operators/Multiplication</b> - Ziehe eine <b>Multiplication</b> Komponente auf die Leinwand	
19.	Verbinde den <b>Graph Mapper</b> des <b>Width</b> Schieberegler mit den A und B Eingabeparametern der <b>Multiplication</b> Komponente	
20.	Verbinde den Ergebnisse (R) Ausgabeparameter der <b>Multiplication</b> Komponente mit dem Radius (R) Eingabeparameter der <b>Circle CNR</b> Komponente	



Nutze die Griffe des **Graph Mapper** um die Profile der Kreise anzupassen.

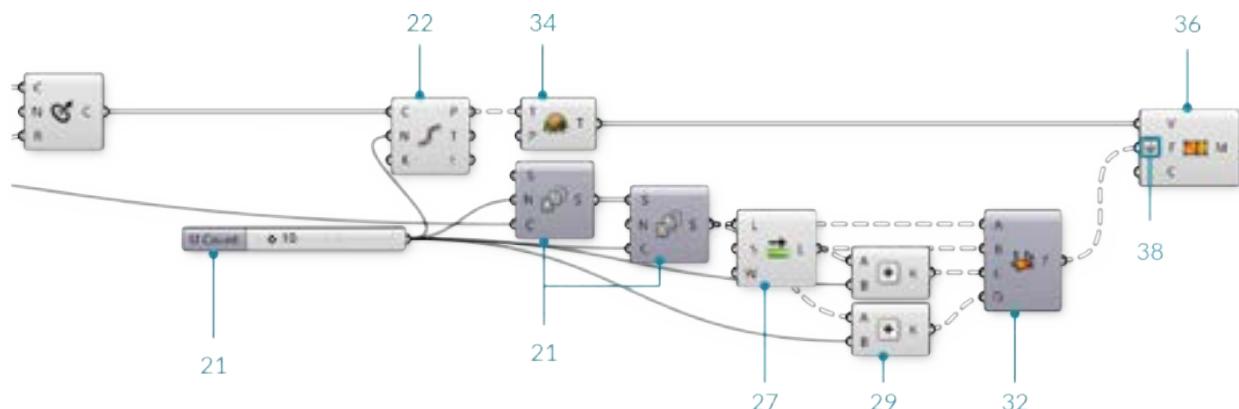
MERKE: Es ist wichtig sich zu versichern, dass der Startpunkt der Bezierkurve auf dem **Graph Mapper** ungleich 0 ist. Indem wir den Startpunkt des "Graph Mapper" grösser 0 wählen, stellen wir sicher, dass die Vase eine horizontale Standfläche hat.



Wie haben nun ein Profil fuer unsere Vase. Als naechstes werden wir die Polynetzflaeche erzeugen. Dies beinhaltet die Erzeugung der Eckpunkte und die Definition der Netzflaechen entsprechend der Indizes der Eckpunkte.

	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand: Name: U Count Rounding: Even Lower Limit: 2 Upper Limit: 100	
21.		
22.	<b>Curve/Division/Divide Curve</b> - Ziehe eine <b>Divide Curve</b> Komponente auf die Leinwand	
23.	Verbinde den Kreise (C) Ausgabeparameter der <b>Circle CNR</b> Komponente mit dem Kurve (C) Eingabeparameter der <b>Divide Curve</b> Komponente, und verbinde den <b>U Count</b> Schieberegler mit dem Anzahl (N) Eingabeparameter  Der Punkte (P) Ausgabeparameter dieser Komponente enthaelt die Eckpunkte, die wir zur Erstellung unseres Polygonnetzes benutzen	
24.	<b>Sets/Sequence/Series</b> - Ziehe zwei <b>Series</b> Komponenten auf die Leinwand	
25.	Verbinde den <b>U Count</b> Schieberegler mit dem Schritte (N) Eingabeparameter der ersten <b>Series</b> Komponente, und verbinde den <b>V Count</b> Schieberegler mit dem Anzahl(C) Eingabeparameter der <b>Series</b> Komponente	
26.	Verbinde den Serie (S) Ausgabaparameter mit der ersten <b>Series</b> Komponente mit dem Start (S) Eingabeparameter der zweiten <b>Series</b> Komponente, und verbinde den <b>U Count</b> Schieberegler mit dem Anzahl (C) Eingabeparameter	
27.	<b>Sets/List/Shift List</b> - Ziehe eine <b>Shift List</b> Komponente auf die Leinwand	
28.	Verbinde den Ausgabeparameter der <b>Series</b> Komponente mit der Liste (L) Eingabeparameter der <b>Shift List</b> Komponente	
29.	<b>Maths/Operators/Addition</b> - Ziehe zwei <b>Addition</b> Komponenten auf die Leinwand	
30.	Verbinde den Ausgabeparameter der zweiten <b>Series</b> Komponente des <b>U Count</b> Schieberegler mit den A und B Eingabeparameter der ersten <b>Addition</b> Komponente	
	Verbinde den Ausgabeparameter der <b>Shift List</b> Komponente des <b>U Count</b> Schiebereglers	

	mit den A und B Eingabeparameter der <b>Addition</b> Komponente	
32.	<b>Mesh/Primitive/Mesh Quad</b> - Ziehe eine <b>Mesh Quad</b> Komponente auf die Leinwand	
	Verbinde die folgenden Eingabeparameter der <b>Mesh Quad</b> Komponente: A - Second **Series** component B - **Shift List** C - Erste **Addition** Komponente D - Zweite **Addition** Komponente	
33.	<p>Wir haben nun eine Topologie fuer unser Polygonnetz zu Grunde gelegt. Diese Netzflaechen werden mit den Eckpunkten kombiniert. Die Ordnung dieser Verbindungen ist essentiell, also pruefe noch einmal alle Verbindungen bis zu diesem Punkt!</p>	
34.	<b>Sets/Tree/Flatten</b> - Ziehe eine <b>Flatten Tree</b> Komponente auf die Leinwand	
35.	Verbinde den Punkte (P) Ausgabeparameter der <b>Divide Curve</b> Komponente mit dem Baum (T) Eingabeparameter der <b>Flatten Tree</b> Komponente	
36.	<b>Mesh/Primitive/Construct Mesh</b> - Ziehe eine <b>Construct Mesh</b> Komponente auf die Leinwand	
37.	Verbinde den Baum (T) Ausgabeparameter der <b>Flatten Tree</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Construct Mesh</b> Komponente	
38.	Verbinde den Netzflaechen (F) Ausgabeparameter der <b>Mesh Quad</b> Komponente mit dem Netzflaechen (F) Eingabeparameter der <b>Construct Mesh</b> Komponente. Rechtklicke den F(Netzflaechen) Eingabeparameter und waehle 'Flatten'	



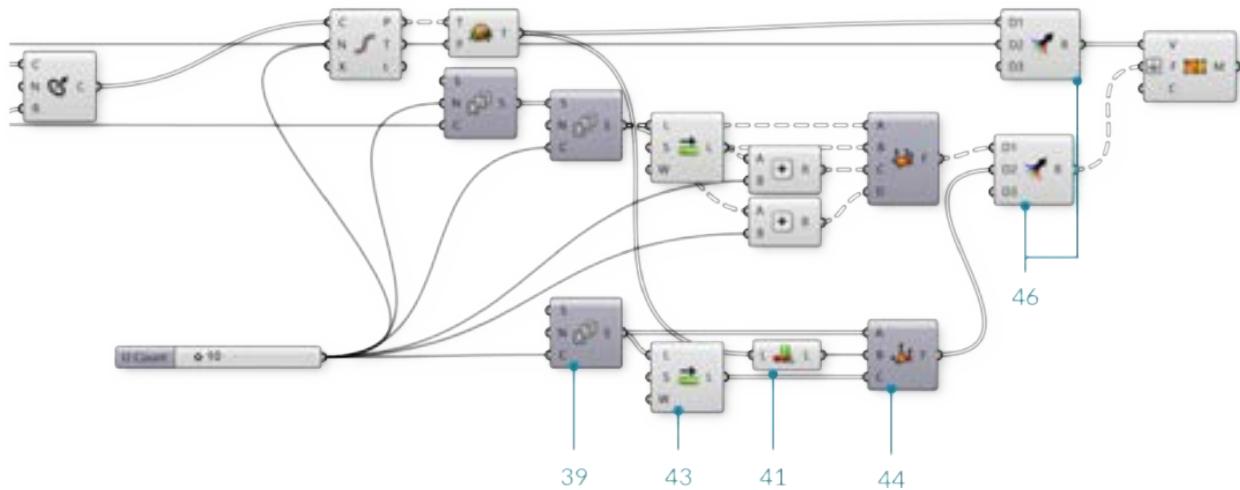
Wir haben nun die Polygonnetzflaechen fuer unsere Vase.



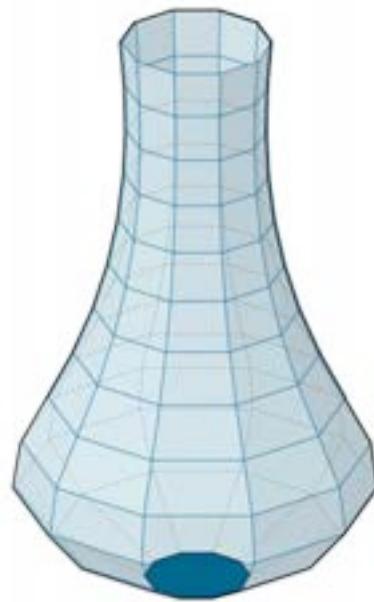
Als nächstes werden wir den Boden der Vase schliessen. Um dies zu tun, werden wir den ursprünglichen Ausgangspunkt zu unserer Liste von Eckpunkten hinzufügen, und dann dreieckige Netzflächen zu diesem Punkt von der unteren Kante der Vase.

39.	Sets/Sequence/Series - Ziehe eine <b>Series</b> Komponente auf die Leinwand	
40.	Verbinde den <b>U Count</b> Schieberegler mit dem Anzahl (C) Eingabeparameter der <b>Series</b> Komponente	
41.	Sets/List/List Length - Ziehe eine <b>List Length</b> Komponente auf die Leinwand	
	Verbinde den Baum (T) Ausgabeparameter der <b>Flatten Tree</b> Komponente mit dem Liste (L) Eingabeparameter der <b>List Length</b> Komponente	
42.	Dies wird der Index des Ursprungs sein, wenn wir ihn zu unserer Liste der Eckpunkte hinzufügt haben.	
43.	Sets/List/Shift List - Ziehe eine <b>Shift List</b> Komponente auf die Leinwand	
44.	Mesh/Primitive/Mesh Triangle - Ziehe eine <b>Mesh Triangle</b> Komponente auf die Leinwand	
45.	Verbinde die folgenden Eingabeparameter der <b>Mesh Triangle</b> Komponente: A - Neueste **Series** Komponente B - **List Length** C - **Shift List**	
46.	Sets/Tree/Merge - Ziehe zwei <b>Merge</b> Komponenten auf die Leinwand	
47.	Verbinde den Baum (T) Ausgabeparameter der <b>Flatten Tree</b> Komponente mit dem D1 Eingabeparameter, und verbinde die ursprüngliche <b>Point</b> Komponente mit dem D2 Eingabeparameter der ersten <b>Merge</b> Komponente	
48.	Verbinde den Netzflächen Ausgabeparameter (F) der <b>Mesh Quad</b> Komponente mit dem D1 Eingabeparameter, und verbinde den <b>Mesh Triangle</b> Ausgabeparameter mit dem D2 Eingabeparameter der zweiten <b>Merge</b> Komponente	
49.	Verbinde die erste <b>Merge</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Construct Mesh</b> Komponente, und verbinde die zweite <b>Merge</b> Komponente mit dem	

Netzflächen (F) Eingabeparameter der **Construct Mesh** Komponente.



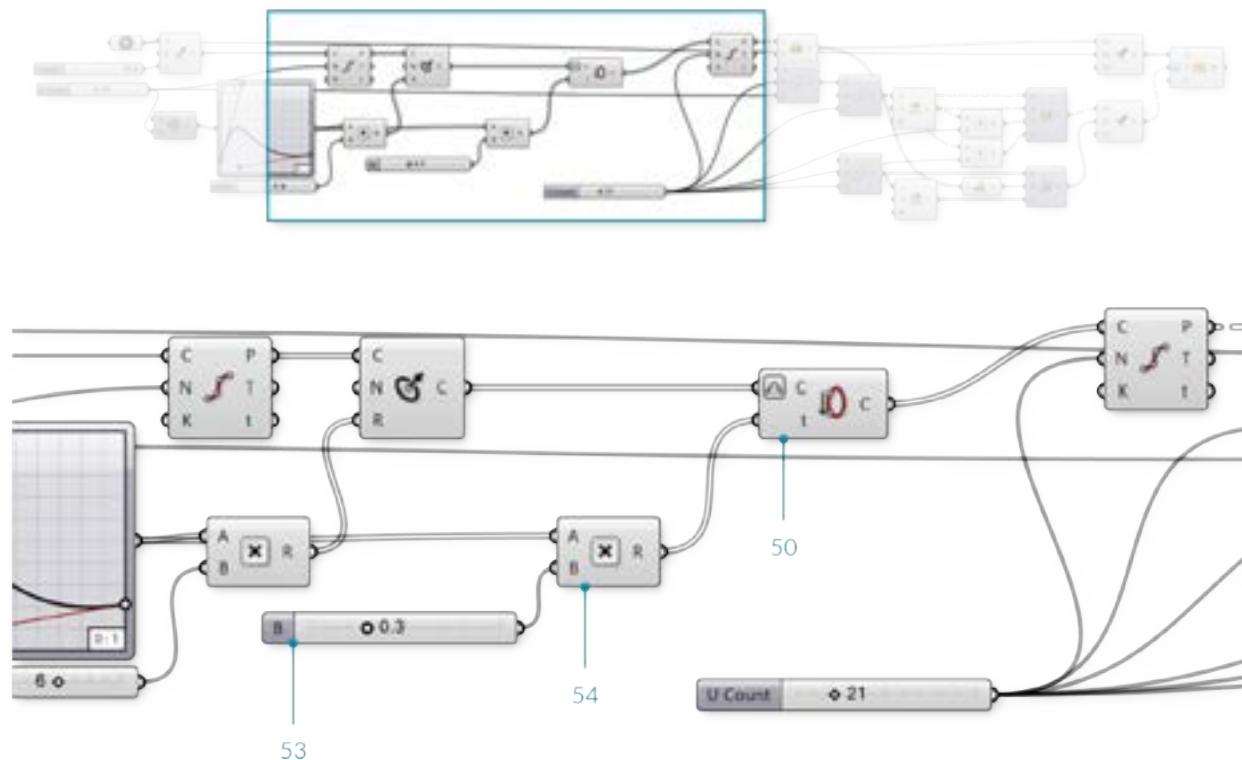
Wir haben den Boden der Vase mit dreieckigen Netzflächen verschlossen.



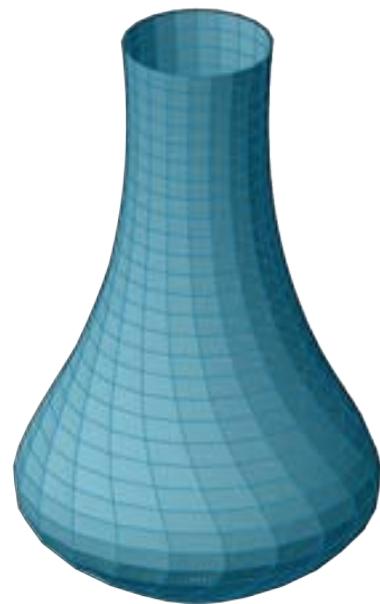
Wir werden jetzt etwas Detail zur Vase hinzufügen. Wir werden beginnen, indem wir der vertikalen Richtung eine Kurve geben, indem wir die Saeume der urspruenglichen Kreise anpassen

50.	<b>Curve/Util/Seam</b> - Ziehe eine <b>Seam</b> Komponente auf die Leinwand	
51.	Verbinde den Kreis (C) Ausgabeparameter der <b>Circle CNR</b> Komponente mit dem Kurve (C) Eingbaeparameter der <b>Seam</b> Komponente	
52.	Rechtsklicke den Kurve (C) Eingabeparameter der <b>Seam</b> Komponente und waehle'Reparameterize'	
53.	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand. Wir werden die Standardeinstellungen fuer diesen Schieberegler anwenden.	
54.	<b>Maths/Operator/Multiplication</b> - Ziehe eine <b>Multiplication</b> Komponente auf die Leinwand.	
55.	Verbinde den Ausgabeparameter der <b>Graph Mapper</b> mit dem AEingabeparameter, und den neuesten <b>Number Slider</b> mit dem BEingabeparameter der <b>Multiplication</b>	

	Komponente	
56.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Multiplication</b> Komponente mit dem Parameter (t) Eingabeparameter der <b>Seam</b> Komponente	



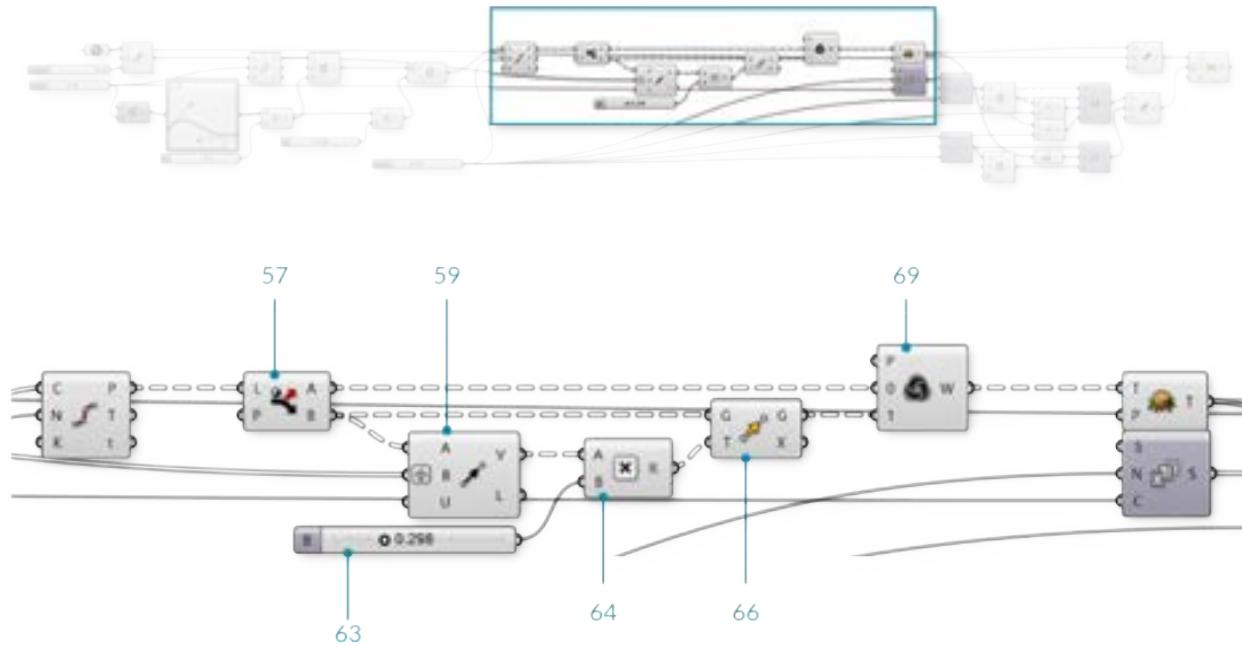
Die Kruemmung wird erreicht, indem wir die Saum Position der urspruenglichen Kreise anpassen, indem wir den selben "Graph Mapper" des Vasenprofils benutzen.



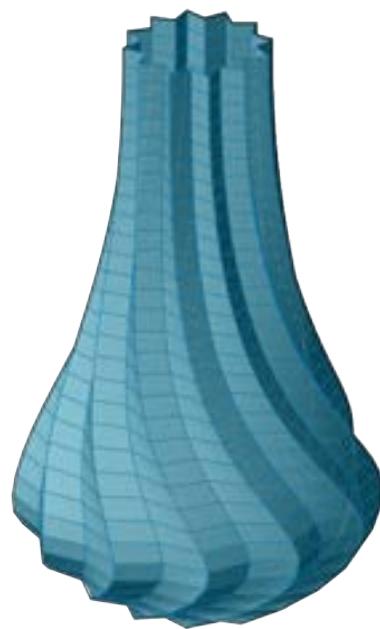
Als naechstes werden wir vertikale Furchen in die Vase einarbeiten.

57.	Sets/List/Dispatch - Ziehe eine <b>Dispatch</b> Komponente auf die Leinwand	
58.	Verbinde den Punkte (P) Ausgabeparameter der zweiten <b>Divide Curve</b> Komponente mit dem Liste (L) Eingabeparameter der <b>Dispatch</b> Komponente	

58.	Wir nutzen den Standardmuster (P) Eingabeparameter der **Dispatch** Komponente um die Punkte in zwei Listen in alternierendem Muster aufzuteilen.	
59.	<b>Vector/Vector/Vector 2Pt</b> - Ziehe eine <b>Vector 2Pt</b> Komponente auf die Leinwand	
60.	Verbinde den B Ausgabeparameter der <b>Dispatch</b> Komponente mit dem A Eingabeparameter der <b>Vector 2Pt</b> Komponente	
61.	Verbinde den Punkte (P) Ausgabeparameter der ersten <b>Divide Curve</b> Komponente mit dem B Eingabeparameter der <b>Vector 2Pt</b> Komponente	
62.	Rechtsklicke den B Eingabeparameter der <b>Vector 2Pt</b> Komponente und waehle 'Graft', und rechtsklicke den Vereinheitlichung (U) Eingabeparameter, gehe zu'Set Boolean' und waehle 'True'	
63.	Dies produziert einen Einheitsvektor fuer jeden Punkt, der auf das Zentrum der Kreise deutet	
64.	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand. Wir werden die Standardeinstellungen verwenden	
65.	<b>Maths/Operator/Multiplication</b> - Ziehe eine <b>Multiplication</b> Komponente auf die Leinwand	
66.	Verbinde den Vektor (V) Ausgabeparameter der <b>Vector 2Pt</b> Komponente mit dem A Eingabeparameter, und verbinde den <b>Number Slider</b> mit dem B Eingabe der <b>Multiplication</b> Komponente	
67.	<b>Transform/Euclidean/Move</b> - Ziehe eine <b>Move</b> Komponente auf die Leinwand	
68.	Verbinde den B Ausgabeparameter der <b>Dispatch</b> Komponente mit dem Geometrie (G) Eingabeparameter der <b>Move</b> Komponente	
69.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Multiplication</b> Komponente mit dem Richtungsvektor (T) Eingabeparameter der <b>Move</b> Komponente	
70.	<b>Sets/List/Weave</b> - Ziehe eine <b>Weave</b> Komponente auf die Leinwand	
71.	Verbinde den AAusgabeparameter der <b>Dispatch</b> Komponente mit dem 0 Eingabeparameter der <b>Weave</b> Komponente	
72.	Verbinde den Geometrie (G) Ausgabeparameter der <b>Move</b> Komponente mit dem 1 Eingabeparameter der <b>Weave</b> Komponente	
73.	Verbinde den Gewebe (W) Ausgabeparameter der <b>Weave</b> Komponente mit dem Baum (T) Eingabeparameter der <b>Flatten Tree</b> Komponente	



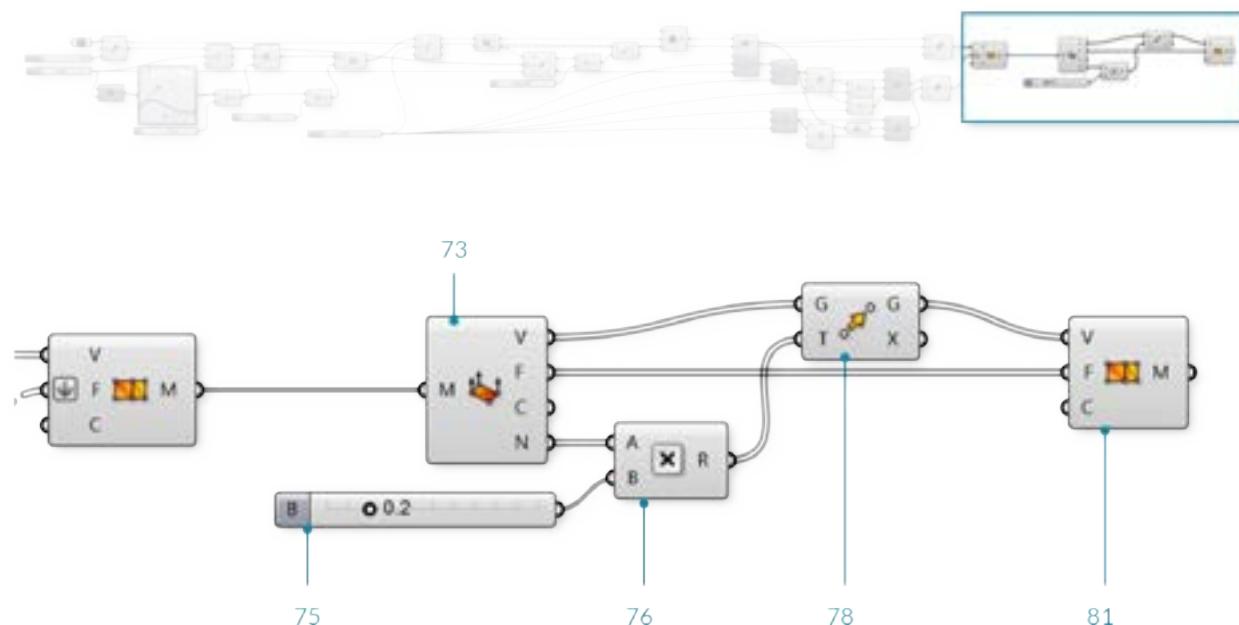
Erinnere Dich daran, zurueckzugehen und Deine Schieberegler und "Graph Mapper" anzupassen, um zu sehen, wie sich das Modell veraendert und um sicherzugehen, dass alles immer noch funktioniert. Dies ist bekannt als "Biegung" des Modells und soll regelmaessig gemacht werden um zu sehen ob Fehler in der Definition vorliegen.



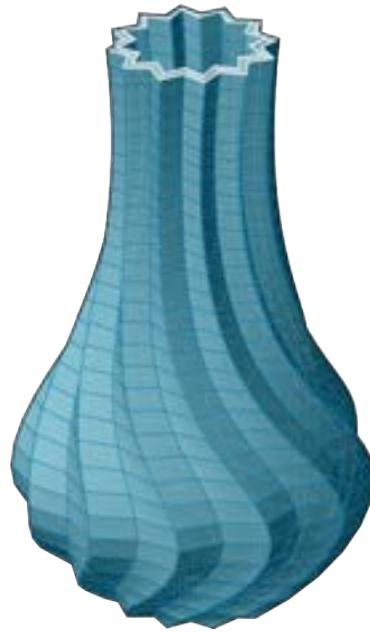
Wir haben jetzt eine einzelne Flaeche fuer unsere Vase. Wenn wir die Vase 3D drucken wollen, benoetigen wir einen geschlossenen Koerper. Wir werden den Koerper erzeugen, indem wir das Polygonnetz versetzen und die beiden Polygonnetze zu einem Koerper zusammenfuegen.

73.	<b>Mesh/Analysis/Deconstruct Mesh</b> - Ziehe eine <b>Deconstruct Mesh</b> Komponente auf die Leinwand	
74.	Verbinde den Polygonnetz (M) Ausgabeparameter der <b>Construct Mesh</b> Komponente mit dem Polygonnetz Eingabeparameter (M) der <b>Deconstruct Mesh</b> Komponente	
75.	<b>Params/Input/Number Slider</b> - Ziehe eine <b>Number Slider</b> Komponente auf die Leinwand. Wir werden die Standardeinstellungen verwenden	

76.	<b>Maths/Operator/Multiplication</b> - Ziehe eine <b>Multiplication</b> Komponente auf die Leinwand	
77.	Verbinde den Normalen (N) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem A Eingabeparameter, und verbinde den <b>Number Slider</b> mit dem B Eingabeparameter der <b>Multiplication</b> Komponente	
78.	<b>Transform/Euclidean/Move</b> - Ziehe eine <b>Move</b> Komponente auf die Leinwand	
79.	Verbinde den Eckpunkte (V) Eingabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem Geometrie (G) Eingabeparameter der <b>Move</b> Komponente	
80.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Multiplication</b> Komponente mit dem Richtungsvektor (T) Eingabeparameter der <b>Move</b> Komponente	
81.	<b>Mesh/Primitive/Construct Mesh</b> Ziehe eine <b>Construct Mesh</b> Komponente auf die Leinwand	
82.	Verbinde den Geometrie (G) Ausgabeparameter der <b>Move</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Construct Mesh</b> Komponente	
83.	Verbinde den Netzflächen (F) Ausgabeparameter der <b>Deconstruct Mesh</b> Komponente mit dem Netzflächen (F) Eingabeparameter der <b>Construct Mesh</b> Komponente	



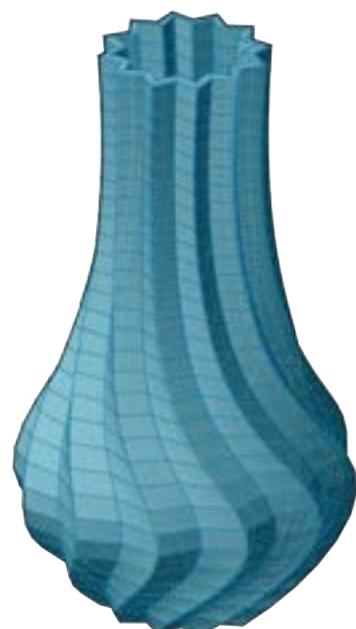
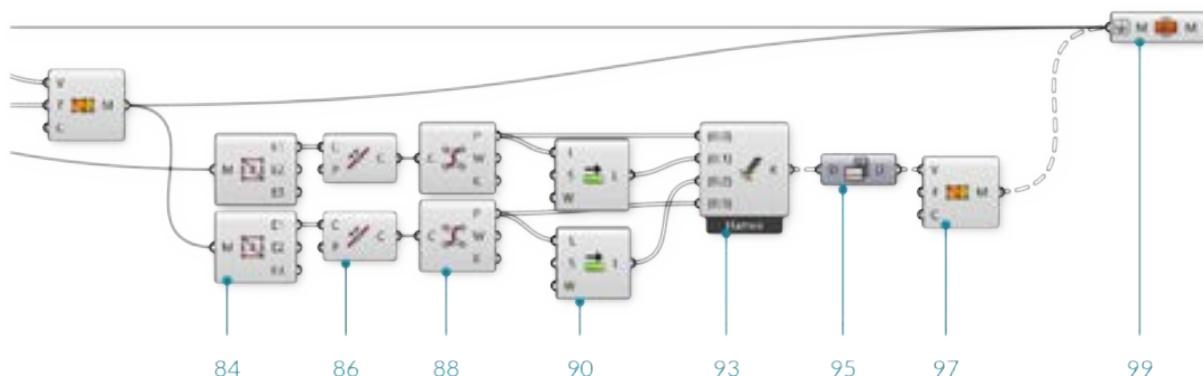
Durch das Versetzen des Polygonnetzes entlang der Eckpunktnormalen haben wir eine Innenseite und eine Aussenseite der Vase, aber immer noch einen Spalt zwischen den beiden Polygonnetzen

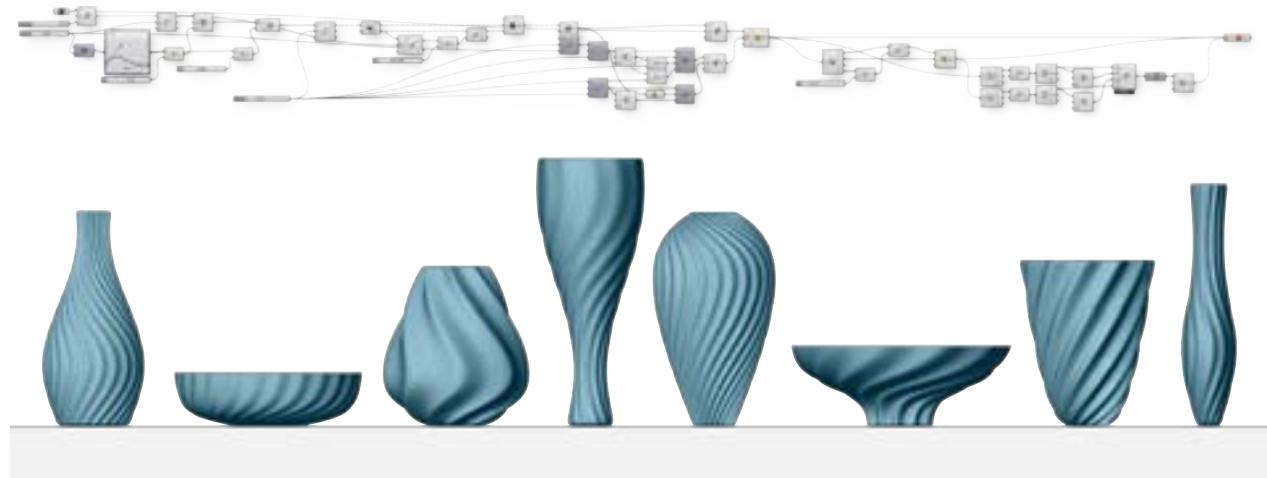


Der letzte Schritt wird es sein das Polygonnetz zu schliessen, indem wir ein neues Polygonnetz erzeugen um den Spalt zu schliessen und dann die Polygonnetze miteinander verbinden.

84.	<b>Mesh/Analysis/Mesh Edges</b> - Ziehe eine <b>Mesh Edges</b> Komponente auf die Leinwand	
85.	Verbinde den Polygonnetz (M) Ausgabeparameter der ersten <b>Construct Mesh</b> Komponente mit dem Polygonnetz (M) Eingabeparameter der <b>Mesh Edges</b> Komponente	
86.	<b>Curve/Util/Join Curves</b> - Ziehe eine <b>Join Curves</b> Komponente auf die Leinwand	
87.	Verbinde den offene Kanten (E1) Ausgabeparameter der <b>Mesh Edges</b> Komponente mit dem Kurven (C) Eingabeparameter der <b>Join Curves</b> Komponente	
88.	<b>Curve/Analysis/Control Points</b> - Ziehe eine <b>Control Points</b> Komponente auf die Leinwand	
	Verbinde den Kurven (C) Ausgabeparameter der <b>Join Curves</b> Komponente mit dem Kurven (C) Eingabeparameter der <b>Control Points</b> Komponente	
89.	Durch das verbinden der Kurven und das Extrahieren der Kontrollpunkte, sichern wir, dass die Ordnung der Punkte einlag des Rands der Vase konsistent ist, was wichtig fuer die Orientierbarkeit und Mannigfaltigkeit des finalen Polygonnetzes ist	
90.	<b>Sets/List/Shift List</b> - Ziehe eine <b>Shift List</b> Komponente auf die Leinwand	
91.	Verbinde den Punkte (P) Ausgabeparameter der <b>Control Points</b> Komponente mit dem Liste (L) Eingabeparameter der <b>Shift List</b> Komponente	
92.	Wiederhole Schritte 84 bis 91 fuer die zweite <b>Construct Mesh</b> Komponente	
93.	<b>Sets/Tree/Entwine</b> - Ziehe eine <b>Entwine</b> Komponente auf die Leinwand	
94.	Zoom auf die <b>Entwine</b> Komponente um die Optionen zu sehen und einen zusätzlichen Eingabeparameter zu erhalten. Wir werden vier Eingabeparameter brauchen. Verbinde sie auf die folgende Art: {0;0} - Punkte (P) der ersten **Control Points** Komponente {0;1} - Ausgabeparameter der ersten **Shift List** {0;2} - Ausgabeparameter der zweiten **Shift List** {0;3} - Punkte (P) der zweiten **Control Points** Komponente	

95.	Sets/Tree/Flip Matrix - Ziehe eine <b>Flip Matrix</b> Komponente auf die Leinwand	
96.	Verbinde den Ergebnis (R) Ausgabeparameter der <b>Entwine</b> Komponente mit dem Daten (D) Eingabeparameter der <b>Flip Matrix</b> Komponente	
97.	<b>Mesh/Primitive/Construct Mesh</b> - Ziehe eine <b>Construct Mesh</b> Komponente auf die Leinwand	
98.	Verbinde den Daten (D) Ausgabeparameter der <b>Flip Matrix</b> Komponente mit dem Eckpunkte (V) Eingabeparameter der <b>Construct Mesh</b> Komponente	
99.	<b>Mesh/Util/Mesh Join</b> - Ziehe eine <b>Mesh Join</b> Komponente auf die Leinwand	
100.	Verbinde alle drei <b>Construct Mesh</b> Komponenten mit dem Eingabeparameter der <b>Mesh Join</b> Komponente, indem Du Shift gedrueckt haeltst, waehren Du die Verbindungen erstellst (oder nutze eine <b>Merge</b> Komponente). Rechtsklicke den Polygonnetz (M) Eingabeparameter der <b>Mesh Join</b> Komponente und waehle 'Flatten'	





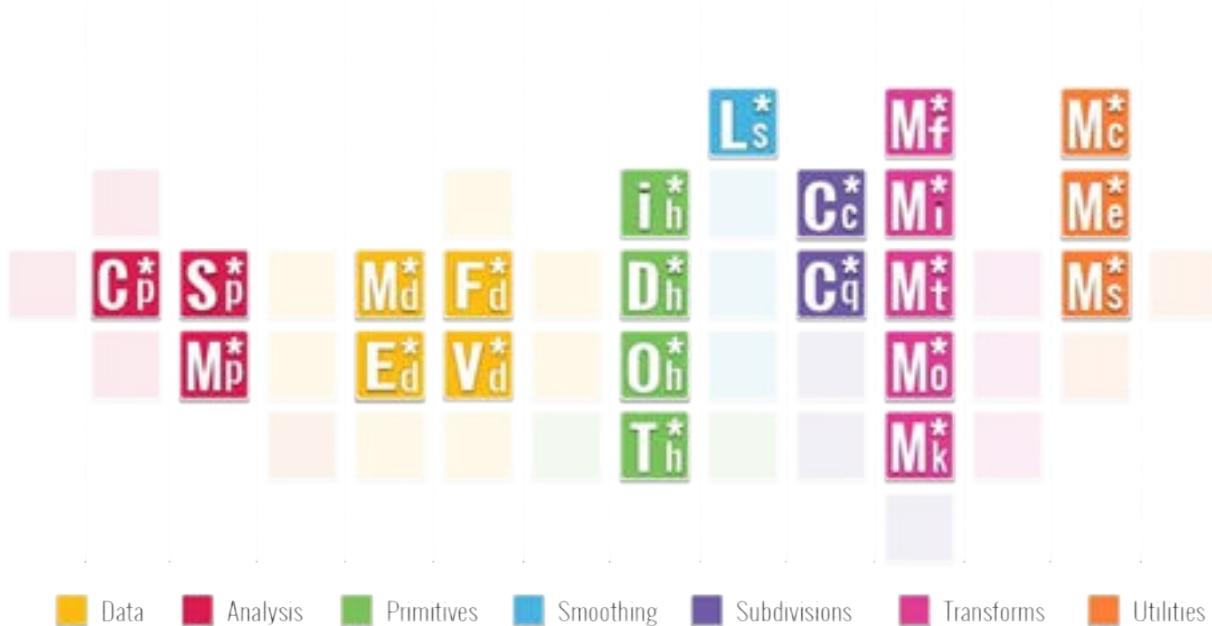
## 2. Erweiterungen

Fundamente sind gedacht um darauf zu bauen. Diese Ausgabe nennt eine Reihe von Schluesselplugins fuer Grasshopper, welche die Funktionalitaet der Anwendung erweitert *und* uns die Moeglichkeit gibt unsere Designs weiterzubringen.



## 2.1. Element\*

**Element\*** ist ein Polygonnetzgeometrie Plug-In fuer Grasshopper, das es ermoeglicht Polygonnetze zu erzeugen, analysieren, transformieren, unterteilen und glaetten. Element\* ermoeglicht Zugang zu Polygonnetztopologiedaten mit der Plankton Halb-Kantendatenstruktur fuer Polygonnetze.



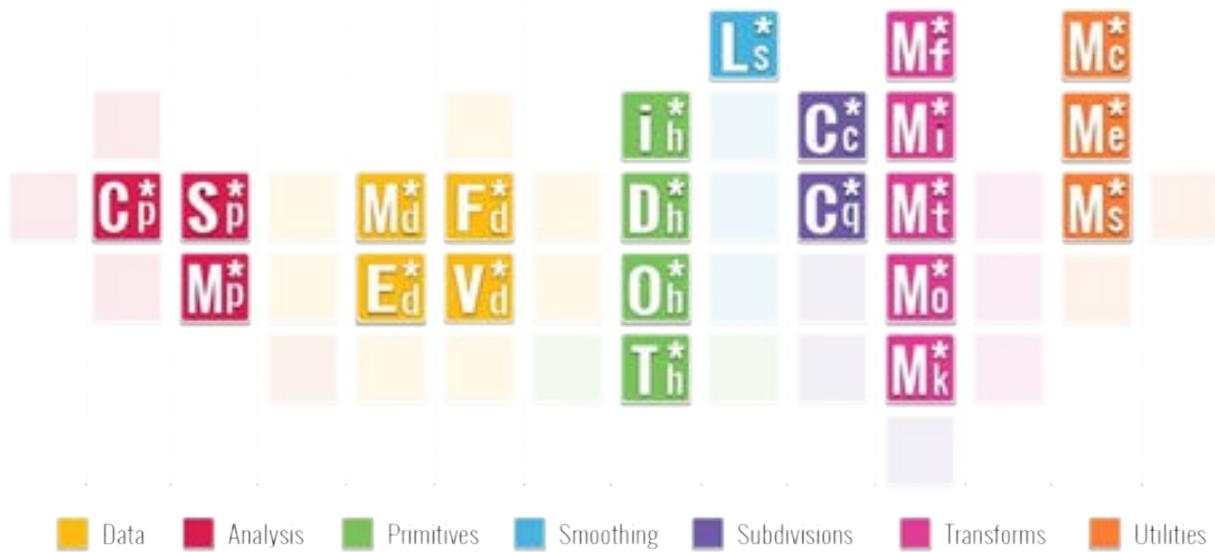
## 2.1.1. Element\*

Die Integration der Nutzung von Polygonnetze in Deinen Arbeitsablauf eröffnet eine grosse Bandbreite an Möglichkeiten Formen zu erzeugen, die von facettiert bis glatt reichen. Element\* erlaubt es Dir noch einen Schritt weiter zu gehen, indem es Dir einen eher intuitiven Zugang zur Analyse von Polygonnetztopologieanalyse und Glatteungs Routinen gibt. Dieses Kapitel ist eine Bedienungsanleitung für Element\* Version 1.1 und wird Dich auf den neuesten Stand bringen.

[Download Das Element\\* plug-in zum Start](#)

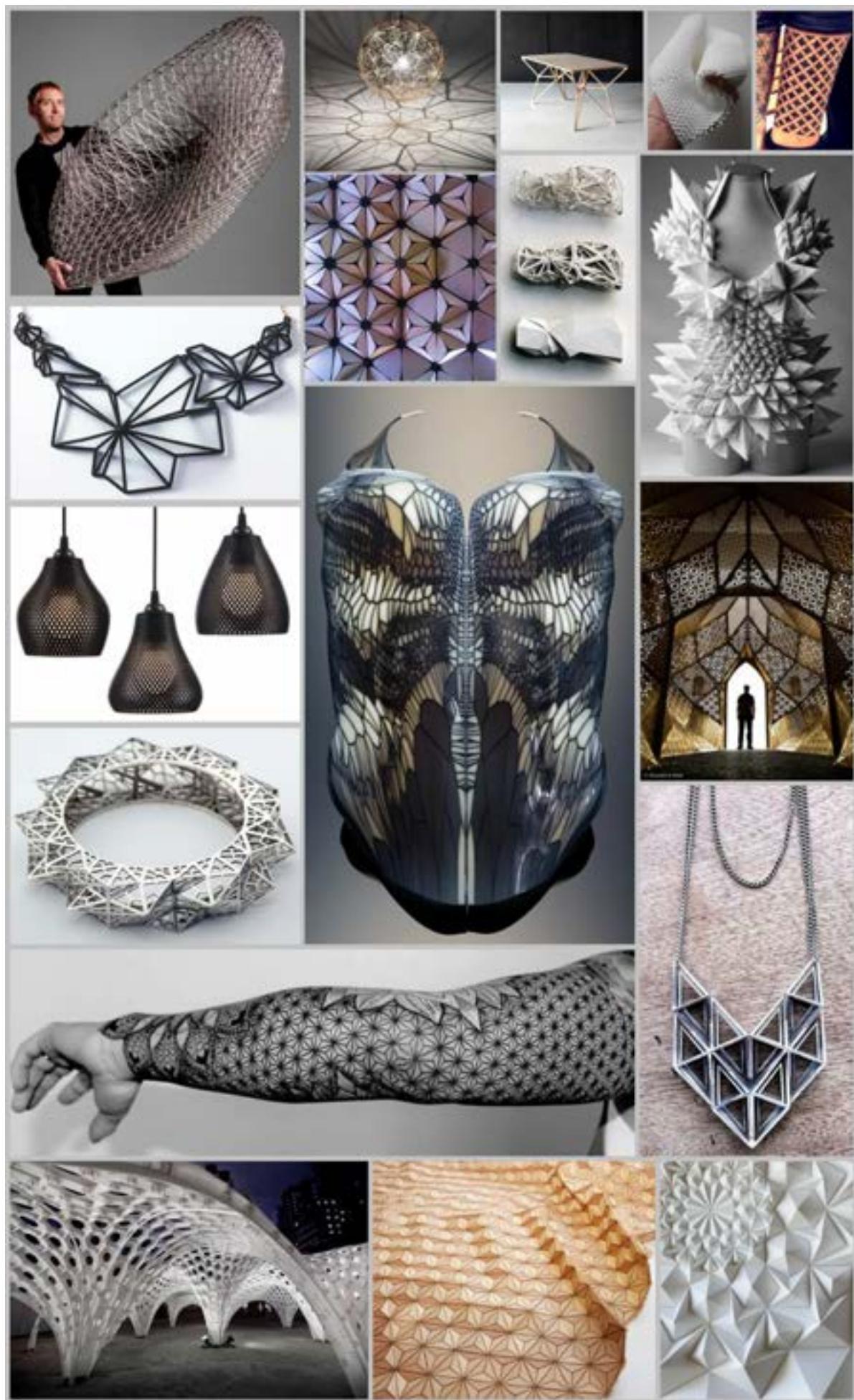


Element\* Komponenten sind basierend auf ihren Operationen kategorisiert. Wie ein Periodensystem, das eine Rahmenstruktur für die Analyse von chemischem Verhalten gibt, stellt Element einen Rahmen für die Analyse und Erkundung von Geometrie basierend auf Polygonnetzdaten und Operatoren. Wir stellen uns vor, dass neue Komponenten erstellt werden, indem die Zusammenhänge zwischen den Komponenten der verschiedenen Komponenten jeder Kategorie analysiert werden.



Unterhalb sind Inspirationsbilder der Arten von Produkten und Applikationen die mit

Element\* erzeugt werden koennen.

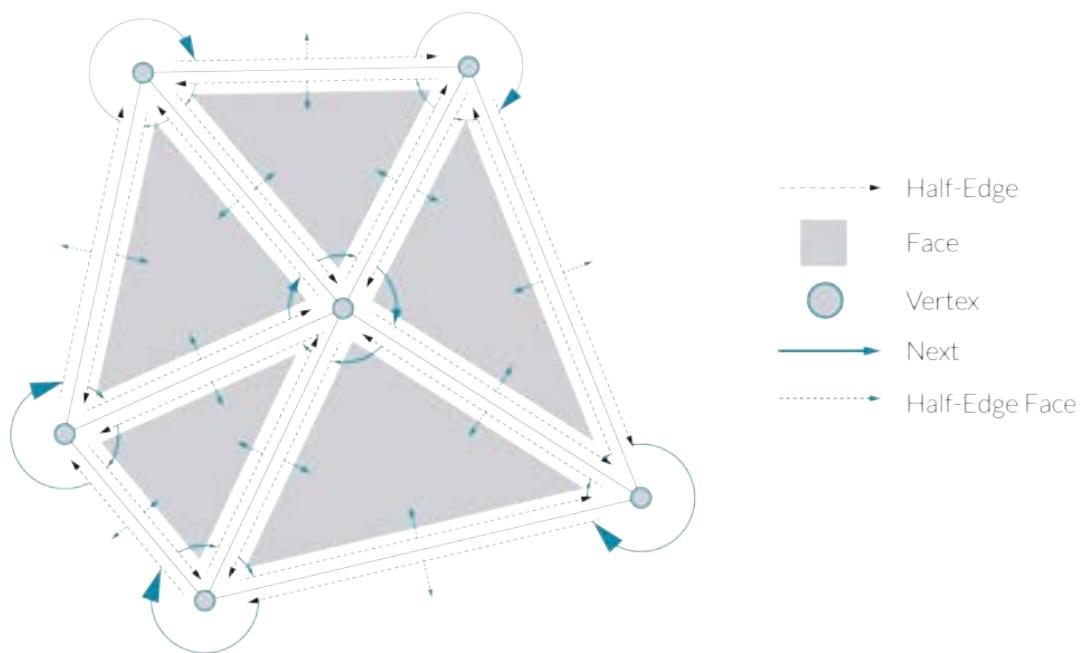




## 2.1.2. Halbkanten Daten

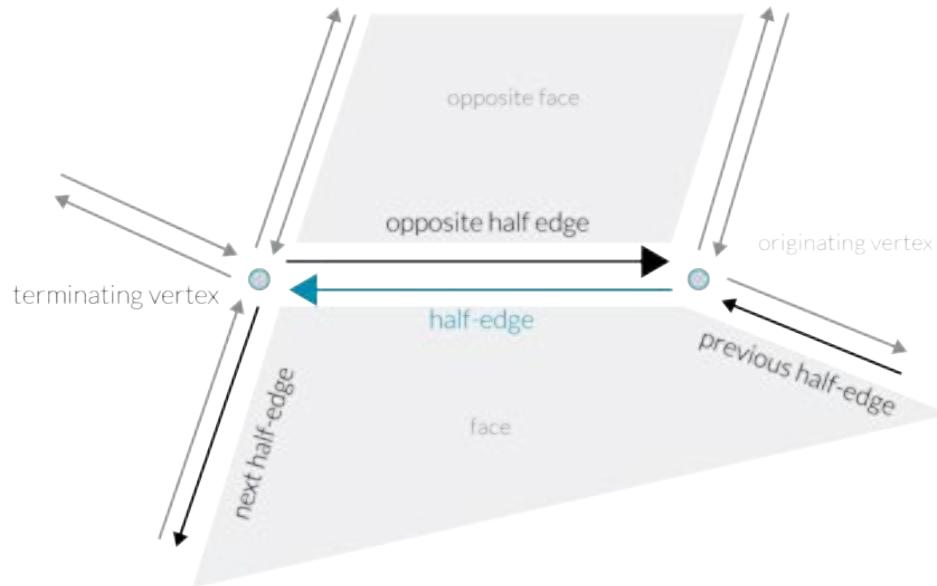
Im Grasshopper primer, haben wir uns angesehen wie ein Polygonnetz in Grasshopper mit der Netzflächen-Eckpunkte-Datenstruktur beschrieben werden kann. Diese ist eine relativ einfache Datenstruktur und wird für viele Polygonnetzapplikationen genutzt, kann aber bei komplexeren Algorithmen recht ineffizient werden. Die Element\* Erweiterung restrukturiert Polygonnetze mit der Halbkanten Datenstruktur, eine kantenfokussierte Datenstruktur, die effiziente Aufrufe von benachbarten Eckpunkten, Netzflächen und Kanten ermöglicht, was die Geschwindigkeit und Performance von Algorithmen stark erhöhen kann. Diese Struktur kann [...] Informationen über Eckpunkte, Netzflächen und Kanten erhalten. Diese Methode ermöglicht die Generierung von neuen Mustern und Geometrien, komplett basiert auf der topologischen Nachbarschaft der Basisgeometrie.

Die Halbkantendatenstruktur ist eine Repräsentation eines Polygonnetzes in dem jede Kante in zwei Halbkanten unterteilt ist, die in entgegengesetzte Richtungen zeigen. Dies erlaubt expliziten und impliziten Zugang zu Daten von einem Polygonnetzelement zu benachbarten Elementen.



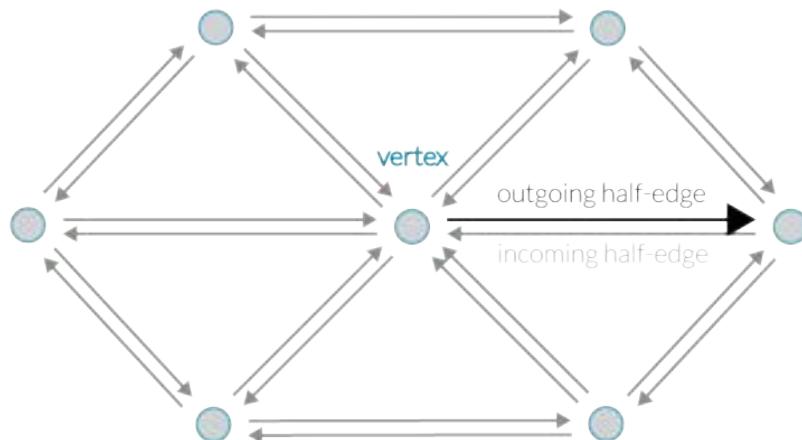
### 2.1.2.1 Halbkantenkonnektivität

Die in blau hervorgehobenen Halbkanten speichern Indizes mit Informationen über die jeweiligen Endpunkte, benachbarte Halbkanten und den zugehörigen Netzflächen. Die anderen Informationen (in grau dargestellt) sind implizit zugänglich.

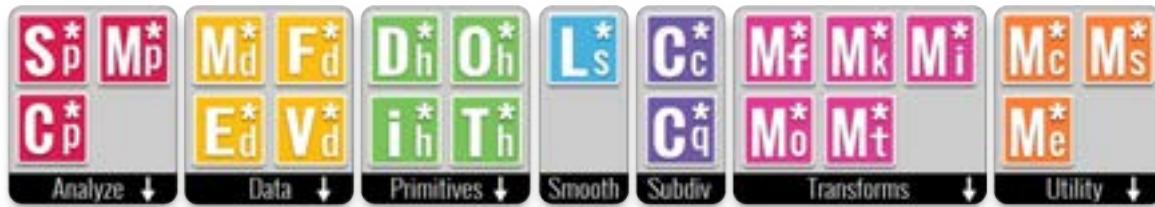


### 2.1.2.2 Eckpunktkonnektivitaet

Die in blau hervorgehobenen Eckpunkte speichern einen Index zu den jeweiligen ausgehenden Halbkanten. Die anderen Informationen (in grau dargestellt) sind implizit zugaenglich.



## 2.1.3. Element\* Komponenten



### 2.1.3.1 Analyse



1



2



3

1. Mesh Closest Point
2. Mesh Evaluate
3. Mesh Sample Plus

#### Element\* Mesh Closest Point

Abweichend von Grasshopper's **Mesh Closest Point** Komponente, berechnet diese Komponente auch den Normalenvektor und die Farbe der ausgegebenen Punkte und ueberkommt die Notwendigkeit der **Mesh Eval** Komponente, was der Graphen auf der Leinwand vereinfacht.

#### Element\* Mesh Evaluate

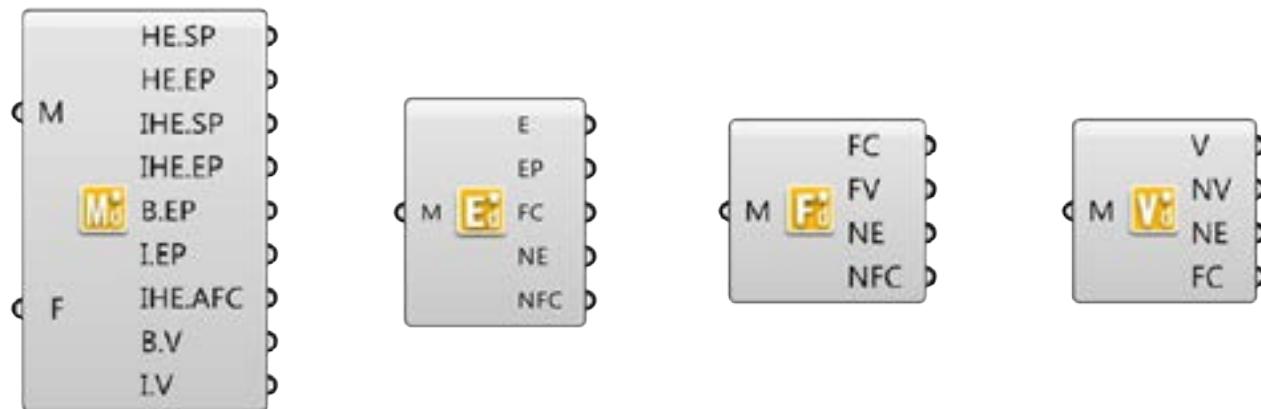
Die Grasshopper eigene **Mesh Eval** Komponente benoetigt einen Polygonnetzparameter als Eingabe, der von der **Mesh Closest Point** Komponente ausgegeben wird, aber recht schwierig haendisch zu erstellen ist. Element's "Closest Point" Komponente ermoeglicht die direkte Eingabe des Index einer Netzflaeche und ihrer baryzentrischen Koordinaten.

Merke - baryzentrische Koordinaten sind auf eine Art und weise definiert, dass sie immer zu eins aufaddieren. Wenn die Eingabewerte U, V und W nicht auf eins aufaddieren, wird diese Komponente das Verhaeltnis der drei Werte beibehalten, waehrend sie normalisiert werden. Zum Beispiel, wenn Du die Eingabewerte 2, 2 und 4 eingibst, wird der Polygonnetzparameter mit {0.25;0.25;0.5} berechnet.

#### Element\* Mesh Sample Plus

Diese Komponente wird benutzt, um schnell Farbinformation aus einem Polygonnetz zu extrahieren. Sie gibt Alpha, Rot, Gruen, Blau, Farbtton, Saettigung, und Leuchtkraftwerte der eingegebenen Punkte aus. Wenn ein gegebener Punkt nicht auf dem Polygonnetz liegt, wird diese Komponente den naehesten Punkt auf dem Polygonnetz berechnen. Diese Komponente nutzt parallele Datenverarbeitung um die Geschwindigkeit zu erhoehen.

## 2.1.3.2 Daten



1

2

3

4

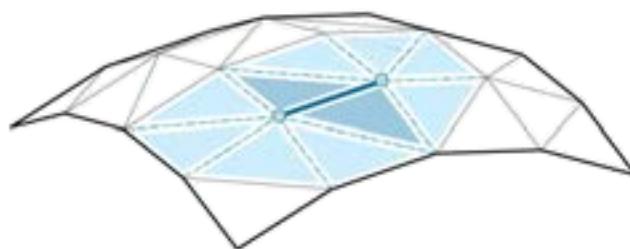
1. Datenvisualisierer
2. Benachbarte Kanten
3. Benachbarte Netzflaechen
4. Benachbarte Eckpunkte

### Element\* Data Visualizer

Diese Komponente wird verwendet um Halbkantendaten der Netzflaechen eines Eingabepolygonnetzes zu visualieren.

### Element\* Edge Neighbors

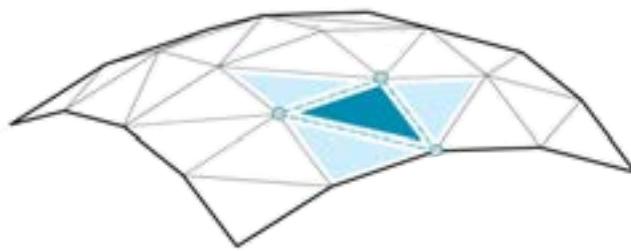
Diese Komponente bietet die Moeglichkeit Daten zur Nachbarschaft von Kanten eines Eingabepolygonnetzes strukturiert auszugeben. Die Ausgabedaten werden als Baum mit einem Ast fuer jede Kante in dem Polygonnetz ausgegeben. Sie gibt die Meshkanten, deren Entpunkte und die Mittelpunkte der Netzflaechen angrenzend an jeder Kante ("dual graph"), die benachbarten Kanten als Linienobjekte (in Reihe entgegen dem Uhrzeigersinn) und benachbarte Netzflaechenmittelpunkte (in Bezug auf die Kanten Start- und Endpunkte) aus.



**Edge Neighbors** - Kanten, Eckpunkte an den Enden, angrenzende Netzflaechenmittelpunkte, benachbarte Kanten und benachbarte Netzflaechenmittelpunkte

### Element\* Face Neighbors

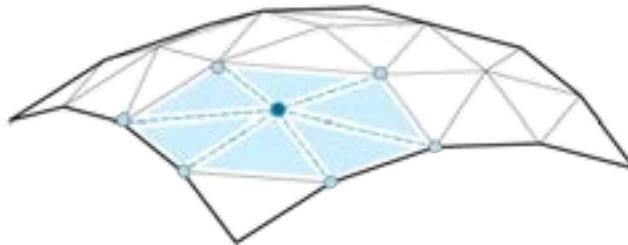
Diese Komponente ist aehnlich zu den anderen in diesem Abschnitt, aber die Daten im Baum sind entsprechende der Netzflaechen geordnet, mit einem Ast pro Netzflaeche. Die Ausgabeparameter sind die Netzflaechenmittelpunkte, die Eckpunkte jeder Netzflaeche (angeordnet entgegen dem Uhrzeigersinn), benachbarte Kanten (angeordnet entgegen dem Uhrzeigersinn) und die Mittelpunkte der benachbarten Flaechen (angeordnet entgegen dem Uhrzeigersinn).



**Face Neighbors** - Netzflächenmittelpunkte, Netzflächeneneckpunkte, benachbarte Kanten und benachbarte Netzflächenmittelpunkte

#### Element\* Vertex Neighbors

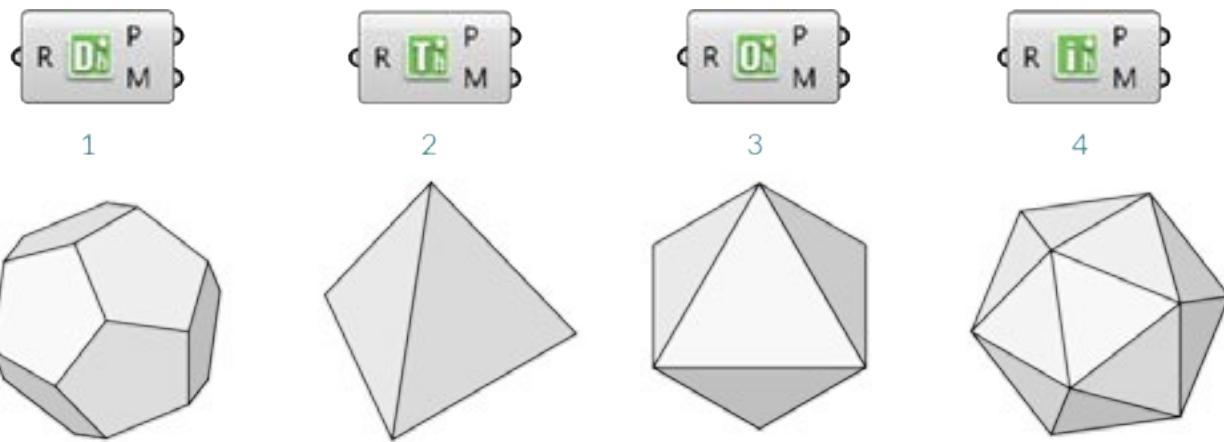
Diese Komponente gibt die Polygonnetzeckpunkte, benachbarte Eckpunkte (angeordnet entgegen dem Uhrzeigersinn), benachbarte Kanten (angeordnet entgegen dem Uhrzeigersinn) und benachbarte Netzflächenmittelpunkte (angeordnet entgegen dem Uhrzeigersinn) in einem strukturierten Datenbaum entsprechend den Eckpunkten des Polygonnetzes aus.



**Vertex Neighbors** - Eckpunkte, benachbarte Eckpunkte, benachbarte Kanten, benachbarte Netzflächenmittelpunkte

### 2.1.3.3 Primitive Polygonnetzkörper

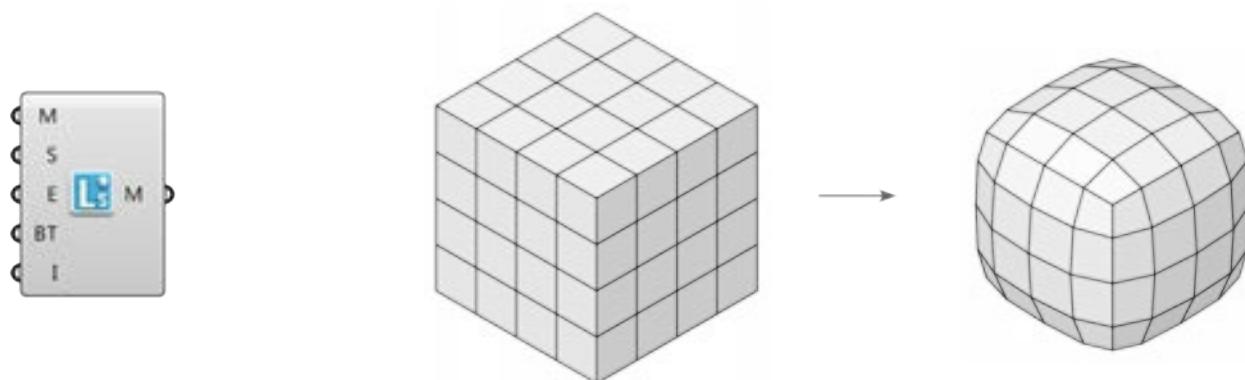
Element\* stellt vier zusätzliche primitive Polygonnetzkörper zur Verfügung: den Dodekaeder, Tetraeder, Oktaeder und Ikosaeder. Diese Komponente nimmt eine einzelne Zahl als Eingabeparameter für den Radius und produziert ein Polygonnetz mit dem Koordinatenursprung als Zentrum, das aus einer Netzfläche pro Seite besteht. Mit dem zusätzlichen Einsatz des Würfels, der bereits in Grasshopper zur Verfügung steht, macht das fünf platonische Körper.



1. Dodekaeder
2. Tetraeder
3. Oktaeder
4. Ikosaeder

### 2.1.3.4 Glaettung

**Element\* Smooth** stellt einen optimierten Glaettungsalgorithmus zur Verfuegung, der effizienter arbeitet als Grasshopper's **Smooth Mesh**, wenn er fuer groessere Datensaetze eingesetzt wird. Er nutzt den Laplaceschen Glaettungsalgorithmus fuer halbkantenstrukturierte Polygonnetze. Er veraendert nicht die Topologie oder die Anzahl der Eckpunkte eines verschweissten Polygonnetzes, wird aber identische Eckpunkte zusammenfuehren, die aus einem unverschweissten Polygonnetz resultieren. Wir koennen die Glaettungsstaerke, Grenbedingungen, Grenztoleranzen, sowie die Anzahl der Iterationen angeben.



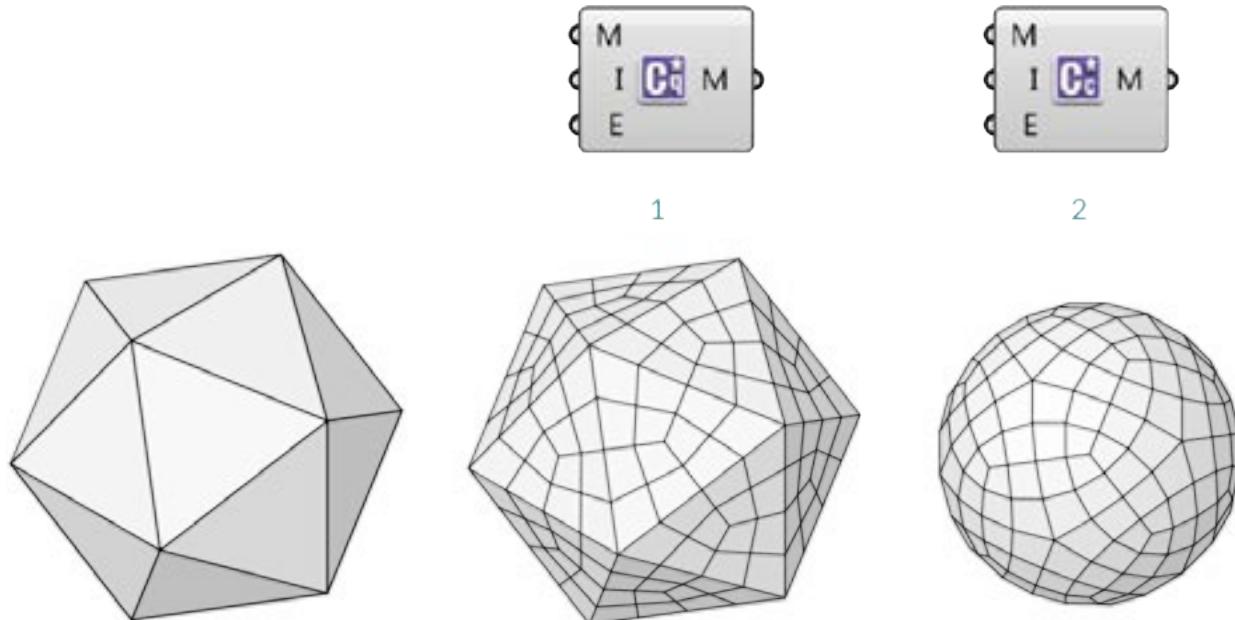
### 2.1.3.5 Unterteilung

#### Element\* Catmull Clark Subdivision

Dies ist eine rekursive Unterteilung basierend auf dem Catmull Clark Algorithmus. Wir koennen die Anzahl der Iterationen, sowie die Art des Umgangs mit offenen Kanten bestimmen.

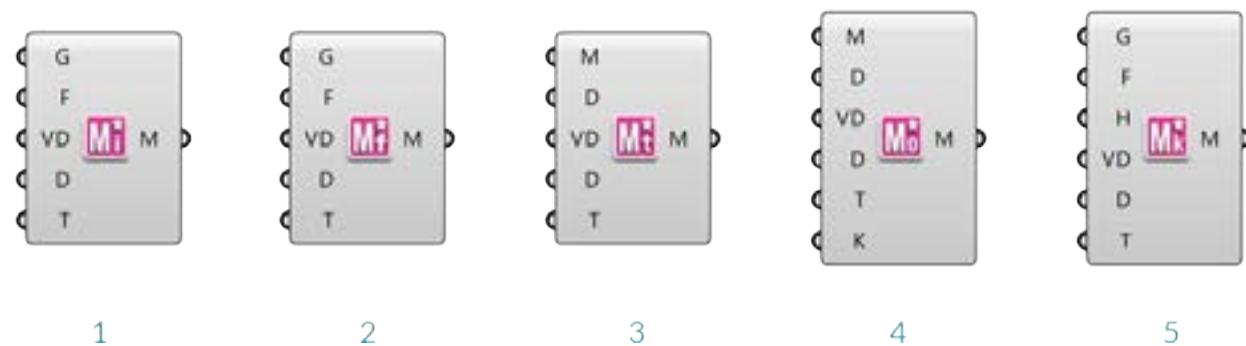
#### Element\* Constant Quad

Diese Unterteilungskomponente wird ein Polygonnetz mit ausschliesslich viereckigen Netzflaechen erzeugen indem sie eine weitere Netzflaeche fuer jede Kante des Polygonnetzes hinzufuegt.



1. Constant Quad Unterteilung
2. Catmull Clark Unterteilung

### 2.1.3.6 Transformation



1. Mesh Window
2. Mesh Frame
3. Mesh Thicken
4. Mesh Offset
5. Mesh Poke Face

Diese Komponenten stellen eine Anzahl von unterschiedlicher Transformationen zur Verfuegung, die unten beschrieben werden. Jede Komponente hat die zusaetzliche Moeglichkeit Distanzdaten pro Eckpunkt anzunehmen um eine Variation der Transformationsstaerke ueber das Polygonnetz zu erreichen.

#### Element\* Mesh Window

Stellt ein neues Polygonnetz innerhalb der Netzaechen mit einem Versatzwert her. Diese Komponente akzeptiert entweder ein Polygonnetz oder eine Liste von geschlossenen Polylinien als Eingabe.

#### Element\* Mesh Frame

Die Ausgabe ist ein Rahmen um die Netzaechen. Jede resultierende Netzaeche wird ein neues Loch im Zentrum haben. Diese Komponente akzeptiert entweder ein Polygonnetz oder eine Liste von geschlossenen

Polylinien als Eingabe.

#### **Element\* Mesh Thicken**

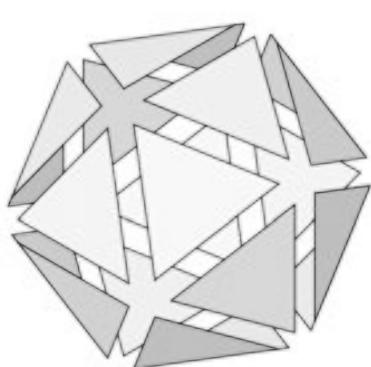
Diese Komponente wird einem Eingabepolygonnetz Materialstaerke entsprechend der Eckpunktnormalen entsprechend der zur Verfuegung gestellten Eingabewerte zuweisen.

#### **Element\* Mesh Offset**

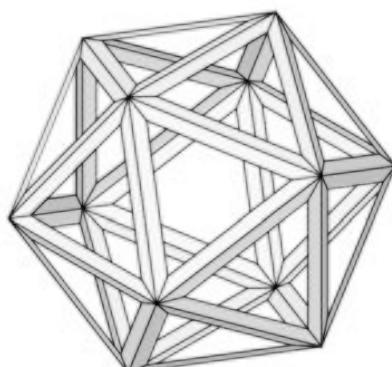
Diese Komponente erstellt einen Versatz zum urspruenglichen Polygonnetz entsprechend der Eckpunktnormalen.

#### **Element\* Mesh Poke Face**

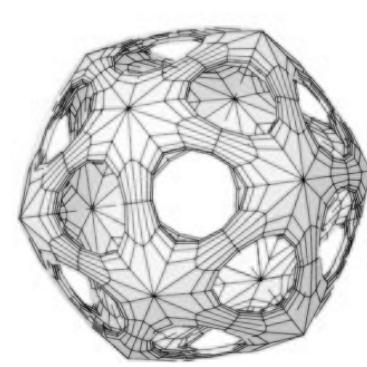
Zuerst werden die Netzflaechen mit einer "Frame" Operation geteilt und dass die Innenflaeche nochmals mittig unterteilt. Fuer diesen Teil der Netzflaeche kann ein Wert angegeben werden, der die innenliegenden Netzflaechen entlang der Flaechennormalen verschiebt. Zum Beispiel wir ein vierseitiges Polygon ("Quad") in vier dreiseitige Polygone, mit einem gemeinsamen Eckpunkt in deren Mitte, unterteilt. Der Hoeheneingabeparameter erlaubt es den Eckpunkt entsprechend zu transformieren.



1



2



3

1. Mesh Window
2. Mesh Frame
3. Ikosaeder nach der aufeinanderfolgenden Anwendung der "Mesh Frame", "Mesh Thick" und Polygonnetzteilung Komponenten.

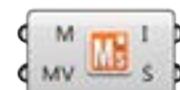
#### **2.1.3.7 Dienstprogramme**



1



2



3

1. Mesh Combine & Clean
2. Mesh Edges

### 3. Mesh Status

#### **Element\* Mesh Combine and Clean**

Diese Komponente verbindet verschiedene Polygonnetze mit den zusätzlichen Optionen das Polygonnetz basierend auf einem Winkel zu verschweissen oder identische Eckpunkte zu verbinden. Diese Komponente stellt ebenso mögliche topologische Probleme fest und gibt diese als Anmerkungen und Warnungen mit detaillierten Beschreibungen aus. Im Fall, dass die Zusammenführung von identischen Eckpunkten eine schlechte Topologie erzeugt, wird die Komponente die unveränderten Eingabepolygonnetze ausgeben, anstatt sie zusammenzuführen. Der Nutzer kann auch angeben das Polygonnetz zusammenzuführen ohne die Eckpunkte zu verändern.

#### **Element\* Mesh Edges**

Diese Komponente gibt die offenen Kanten, Kante, Netzflächenpolylinien und für unverschweißte Polygonnetze die unverschweißten Polygonnetzkanten aus.

#### **Element\* Mesh Status**

Diese Komponente gibt Polygonnetzinformationen basierend auf der Topologie aus. Es gibt zwei verschiedene Modi in denen wir die Informationen ansehen können. Die erste ist die Ausgabe von Daten zur Geometrie, wie Polygonnetzvalidität, Anzahl der Eckpunkte, Anzahl der Netzflächen und Anzahl der normalen. Die andere ist die Ausgabe des Polygonnetzstatus, welcher den Zustand des Polygonnetzes beschreibt, ob es nicht-mannigfaltige Kanten, degenerierte Netzflächen, die Anzahl der offenen Kanten oder die Anzahl unverbundener Netzflächen ist. Diese Komponente operiert nicht auf einem Polygonnetz, es gibt lediglich Informationen für den Nutzer aus. Es gibt auch eine Option für die Kombination identischer Eckpunkte, wodurch der Nutzer den Effekt dieser Option auf die entsprechenden Daten beobachten kann.

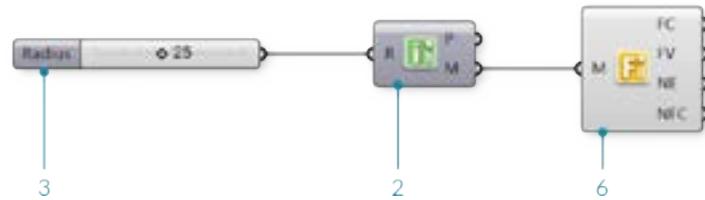
## 2.1.4. Exercise

In this section, we will work through a simple exercise using the Element\* primitives as a base. We will incorporate the half-edge data structure as well using both features of the transform components (uniform and per vertex)



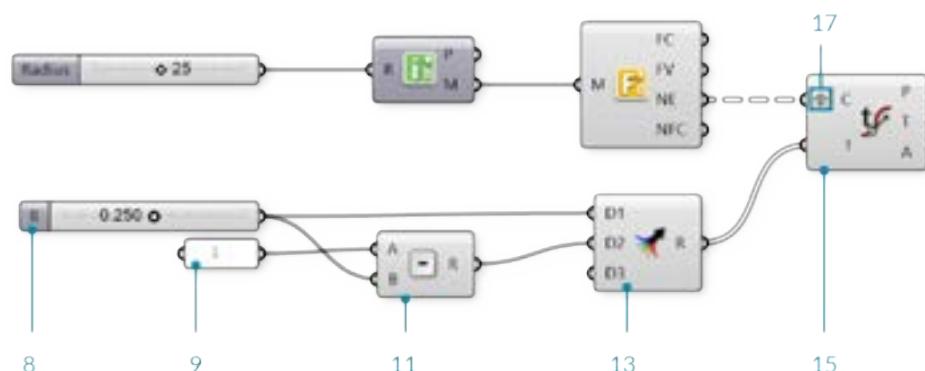
Example files that accompany this section: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

01.	Start a new definition, type Ctrl-N (in Grasshopper)	
02.	<b>Element*/Primitive/Icosohedron</b> - Drag and drop the <b>Icosohedron</b> component onto the canvas	
03.	<b>Params/Input/Number Slider</b> - Drag and drop the <b>Number Slider</b> component onto the canvas	
04.	Connect the <b>Number Slider</b> to the Radius (R) input of the <b>Icosohedron</b> component	
05.	Double-click the <b>Number Slider</b> and set appropriate values. For this example, we used: Name: Radius Rounding: Integer Lower Limit: 5 Upper Limit: 50 Value: 25	
06.	<b>Element*/Data/Face Neighbors</b> - Drag and drop the <b>Face Neighbors</b> component onto the canvas	
07.	Connect the Mesh (M) output of the <b>Icosohedron</b> component to the Mesh (M) input of the <b>Face Neighbors</b> component.	



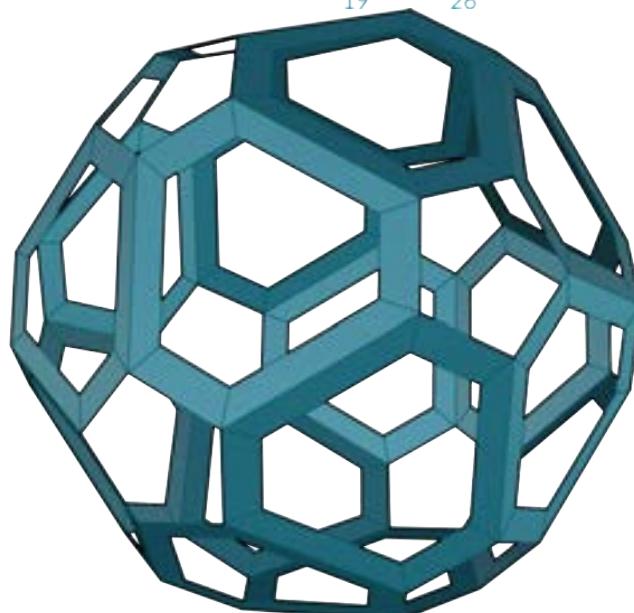
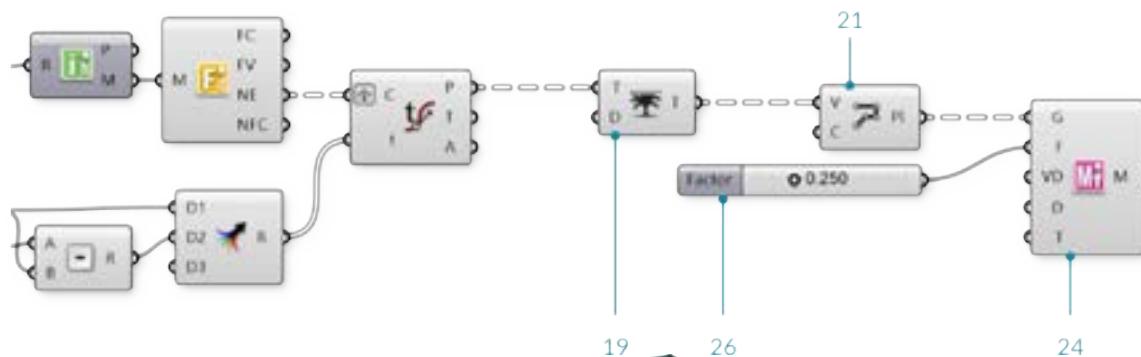
Looking at the data of the Neighboring Face Edges (NE) output, we see that we have a tree with 20 branches, where each branch contains three lines. The 20 branches each represent a face of the icosohedron which has 20 sides, while the three lines are the edges of each triangular face.

	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas and set the following values: Rounding: Float Lower Limit:0 Upper Limit: 0.5	
08.	<b>Params/Input/Panel</b> - Drag and drop a <b>Panel</b> component onto the canvas	
09.	Double-click the <b>Panel</b> component and enter "1" into the text-field	
10.	<b>Math/Operators/Subtraction</b> - Drag and drop a <b>Subtraction</b> component onto the canvas	
11.	Connect the <b>Panel</b> with a value of "1" into the Ainput and connect the number slider to the Binput of the <b>Subtraction</b> component	
12.	<b>Sets/Tree/Merge</b> - Drag and drop a <b>Merge</b> component onto the canvas	
13.	Connect the <b>Number Slider</b> to the D1 input of <b>Merge</b> , and connect the output R of the <b>Subtraction</b> component to the D2 input of <b>Merge</b>	
14.	<b>Curve/Analysis/Evaluate Curve</b> - Drag and drop an <b>Evaluate Curve</b> component onto the canvas	
15.	Connect the Face Edges (NE) output of the <b>Face Neighbors</b> component to the Curve (C) input of the <b>Evaluate Curve</b> component	
16.	Right click the Curve (C) input of the <b>Evaluate Curve</b> component and select Graft. This will create a new branch for each edge.	
17.	Connect the Result (R) output of the <b>Merge</b> component to the Parameter (t) input of the <b>Evaluate Curve</b> component. Because we grafted the Curve input, each edge is evaluated at both parameters from <b>Merge</b>	

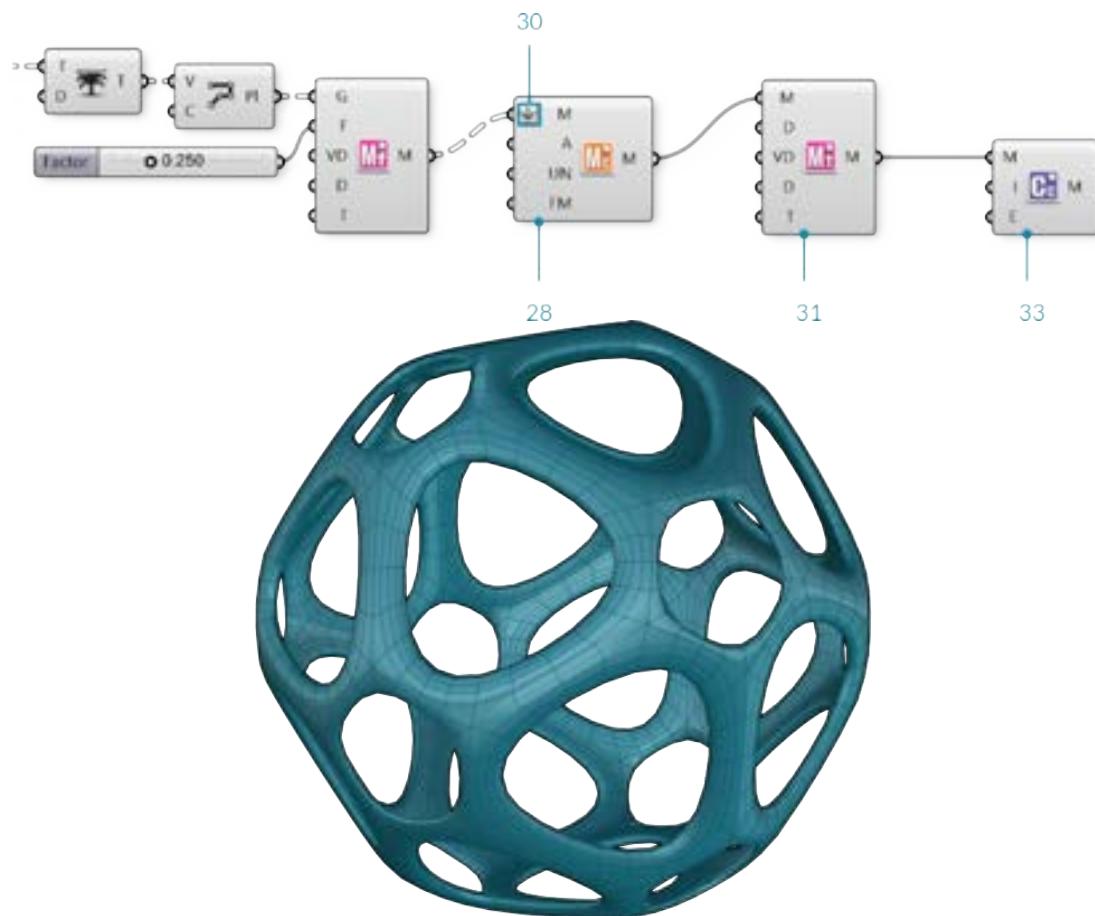


18.	<b>Sets/Tree/Trim Tree</b> - Drag and drop a <b>Trim Tree</b> component onto the canvas	
	Connect the Points (P) output of <b>Evaluate Curve</b> to the Tree (T) input of the <b>Trim Tree</b>	

	component.	
20.	The default value of Depth (D) input for **Trim Tree** is 1. This reduce the depth of our data tree one level by merging the outer most branch. The result is 20 branches, each with six points.	
21.	<b>Curve/Spline/Polyline</b> - Drag and drop a <b>Polyline</b> component onto the canvas	
22.	Connect the Tree (T) output of the <b>Trim Tree</b> component to the Vertices (V) input of the <b>Polyline</b> component	
23.	Right click the Closed (C) input of the <b>Polyline</b> component, click "Set Boolean" and set the value to True This has created a closed polyline of six sides for each original face of the mesh.	
24.	<b>Element*/Transform/Mesh Frame</b> - Drag and drop a <b>Mesh Frame</b> component onto the canvas.	
25.	Connect the Polyline (PI) output of the <b>Polyline</b> component to the Geometry (G) input of the <b>Mesh Frame</b> component Note that the **Mesh Frame** component can take either meshes or a list of closed polyline curves as input	
26.	<b>Params/Input/Number Slider</b> - Drage and drop a <b>Number Slider</b> component onto the canvas. We will keep the default range of 0 to 1 for this slider	
27.	Connect the <b>Number Slider</b> to the Factor (F) input of the <b>Mesh Frame</b> component	



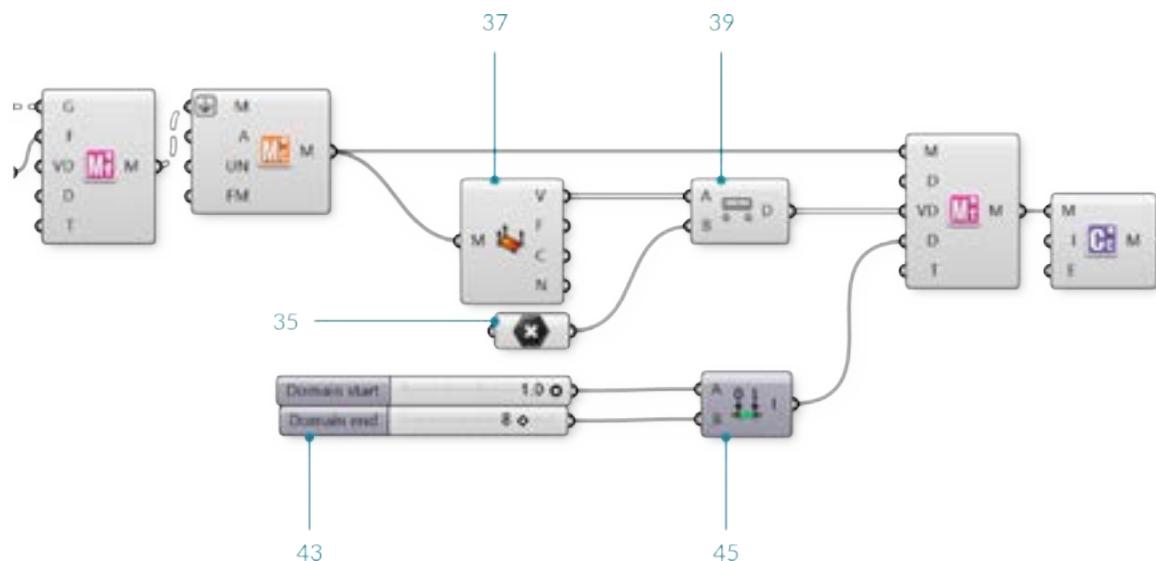
28.	<b>Element*/Utility/Mesh Combine and Clean</b> - Drag and drop a <b>Mesh Combine and Clean</b> component on the canvas	
29.	Connect the Mesh (M) output of <b>Mesh Frames</b> to the Mesh (M) input of the <b>Mesh Combine and Clean</b> component	
30.	Right click the Mesh (M) input of <b>Mesh Combine and Clean</b> and select Flatten By flattening the tree of meshes, **Combine and Clean** will merge all 20 face meshes into a single mesh	
31.	<b>Element*/Transform/Mesh Thicken</b> - Drag and drop a <b>Mesh Thicken</b> component onto the canvas	
32.	Connect the Mesh (M) output of <b>Combine and Clean</b> to the Mesh (M) input of <b>Mesh Thicken</b>	
33.	<b>Element*/Subdivide/Catmull Clark Subdivision</b> - Drag and drop a <b>Catmull Clark Subdivision</b> component onto the canvas	
34.	Connect the Mesh (M) output of <b>Mesh Thicken</b> to the Mesh (M) input of the <b>Catmull Clark Subdivision</b> component	

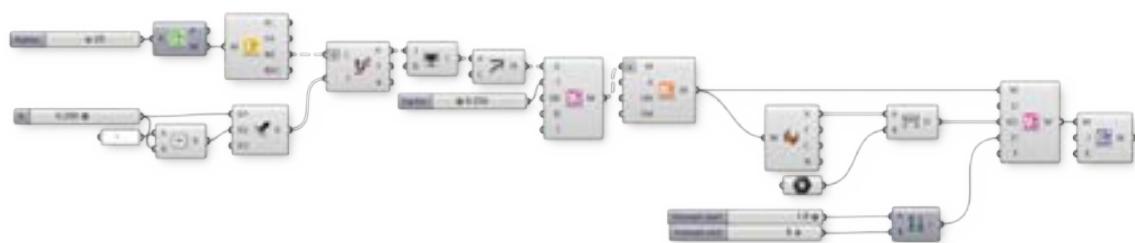
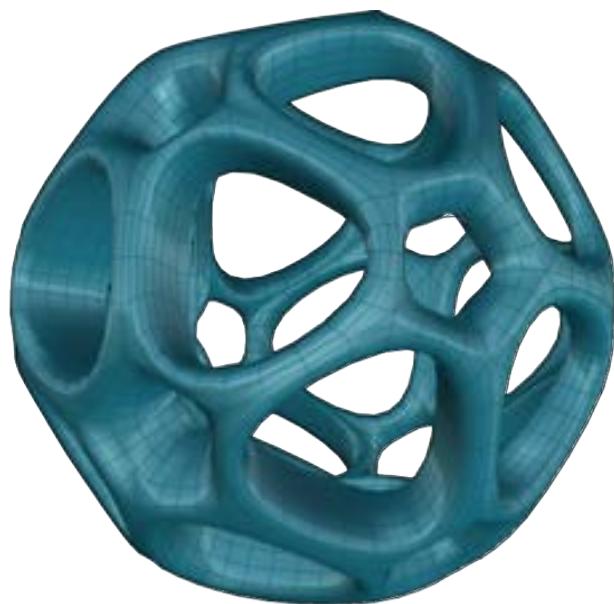


We have truncated the triangular faces of the initial mesh, effectively also creating rings around each original vertex. We have also created a frame for each face, then thickened the mesh and refined it with subdivision. Next we will take advantage of the Per Vertex capabilities of the transform components by using an attractor point.

35.	<b>Params/Geometry/Point</b> - Drag and drop a <b>Point</b> parameter onto the canvas	
	Right click the <b>Point</b> parameter and select "Set on point" to select a point in the Rhino viewport	

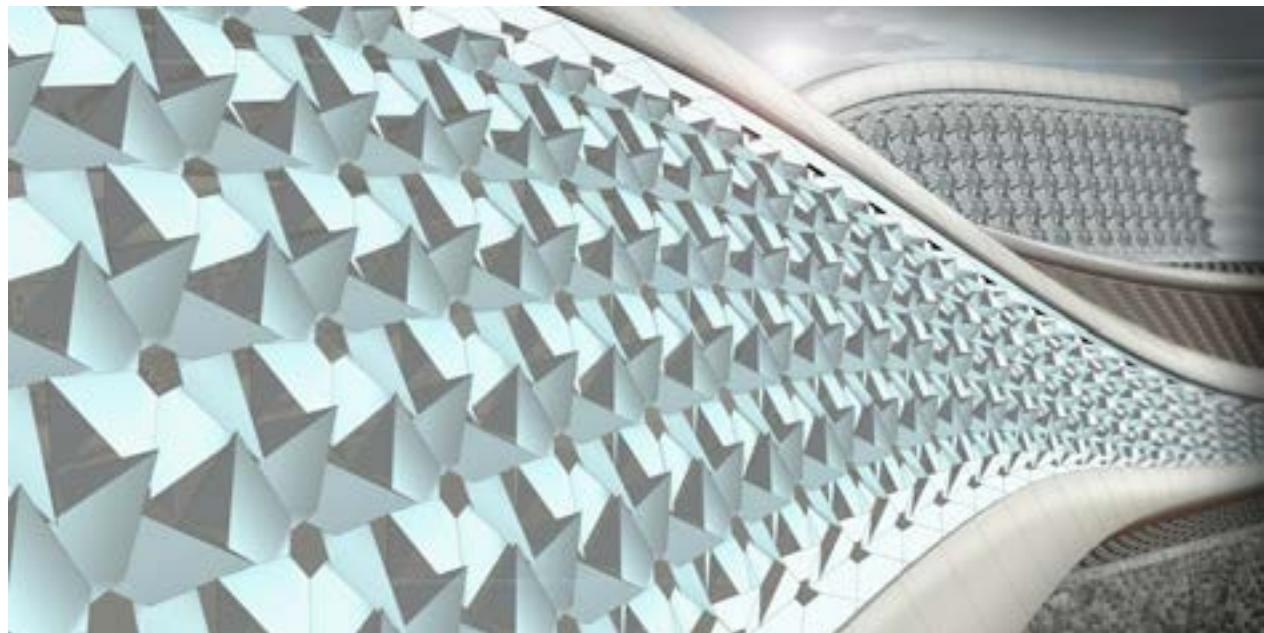
36.	Tip - you can also create a point directly in Grasshopper by double-clicking the canvas to bring up the Search window, then typing a point coordinate such as "10,10,0" (without the quotes)	
37.	<b>Mesh/Analysis/Deconstruct Mesh</b> - Drag and drop a <b>Deconstruct Mesh</b> component onto the canvas 	
38.	Connect the Mesh (M) output of the <b>Combine and Clean</b> component to the Mesh (M) input of the <b>Deconstruct Mesh</b> component. We will use this to extract the vertices of our combined mesh, and then apply an attractor point to these vertices	
39.	<b>Vector/Point/Distance</b> - Drag and drop a <b>Distance</b> component onto the canvas 	
40.	Connect the Vertices (V) output of the <b>Deconstruct Mesh</b> component to the Ainput of the <b>Distance</b> component	
41.	Connect the <b>Point</b> parameter to the B input of the <b>Distance</b> component	
42.	Connect the Distance (D) output of the <b>Distance</b> component to the PerVertex Data (VD) input of the <b>Thicken</b> component	
43.	<b>Params/Input/Number Slider</b> - Drag and drop two <b>Number Slider</b> components onto the canvas. We will use these to set the lower and upper limits for the <b>Mesh Thickener</b> component	
44.	Double-click the <b>Number Sliders</b> and set the values. In this example, we left the first slider at default values, and set the Upper Limit of the second slider to 5.0	
45.	<b>Maths/Domain/Construct Domain</b> - Drag and drop a <b>Construct Domain</b> component onto the canvas 	
46.	Connect the two number sliders to the Aand B inputs of the <b>Construct Domain</b> component	
47.	Connect the Domain (I) output of the <b>Construct Domain</b> component to the Min and Max Values (D) input of the <b>Mesh Thickener</b> component.	
48.	Right click the Type (T) input of the <b>Thicken</b> component, select "Set Integer" and enter a value of 1 You can also enable the PerVertex Data by using a **Boolean Toggle** component set to True.	





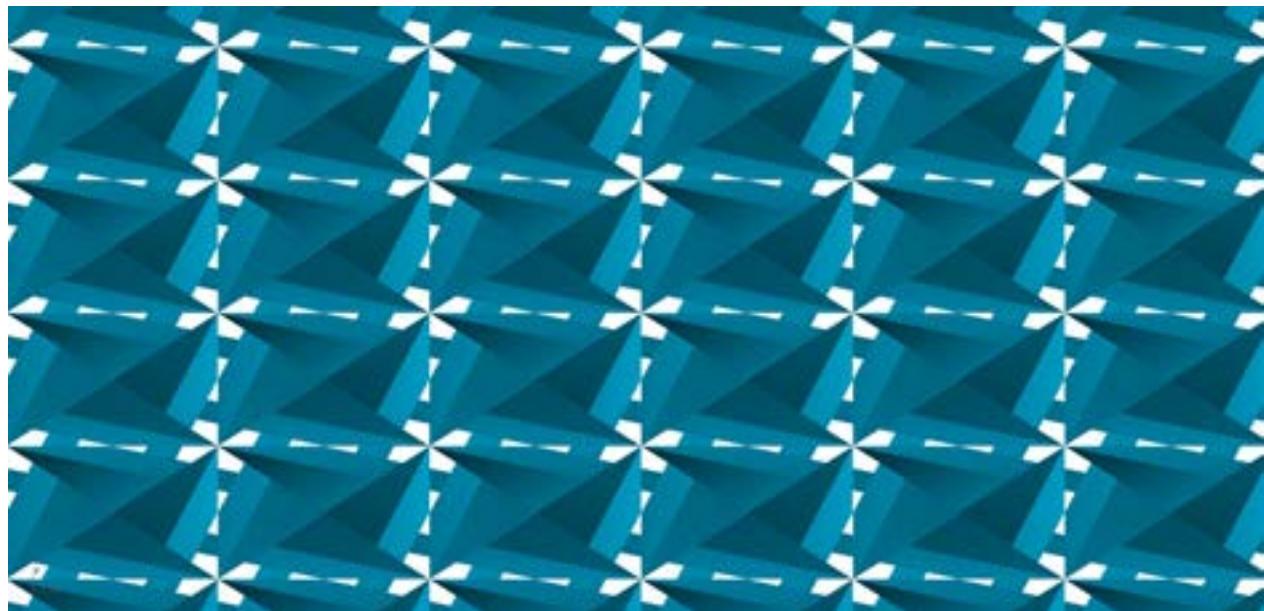
## 2.1.5. Element\* Architectural Case Study

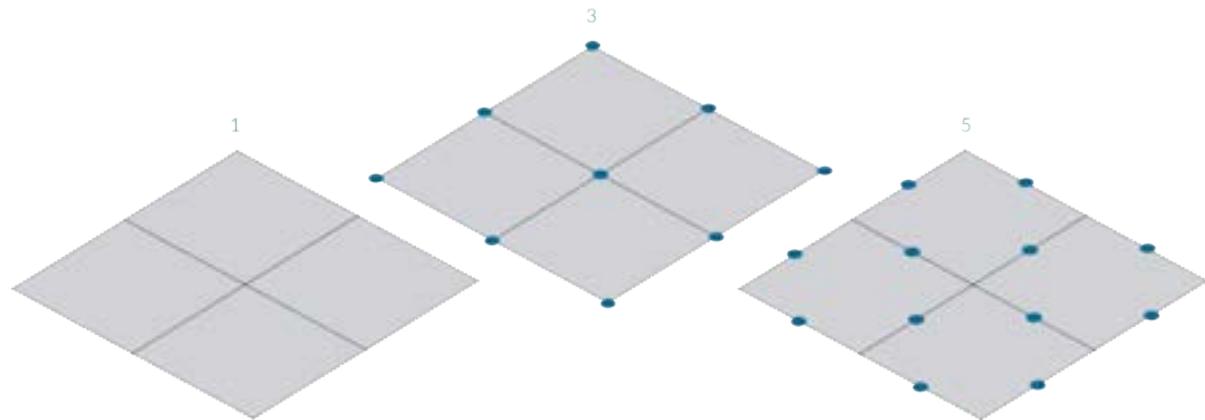
In this section, we will work through a simple exercise file that is meant as an introduction to working with Element tools. We will explore some patterning and facade treatments in the field of Architecture which will incorporate Half Edge data structures along with basic Element components without the use of per vertex features.



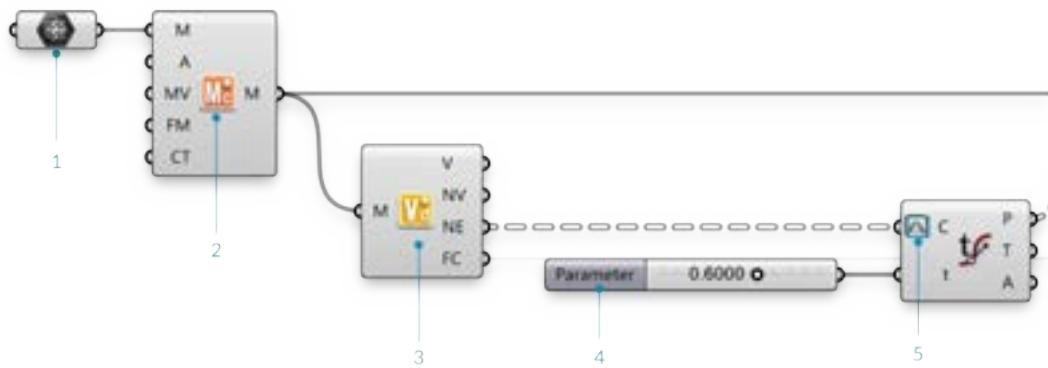
### 2.1.5.1 Example 1

Example files that accompany this section: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

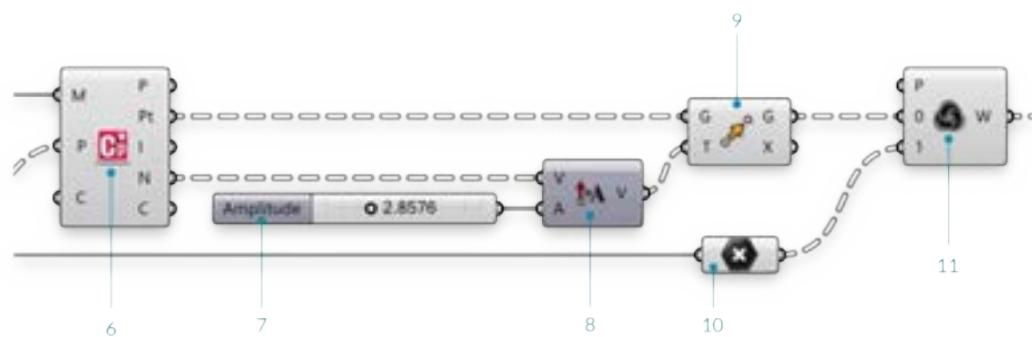


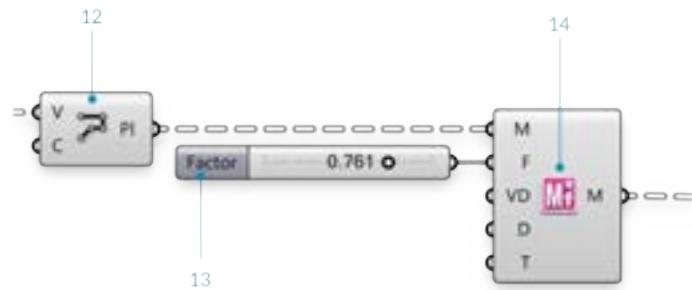


00.	Create a meshplane in Rhino with <b>XFaces = 2 &amp; YFaces = 2</b> and Start a new definition, type Ctrl-N (in Grasshopper)	
01.	<b>Params/Geometry/Mesh</b> - Drag and drop a <b>Mesh</b> container onto the canvas	
01b.	Reference a mesh in Rhino by right-clicking the <b>Mesh</b> component and selecting "Set one Mesh". We are going to use a simple mesh plane to walk through the definition, feel free to swap out the mesh with your own mesh	
02.	<b>Element*/Utility/Mesh Combine and Clean</b> - Drag and drop a <b>Element* Mesh Combine and Clean</b> component on the canvas	
03.	<b>Element*/Data/Vertex Neighbors</b> - Drag and drop the <b>Element* Vertex Neighbors</b> component onto the canvas	
04.	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas and set the following values: Lower Limit: 0.0000 Upper Limit: 1.0000	
05.	<b>Curve/Analysis/Evaluate Curve</b> - Drag and drop a <b>Evaluate Curve</b> container onto the canvas	
05b.	Connect the Neighbouring Edges (NE) output of the <b>Element* Vertex Neighbors</b> component to the Curve (C) input of the <b>Evaluate Curve</b> component	
05c.	Connect the <b>Number Slider</b> to the Float (t) input of the <b>Evaluate Curve</b> component and set the value to 0.5000	
05d.	Right click the Curve (C) input of the <b>Evaluate Curve</b> component and enable <b>Reparameterize</b>	

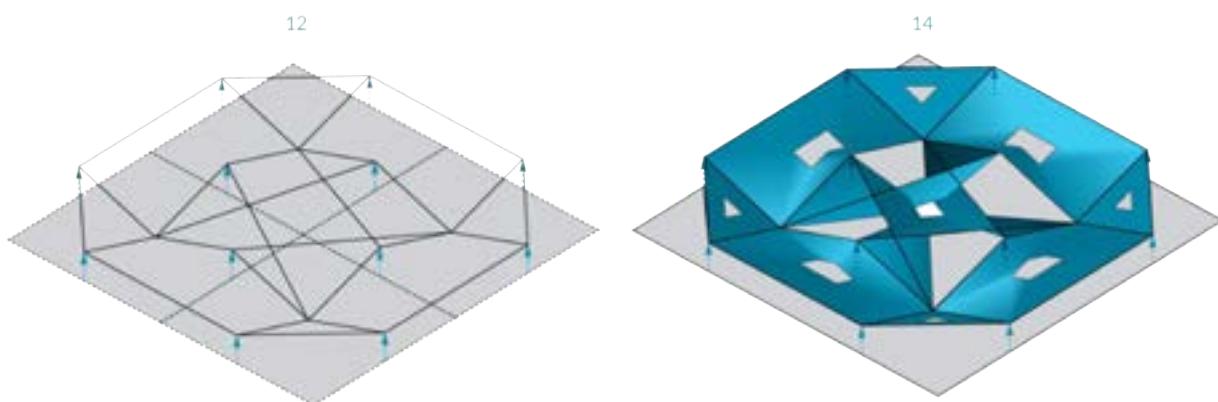


06.	<b>Element*/Analyse/Mesh Closest Point</b> - Drag and drop a <b>Element* Mesh Closest Point</b> container onto the canvas	
06a.	Connect the Mesh output (M) of the <b>Element*/Utility/Mesh Combine and Clean</b> component to the Mesh (M) input of the <b>Element* Mesh Closest Point</b> component	
06b.	Connect the Points output (P) of the <b>Curve/Analysis/Evaluate Curve</b> component to the Point (P) input of the <b>Element* Mesh Closest Point</b> component	
07.	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas and set the following values: Rounding: Float Lower Limit: 0 Upper Limit: 10.000	
08.	<b>Vector/Vector/Amplitude</b> - Drag and drop a <b>Amplitude</b> component onto the canvas	
09.	<b>Transform/Euclidean/Move</b> - Drag and drop a <b>Move</b> component onto the canvas	
10.	<b>Params/Geometry/Point</b> - Drag and drop a <b>Point</b> container onto the canvas	
10b.	Connect the Face Centers output (FC) of the <b>Element* Vertex Neighbors</b> component to the <b>Point</b> component	
11.	<b>Sets/List/Weave</b> - Drag and drop a <b>Weave</b> component onto the canvas	



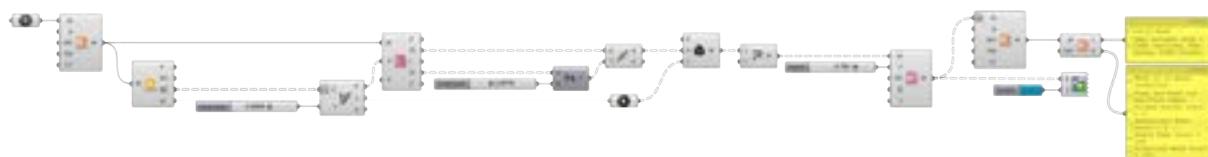
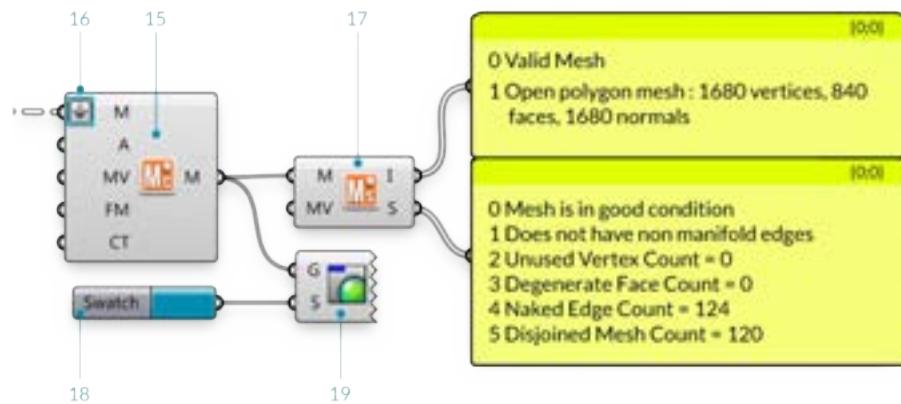


12.	<b>Curve/Primitive/Polyline</b> - Drag and drop a <b>Polyline</b> component onto the canvas	
12b.	Connect the Weave output (W) of the <b>Weave</b> component to the Vertices (V) input of the <b>Polyline</b> component	
12c.	Right click the Closed (C) input of the <b>Polyline</b> component, click "Set Boolean" and set the value to True This has created a closed polyline.	
13.	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas. We will keep the default range of 0 to 1 for this slider	
14.	<b>Element*/Transform/Mesh Frame</b> - Drag and drop a <b>Element* Mesh Frame</b> component onto the canvas.	
14b.	Connect the Polyline (Pl) output of the <b>Polyline</b> component to the Geometry (G) input of the <b>Mesh Frame</b> component Note that the **Mesh Frame** component can take either meshes or a list of closed polyline curves as input	
14c.	Connect the <b>Number Slider</b> to the Factor (F) input of the <b>Mesh Frame</b> component	



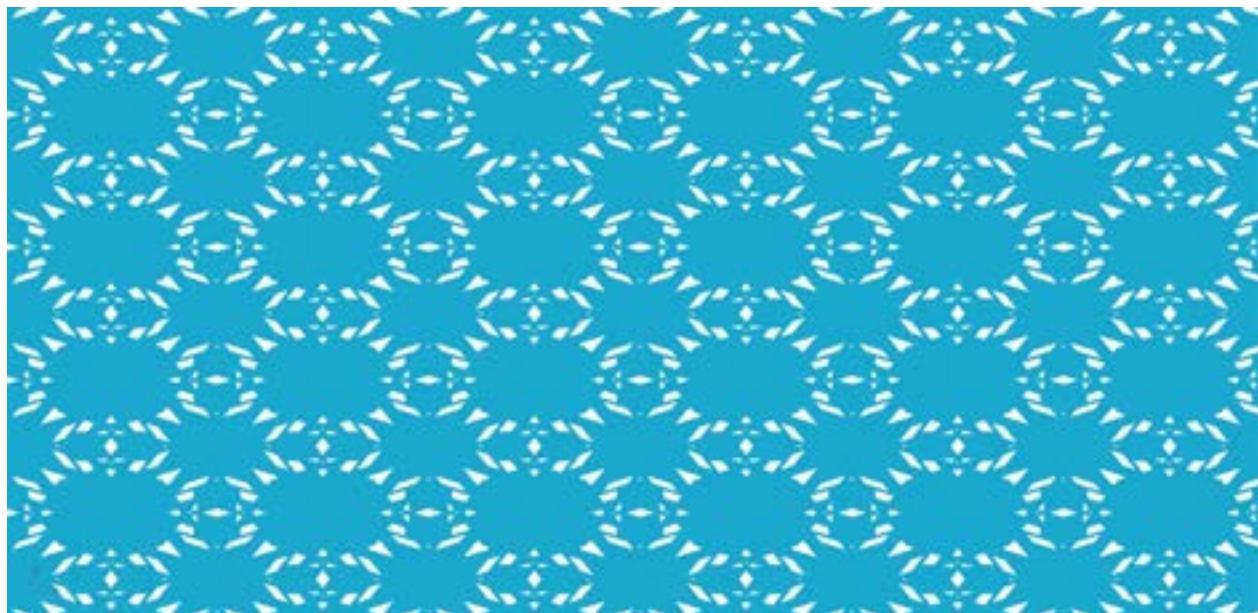
15.	<b>Element*/Utility/Mesh Combine and Clean</b> - Drag and drop a <b>Element* Mesh Combine and Clean</b> component on the canvas	
	Right click the Combine Type (CT) input of the <b>Element* Mesh Combine and Clean</b> component, click "Set Integer" and set the value to 1.	

15b.	The Combine Type input has two options (0, which combines and cleans the meshes) and (1, which joins the meshes in the list without merging vertices). In this example we want to join the meshes
16.	Right click the Mesh (M) input of the <b>Element* Mesh Combine and Clean</b> component, click "Flatten". This will flatten the list so we can join the mesh list together.
17.	<b>Element*/Utility/Mesh Status</b> - Drag and drop a <b>Element* Mesh Status</b> component on the canvas 
17b	Connect the Info (I) and Status (S) outputs of <b>Element* Mesh Status</b> to a <b>Params/Input/Panel</b> component The mesh **Info** output contains mesh validity information, closed or open type and mesh component counts (vertices, faces, normals). The mesh **Status** informs the user if the mesh is in "Good" condition as well as data regarding non manifold edges, unused vertex count, degenerate face count, naked edge count and disjoined mesh count.
18.	<b>Params/Input/Colour Swatch</b> - Drag and drop a <b>Colour Swatch</b> component on the canvas
19.	<b>Display/Preview/Custom Preview</b> - Drag and drop a <b>Custom Preview</b> component on the canvas

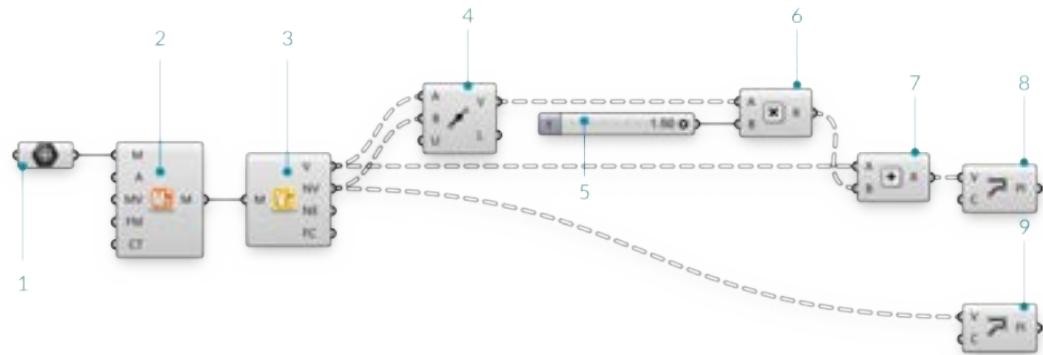


### 2.1.5.2 Example 2

Example files that accompany this section: [http://grasshopperprimer.com/appendix/A-2/1\\_gh-files.html](http://grasshopperprimer.com/appendix/A-2/1_gh-files.html)

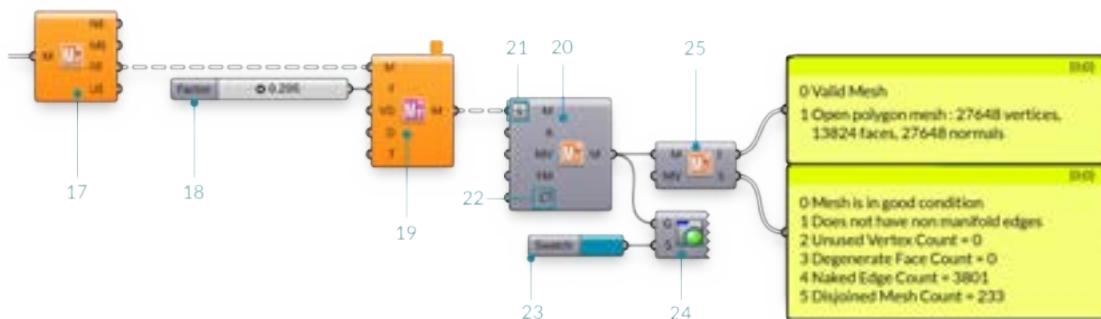
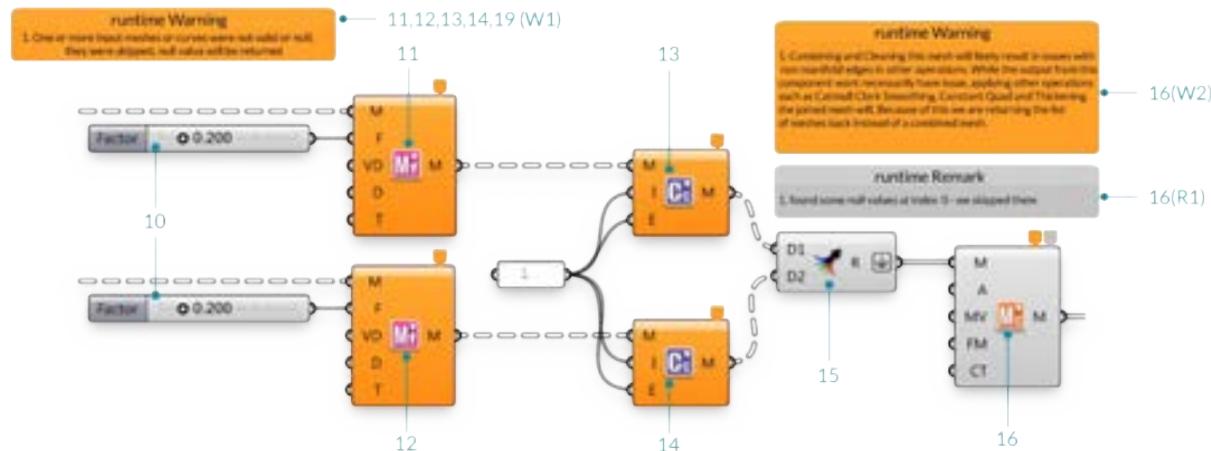


00.	Create a meshplane in Rhino with <b>XFaces = 2 &amp; YFaces = 2</b> and Start a new definition, type Ctrl-N (in Grasshopper)	
01.	<b>Params/Geometry/Mesh</b> - Drag and drop a <b>Mesh</b> container onto the canvas	
01b.	Reference a mesh in Rhino by right-clicking the <b>Mesh</b> component and selecting "Set one Mesh". We are going to use a simple mesh plane to walk through the definition, feel free to swap out the mesh with your own mesh	
02.	<b>Element*/Utility/Mesh Combine and Clean</b> - Drag and drop a <b>Element* Mesh Combine and Clean</b> component on the canvas	
03.	<b>Element*/Data/Vertex Neighbors</b> - Drag and drop the <b>Element* Vertex Neighbors</b> component onto the canvas	
04.	<b>Vector/Vector/Vector2Pt</b> - Drag and drop a <b>Vector2Pt</b> component onto the canvas	
05.	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas and set the following values: Rounding: Float Lower Limit:0 Upper Limit: 2.000	
06.	<b>Maths/Operator/Multiplication</b> - Drag and drop a <b>Multiplication</b> component onto the canvas	
07.	<b>Maths/Operators/Addition</b> - Drag and drop two <b>Addition</b> components onto the canvas	
08.	<b>Curve/Primitive/Polyline</b> - Drag and drop a <b>Polyline</b> component onto the canvas	
09.	<b>Curve/Primitive/Polyline</b> - Drag and drop a <b>Polyline</b> component onto the canvas	



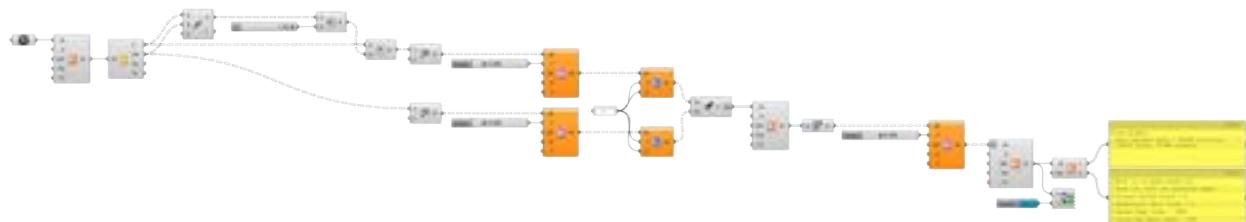
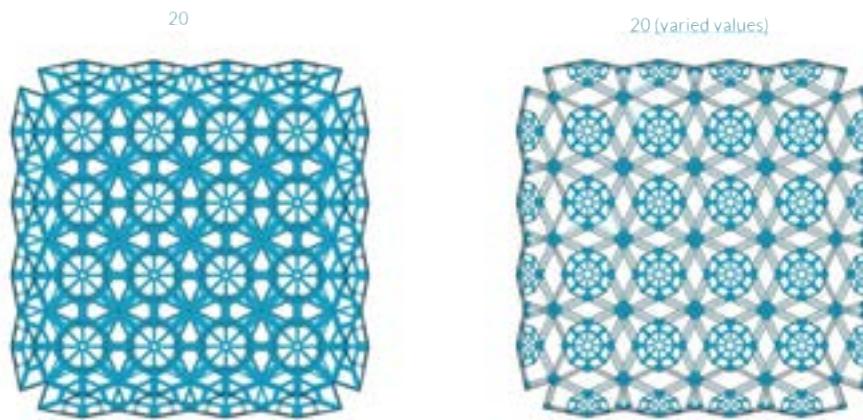
	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas and set the following values: Rounding: Float Lower Limit:0 Upper Limit: 1.000	
10.	<b>Element*/Transform/Mesh Frame</b> - Drag and drop a <b>Element* Mesh Frame</b> component onto the canvas.	
11,12.	Connect the Polyline (PI) output of the <b>Polyline</b> component to the Geometry (G) input of the <b>Mesh Frame</b> component	
11b,12b.	<p>Note that the **Mesh Frame** component can take either meshes or a list of closed polyline curves as input</p>	
11c,12c.	Connect the <b>Number Slider (10)</b> to the Factor (F) input of the <b>Mesh Frame</b> component	
13,14.	<b>Element*/Subdivide/Catmull Clark Subdivision</b> - Drag and drop a <b>Catmull Clark Subdivision</b> component onto the canvas  We will set the Iterations input (I) value to 1 as well as the **Edge Condition** input (E) to a value of 1. The edge condition input options are 0 = Fixed, 1 == Smooth, 2 == Corners Fixed.	
15.	<b>Sets/Tree/Merge</b> - Drag and drop two <b>Merge</b> components onto the canvas	
15b.	Right click the Result (R) output of the <b>Merge</b> component and click "Flatten".	
16.	<b>Element*/Utility/Mesh Combine and Clean</b> - Drag and drop a <b>Element* Mesh Combine and Clean</b> component on the canvas	

Components have detailed remarks and warnings to inform the user of the current or potential issues that might come about from interaction with other components. In some instances you might use the *Element Combine and Clean component* to join and combine identical vertices on a mesh which could lead to non manifold edges if that mesh is thickened later on. The *Element Combine and Clean component* will inform you of this issue that will return the list back to you. You have the option of setting the Combine Type to a value of 1 which will combine the meshes in the list but not combine identical vertices.



17.	<b>Element*/Utility/Mesh Edges</b> - Drag and drop a <b>Element* Mesh Edges</b> component on the canvas	
17b	Connect the Mesh (M) output of the <b>Element* Mesh Combine and Clean</b> component (16) to the Mesh input (M) of the <b>Element* Mesh Edges</b> component	
18.	<b>Params/Input/Number Slider</b> - Drag and drop a <b>Number Slider</b> component onto the canvas and set the following values: Rounding: Float Lower Limit:0 Upper Limit: 1.000	
19.	<b>Element*/Transform/Mesh Frame</b> - Drag and drop a <b>Element* Mesh Frame</b> component onto the canvas.	
19b	Connect the Face Polylines (FP) output of the <b>Element* Mesh Edges</b> component to the Mesh input (M) of the <b>Element* Mesh Frame</b> component	
19c	Connect the <b>Number Slider</b> to the Float (f) input of the <b>Element* Mesh Frame</b> component	
20.	<b>Element*/Utility/Mesh Combine and Clean</b> - Drag and drop a <b>Element* Mesh Combine and Clean</b> component on the canvas	
21.	Right click the Mesh (M) input of the <b>Element* Mesh Combine and Clean</b> component and click "Flatten".	
	Right click the Combine Type (CT) input of the <b>Element* Mesh Combine and Clean</b>	

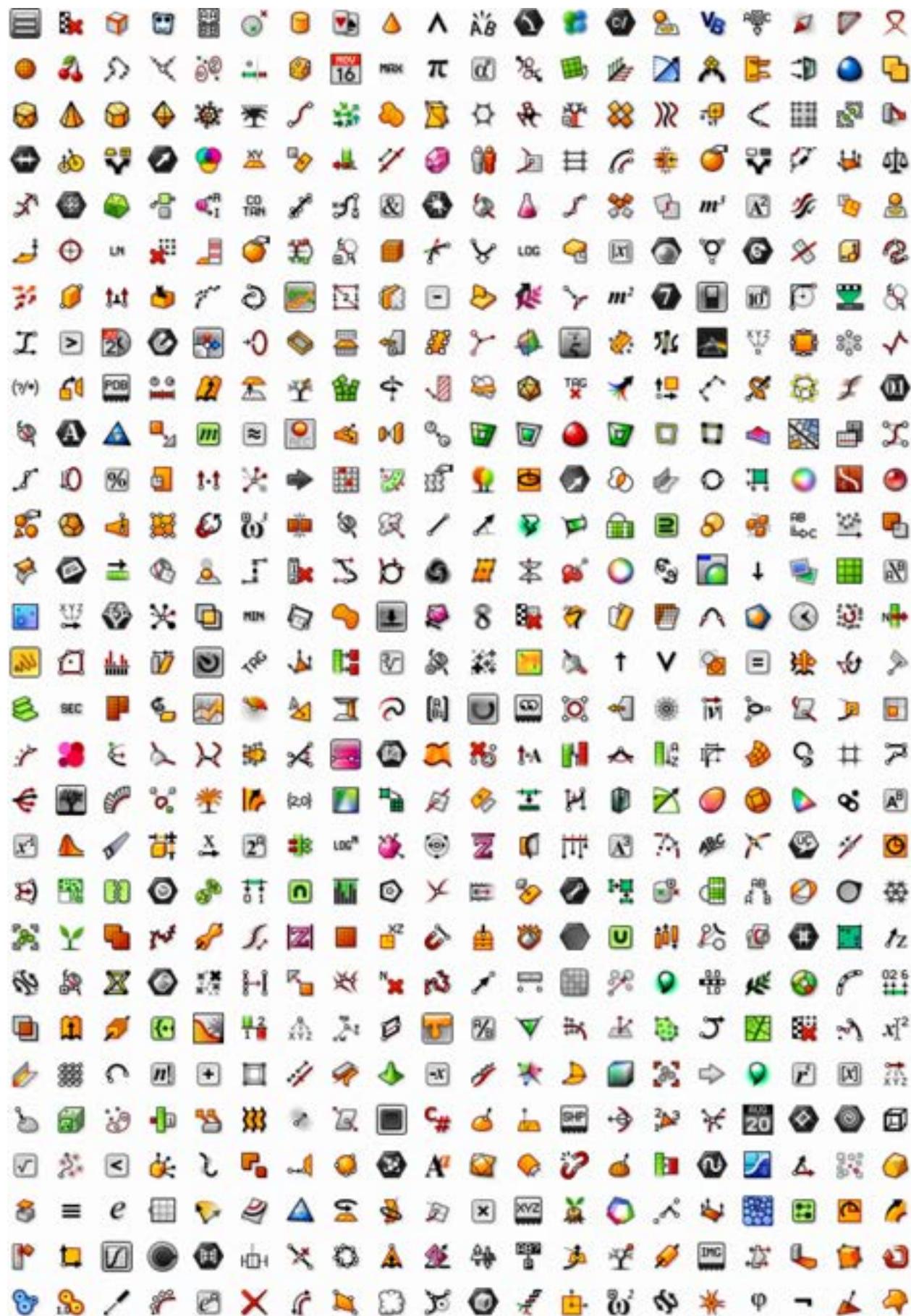
	component, click "Set Integer" and set the value to 1. The Combine Type input has two options (0, which combines and cleans the meshes) and (1, which joins the meshes in the list without merging vertices). In this example we want to join the meshes	
22.		
23.	<b>Params/Input/Colour Swatch</b> - Drag and drop a <b>Colour Swatch</b> component on the canvas	
24.	<b>Display/Preview/Custom Preview</b> - Drag and drop a <b>Custom Preview</b> component on the canvas	
25.	<b>Element*/Utility/Mesh Status</b> - Drag and drop a <b>Element* Mesh Status</b> component on the canvas	
25b	Connect the Info (I) and Status (S) outputs of <b>Element* Mesh Status</b> to a <b>Params/Input/Panel</b> component  The mesh **Info** output contains mesh validity information, closed or open type and mesh component counts (vertices, faces, normals). The mesh **Status** informs the user if the mesh is in "Good" condition as well as data regarding non manifold edges, unused vertex count, degenerate face count, naked edge count and disjoined mesh count.	



## Appendix

---

**The following section contains useful references including an index of all the components used in this primer, as well as additional resources to learn more about Grasshopper.**

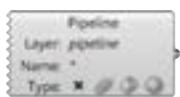


## 2.1. Index

This index provides additional information on all the components used in this primer, as well as other components you might find useful. This is just an introduction to over 500 components in the Grasshopper plugin.

### Parameters

#### Geometry

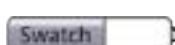
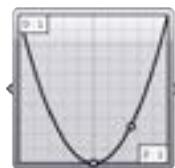
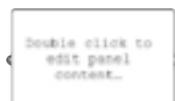
P.G.Crv	Curve Parameter Represents a collection of Curve geometry. Curve geometry is the common denominator of all curve types in Grasshopper.	
P.G.Circle	Circle Parameter Represents a collection of Circle primitives.	
P.G.Geo	Geometry Parameter Represents a collection of 3D Geometry.	
P.G.Pipeline	Geometry Pipeline Defines a geometry pipeline from Rhino to Grasshopper.	
P.G.Pt	Point Parameter Point parameters are capable of storing persistent data. You can set the persistent records through the parameter menu.	
P.G.Srf	Surface Parameter Represents a collection of Surface geometry. Surface geometry is the common denominator of all surface types in Grasshopper.	

#### Primitive

P.P.Bool	Boolean Parameter Represents a collection of Boolean (True/False) values.	
P.P.D	Domain Parameter Represents a collection of one-dimensional Domains. Domains are typically used to represent curve fragments and continuous numeric ranges. A domain consists of two numbers that indicate the limits of the domain, everything in between these numbers is part of the domain.	
P.P.D2	Domain <sup>2</sup> Parameter Contains a collection of two-dimensional domains. 2D Domains are typically used to represent surface fragments. A two-dimensional domain consists of two one-dimensional domains.	
P.P.ID	Guid Parameter Represents a collection of Globally Unique Identifiers. Guid parameters are capable of storing persistent data. You can set the persistent records through the parameter menu.	
P.P.Int	Integer Parameter Represents a collection of Integer numeric values. Integer parameters are capable of storing persistent data. You can set	

	the persistent records through the parameter menu.	
P.P.Num	Number Parameter Represents a collection of floating point values. Number parameters are capable of storing persistent data. You can set the persistent records through the parameter menu.	
P.P.Path	File Path Contains a collection of file paths.	

## Input

P.I.Toggle	Boolean Toggle Boolean (true/false) toggle.	
P.I.Button	Button Button object with two values. When pressed, the button object returns a true value and then resets to false.	
P.I.Swatch	Color Swatch A swatch is a special interface object that allows for quick setting of individual color values. You can change the color of a swatch through the context menu.	
P.I.Grad	Gradient Control Gradient controls allow you to define a color gradient within a numeric domain. By default the unit domain (0.0 ~ 1.0) is used, but this can be adjusted via the L0 and L1 input parameters. You can add color grips to the gradient object by dragging from the color wheel at the upper left and set color grips by right clicking them.	
P.I.Graph	Graph Mapper Graph mapper objects allow you to remap a set of numbers. By default the {x} and {y} domains of a graph function are unit domains (0.0 ~ 1.0), but these can be adjusted via the Graph Editor. Graph mappers can contain a single mapping function, which can be picked through the context menu. Graphs typically have grips (little circles), which can be used to modify the variables that define the graph equation. By default, a graph mapper object contains no graph and performs a 1:1 mapping of values.	
P.I.Slider	Number Slider A slider is a special interface object that allows for quick setting of individual numeric values. You can change the values and properties through the menu, or by double-clicking a slider object. Sliders can be made longer or shorter by dragging the rightmost edge left or right. Note that sliders only have an output (ie. no input).	
P.I.Panel	Panel A panel for custom notes and text values. It is typically an inactive object that allows you to add remarks or explanations to a Document. Panels can also receive their information from elsewhere. If you plug an output parameter into a Panel, you can see the contents of that parameter in real-time. All data in Grasshopper can be viewed in this way. Panels can also stream their content to a text file.	
P.I.List	Value List	

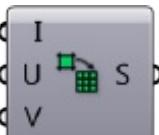
## Utilities

P.U.Cin	Cluster Input Represents a cluster input parameter.	
P.U.COut	Cluster Output Represents a cluster input parameter.	
P.U.Dam	Data Dam Delay data on its way through the document.	
P.U.Jump	Jump Jump between different locations.	
P.U.Viewer	Param Viewer A viewer for data structures.	
P.U.Scribble	Scribble A quick note.	Doubleclick Me!

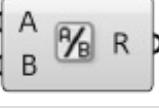
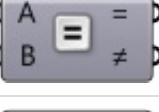
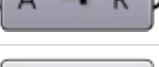
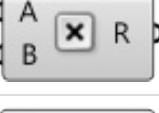
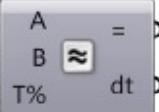
## Maths

### Domain

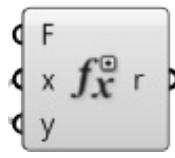
M.D.Bnd	Bounds Create a numeric domain which encompasses a list of numbers.	
M.D.Consec	Consecutive Domains Create consecutive domains from a list of numbers.	
M.D.Dom	Construct Domain Create a numeric domain from two numeric extremes.	
M.D.Dom2Num	Construct Domain <sup>2</sup> Create a two-dimensional domain from four numbers.	
M.D.DeDomain	Deconstruct Domain Deconstruct a numeric domain into its component parts.	
M.D.DeDom2Num	Deconstruct Domain <sup>2</sup> Deconstruct a two-dimensional domain into four numbers.	
M.D.Divide	Divide Domain <sup>2</sup> Divides a two-dimensional domain into equal segments.	

M.D.Divide	Divide Domain <sup>2</sup> Divides a two-dimensional domain into equal segments.	
M.D.Inc	Includes Test a numeric value to see if it is included in the domain.	
M.D.ReMap	Remap Numbers Remap numbers into a new numeric domain.	

## Operators

M.O.Add	Addition Mathematical addition.	
M.O.Div	Division Mathematical division.	
M.O.Equals	Equality Test for (in)equality of two numbers.	
M.O.And	Gate And Perform boolean conjunction (AND gate). Both inputs need to be True for the result to be True.	
M.O.Not	Gate Not Perform boolean negation (NOT gate).	
M.O.Or	Gate Or Perform boolean disjunction (OR gate). Only a single input has to be True for the result to be True.	
M.O.Larger	Larger Than Larger than (or equal to).	
M.O.Multiply	Multiplication Mathematical multiplication.	
M.O.Smaller	Smaller Than Larger than (or equal to).	
M.O.Similar	Similarity Test for similarity of two numbers.	
M.O.Sub	Subtraction Mathematical subtraction.	

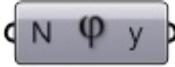
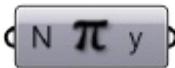
## Script

M.S.Eval	Evaluate Evaluate an expression with a flexible number of variables.	
M.S.Expression	Expression Evaluate an expression.	

## Trig

M.T.Cos	Cosine Compute the cosine of a value.	
M.T.Deg	Degrees Convert an angle specified in radians to degrees.	
M.T.Rad	Radians Convert an angle specified in degrees to radians.	
M.T.Sin	Sine Compute the sine of a value.	

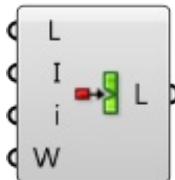
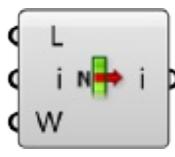
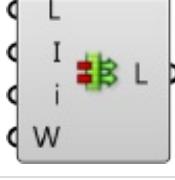
## Utilities

M.U.Avr	Average Solve the arithmetic average for a set of items.	
M.U.Phi	Golden Ratio Returns a multiple of the golden ratio (Phi).	
M.U.Pi	Pi Returns a multiple of Pi.	

## Sets

### List

S.L.Combine	Combine Data Combine non-null items out of several inputs.	
S.L.CrossRef	Cross Reference Cross Reference data from multiple lists.	
S.L.Dispatch	Dispatch Dispatch the items in a list into two target lists. List dispatching is very similar to the [Cull Pattern] component, with the exception that both lists are provided as outputs.	
S.L.Ins	Insert Items Insert a collection of items into a list.	

		
S.L.Item	List Item Retrieve a specific item from a list.	
S.L.Lng	List Length Measure the length of a list. Elements in a list are identified by their index. The first element is stored at index zero, the second element is stored at index one and so on and so forth. The highest possible index in a list equals the length of the list minus one.	
S.L.Long	Longest List Grow a collection of lists to the longest length amongst them.	
S.L.Split	Split List Split a list into separate parts.	
S.L.Replace	Replace Items Replace certain items in a list.	
S.L.Rev	Reverse List Reverse the order of a list. The new index of each element will be N-i where N is the highest index in the list and i is the old index of the element.	
S.L.Shift	Shift List Offset all items in a list. Items in the list are offset (moved) towards the end of the list if the shift offset is positive. If Wrap equals True, then items that fall off the ends are re-appended.	
S.L.Short	Shortest List Shrink a collection of lists to the shortest length amongst them.	
S.L.Sift	Sift Pattern Sift elements in a list using a repeating index pattern.	
S.L.Sort	Sort List Sort a list of numeric keys. In order for something to be sorted, it must first be comparable. Most types of data are not comparable, Numbers and Strings being basically the sole exceptions. If you want to sort other types of data, such as curves, you'll need to create a list of keys first.	
S.L.Weave	Weave	

Weave a set of input data using a custom pattern. The pattern is specified as a list of index values (integers) that define the order in which input data is collected.

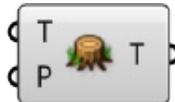
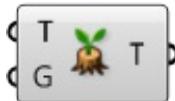


## Sets

S.S.Culli	Cull Index Cull (remove) indexed elements from a list.	
S.S.Cull	Cull Pattern Cull (remove) elements in a list using a repeating bit mask. The bit mask is defined as a list of Boolean values. The bit mask is repeated until all elements in the data list have been evaluated.	
S.S.Dup	Duplicate Data Duplicate data a predefined number of times. Data can be duplicated in two ways, either copies of the list are appended at the end until the number of copies has been reached, or each item is duplicated a number of times before moving on to the next item.	
S.S.Jitter	Jitter Randomly shuffles a list of values. The input list is reordered based on random noise. Jittering is a good way to get a random set with a good distribution. The jitter parameter sets radius of the random noise. If jitter equals 0.5, then each item is allowed to reposition itself randomly to within half the span of the entire set.	
S.S.Random	Random Generate a list of pseudo random numbers. The number sequence is unique but stable for each seed value. If you do not like a random distribution, try different seed values.	
S.S.Range	Range Create a range of numbers. The numbers are spaced equally inside a numeric domain. Use this component if you need to create numbers between extremes. If you need control over the interval between successive numbers, you should be using the [Series] component.	
S.S.Repeat	Repeat Data Repeat a pattern until it reaches a certain length.	
S.S.Series	Series Create a series of numbers. The numbers are spaced according to the {Step} value. If you need to distribute numbers inside a fixed numeric range, consider using the [Range] component instead.	

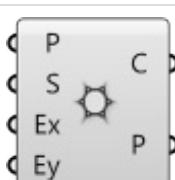
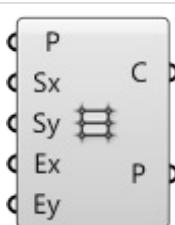
## Tree

S.T.Explode	Explode Tree Extract all the branches from a tree.	
-------------	---	--

S.T.Flatten	Flatten Tree Flatten a data tree by removing all branching information.	
S.T.Flip	Flip Matrix Flip a matrix-like data tree by swapping rows and columns.	
S.T.Graft	Graft Tree Typically, data items are stored in branches at specific index values (0 for the first item, 1 for the second item, and so on and so forth) and branches are stored in trees at specific branch paths, for example: {0;1}, which indicates the second sub-branch of the first main branch. Grafting creates a new branch for every single data item.	
S.T.Merge	Merge Merge a bunch of data streams.	
S.T.Path	Path Mapper Perform lexical operations on data trees. Lexical operations are logical mappings between data paths and indices which are defined by textual (lexical) masks and patterns.	
S.T.Prune	Prune Tree Removes all branches from a Tree that carry a special number of Data items. You can supply both a lower and an upper limit for branch pruning.	
S.T.Simplify	Simplify Tree Simplify a tree by removing the overlap shared amongst all branches.	
S.T.TStat	Tree Statistics Get some statistics regarding a data tree.	
S.T.Unflatten	Unflatten Tree Unflatten a data tree by moving items back into branches.	

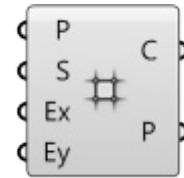
## Vector

### Grid

V.G.HexGrid	Hexagonal 2D grid with hexagonal cells.	
V.G.RecGrid	Rectangular 2D grid with rectangular cells.	

V.G.SqGrid

Square  
2D grid with square cells



## Point

V.Pt	Construct Point Construct a point from {xyz} coordinates.	
V.P.pDecon	Deconstruct Deconstruct a point into its component parts.	
V.Dist	Distance Compute Euclidean distance between two point coordinates.	

## Vector

V.V.X	Unit X Unit vector parallel to the world {x} axis.	
V.V.Y	Unit Y Unit vector parallel to the world {y} axis.	
V.V.Vec2Pt	Vector 2Pt Create a vector between two points.	

## Curve

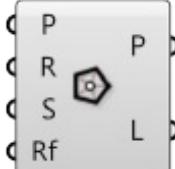
### Analysis

C.ACP	Control Points Extract the nurbs control points and knots of a curve.	
-------	--	--

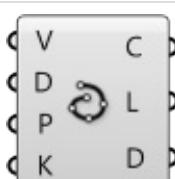
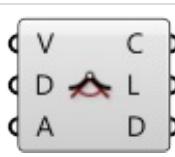
### Division

C.D.Divide	Divide Curve Divide a curve into equal length segments.	
------------	--	--

## Primitive

C.P.Cir	Circle Create a circle defined by base plane and radius.	
C.P.Cir3Pt	Circle 3Pt Create a circle defined by three points.	
C.P.CirCNR	Circle CNR Create a circle defined by center, normal and radius.	
C.P.Line	Line SDL Create a line segment defined by start point, tangent and length.	
C.P.Polygon	Polygon Create a polygon with optional round edges.	

## Spline

C.S.IntCrv	Interpolate Create an interpolated curve through a set of points.	
C.S.KinkCrv	Kinky Curve Construct an interpolated curve through a set of points with a kink angle threshold.	
C.S.Nurbs	Nurbs Curve Construct a nurbs curve from control points.	
C.S.PLine	PolyLine Create a polyline connecting a number of points.	

## Util

C.U.Explode	Explode Explode a curve into smaller segments.	
-------------	---	---

<C.U.Join	Join Curves Join as many curves as possible.	
C.U.Offset	Offset Offset a curve with a specified distance.	

## Surface

### Analysis

S.A.DeBrep	Deconstruct Brep Deconstruct a brep into its constituent parts.	
------------	--	---

### Freeform

S.F.Boundary	Boundary Surfaces Create planar surfaces from a collection of boundary edge curves.	
S.F.Extr	Extrude Extrude curves and surfaces along a vector.	
S.F.ExtrPt	Extrude Point Extrude curves and surfaces to a point.	
S.F.Loft	Loft Create a lofted surface through a set of section curves.	
S.F.RevSrf	Revolution Create a surface of revolution.	
S.F.Swp2	Sweep2 Create a sweep surface with two rail curves.	

### Primitive

S.P.BBox	Bounding Box Solve oriented geometry bounding boxes.	
----------	---	--



## Util

S.U.SDivide	Divide Surface Generate a grid of {uv} points on a surface.	
S.U.SubSrf	Isotrim Extract an isoparametric subset of a surface.	

## Mesh

### Triangulation

M.T.Voronoi	Voronoi Planar voronoi diagram for a collection of points.	
-------------	---	--

## Transform

### Affine

T.A.RecMap	Rectangle Mapping Transform geometry from one rectangle into another.	
------------	--	--

### Array

T.A.ArrLinear	Linear Array Create a linear array of geometry.	
---------------	--	--

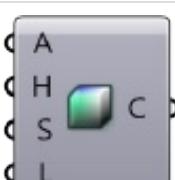
### Morph

T.M.Morph	Box Morph Morph an object into a twisted box.	
-----------	--	--

T.M.SBox	Surface Box Create a twisted box on a surface patch.	
----------	---	---

## Display

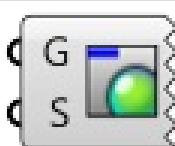
### Color

D.C.HSL	Colour HSL Create a colour from floating point {HSL} channels.	
---------	---	---

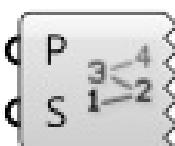
### Dimensions

D.D.Tag	Text tags A text tag component allows you to draw little Strings in the viewport as feedback items. Text and location are specified as input parameters. When text tags are baked they turn into Text Dots.	
D.D.Tag3D	Text Tag 3D Represents a list of 3D text tags in a Rhino viewport	

### Preview

D.P.Preview	Custom Preview Allows for customized geometry previews.	
-------------	--	---

### Vector

D.V.Points	Point List Displays details about lists of points.	
------------	---	---

## 2.2. Grasshopper Example Files

These example files accompany the Grasshopper Primer, and are organized according to section.

<b>1.2.</b>	<a href="#">1.2.5_the grasshopper definition.gh</a>
<b>1.3.</b>	<a href="#">1.3.2.1_attractor definition.gh</a> <a href="#">1.3.3_operators and conditionals.gh</a> <a href="#">1.3.3.4_trigonometry components.gh</a> <a href="#">1.3.3.5_expressions.gh</a> <a href="#">1.3.4_domains and color.gh</a> <a href="#">1.3.5_booleans and logical operators.gh</a>
<b>1.4.</b>	<a href="#">1.4.1.2_grasshopper spline components.gh</a> <a href="#">1.4.3_data matching.gh</a> <a href="#">1.4.4_list creation.gh</a> <a href="#">1.4.5_list visualization.gh</a> <a href="#">1.4.6_list management.gh</a> <a href="#">1.4.7_working with lists.gh</a>
<b>1.5.</b>	<a href="#">1.5.1.3_morphing definition.gh</a> <a href="#">1.5.2.1_Data Tree Visualization.gh</a> <a href="#">1.5.3_working with data trees.gh</a> <a href="#">1.5.3.6_weaving definition.gh</a> <a href="#">1.5.4_rail intersect definition.gh</a>

## 2.3. Resources

**There are many resources available to learn more about Grasshopper and parametric design concepts. There are also over a hundred plugins and add-ons that extend Grasshopper's functionality. Below are some of our favorites.**

### Plug-in Communities



food4Rhino (WIP) is the new Plug-in Community Service by McNeel. As a user, find the newest Rhino Plug-ins, Grasshopper Add-ons, Textures and Backgrounds, add your comments, discuss about new tools, get in contact with the developers of these applications, share your scripts. <http://www.food4rhino.com/>



Grasshopper add-ons page <http://www.grasshopper3d.com/page/addons-forgrasshopper>

### Add-ons We Love



DIVA-for-Rhino allows users to carry out a series of environmental performance evaluations of individual buildings and urban landscapes. <http://diva4rhino.com/>



Element is a mesh geometry plugin for Grasshopper, enabling mesh creation, analysis, transformation, subdivision, and smoothing. <http://www.food4rhino.com/project/element>



Firefly offers a set of comprehensive software tools dedicated to bridging the gap between Grasshopper and the Arduino micro-controller. <http://fireflyexperiments.com>



GhPython is the Python interpreter component for Grasshopper that allows you to execute dynamic scripts of any type. Unlike other scripting components, GhPython allows the use of rhinoscriptsyntax to start scripting without needing to be a programmer. <http://www.food4rhino.com/project/ghpython>



HAL is a Grasshopper plugin for industrial robots programming supporting ABB, KUKA and Universal Robots machines. <http://hal.thibaultschwartz.com/>



Extends Grasshopper's ability to create and reference geometry including lights, blocks, and text objects. Also enables access to information about the active Rhino document, pertaining to materials, layers, linetypes, and other settings. <http://www.food4rhino.com/project/human>



Karamba is an interactive, parametric finite element program. It lets you analyze the response of 3-dimensional beam and shell structures under arbitrary loads. <http://www.karamba3d.com/>



Kangaroo is a Live Physics engine for interactive simulation, optimization and form-finding directly within Grasshopper. <http://www.food4rhino.com/project/kangaroo>



Fold panels using curved folding and control panel distribution on surfaces with a range of attractor systems. <http://www.food4rhino.com/project/robofoldkingkong>



LunchBox is a plug-in for Grasshopper for exploring mathematical shapes, paneling, structures, and workflow. <http://www.food4rhino.com/project/lunchbox>



Meshedit is a set of components which extend Grasshopper's ability to work with meshes. <http://www.food4rhino.com/project/meshedittools>



Parametric tools to create and manipulate rectangular grids, attractors and support creative morphing of parametric patterns. <http://www.food4rhino.com/project/pt-gh>



Platypus allows Grasshopper authors to stream geometry to the web in real time. It works like a chatroom for parametric geometry, and allows for on-the-fly 3D model mashups in the web browser.

<http://www.food4rhino.com/project/platypus>



TT Toolbox features a range of different tools that we from the Core Studio at Thornton Tomasetti use on a regular basis, and we thought some of you might appreciate these. <http://www.food4rhino.com/project/tttoolbox>



Weaverbird is a topological modeler that contains many of the known subdivision and transformation operators, readily usable by designers. This plug-in reconstructs the shape, subdivides any mesh, even made by polylines, and helps preparing for fabrication. <http://www.giuliopiacentino.com/weaverbird/>

## Additional Primers

**The Firefly Primer** This book is intended to teach the basics of electronics (using an Arduino) as well as various digital/physical prototyping techniques to people new to the field. It is not a comprehensive book on electronics (as there are already a number of great resources already dedicated to this topic). Instead, this book focuses on expediting the prototyping process. Written by Andrew Payne. <http://fireflyexperiments.com/resources/>

**Essential Mathematics** Essential Mathematics uses Grasshopper to introduce design professionals to foundation mathematical concepts that are necessary for effective development of computational methods for 3D modeling and computer graphics. Written by Rajaa Issa.

<http://www.rhino3d.com/download/rhino/5.0/EssentialMathematicsThirdEdition/>

**Generative Algorithms** A series of books which is aimed to develop different concepts in the field of Generative Algorithms and Parametric Design. Written by Zubin Khabazi. <http://www.morphogenesism.com/media.html>

**Rhino Python Primer** This primer is intended to teach programming to absolute beginners, people who have tinkered with programming a bit or expert programmers looking for a quick introduction to the methods in Rhino. Written by Skylar Tibbits. <http://www.rhino3d.com/download/IronPython/5.0/RhinoPython101>

## General References

**Wolfram MathWorld** is an online mathematics resource., assembled by Eric W. Weisstein with assistance from thousands of contributors. Since its contents first appeared online in 1995, MathWorld has emerged as a nexus of mathematical information in both the mathematics and educational communities. Its entries are extensively referenced in journals and books spanning all educational levels. <http://mathworld.wolfram.com/>

## Further Reading

Burry, Jane, and Mark Burry. *The New Mathematics of Architecture*. London: Thames & Hudson, 2010.

Burry, Mark. *Scripting Cultures: Architectural Design and Programming*. Chichester, UK: Wiley, 2011.

Hensel, Michael, Achim Menges, and Michael Weinstock. *Emergent Technologies and Design: Towards a Biological Paradigm for Architecture*. Oxon: Routledge, 2010.

Jabi, Wassim. *Parametric Design for Architecture*. Laurence King, 2013.

Menges, Achim, and Sean Ahlquist. *Computational Design Thinking*. Chichester, UK: John Wiley & Sons, 2011.

Menges, Achim. *Material Computation: Higher Integration in Morphogenetic Design*. Hoboken, NJ: Wiley, 2012.

Peters, Brady, and Xavier De Kestelier. *Computation Works: The Building of Algorithmic Thought*. Wiley, 2013.

Peters, Brady. *Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design*. Chichester: Wiley, 2013.

Pottmann, Helmut, and Dariil Bentley. *Architectural Geometry*. Exton, PA: Bentley Institute, 2007.

Sakamoto, Tomoko, and Albert Ferré. *From Control to Design: Parametric/algorithmic Architecture*. Barcelona: Actar-D, 2008.

Woodbury, Robert. *Elements of Parametric Design*. London: Routledge, 2010.

