AUTODESK.

TECHNISCHE INFORMATION BIM

Dynamo für Revit

Ausgabe Nr. 03/2016

AUTODESK TECHNICAL SPECIALISTS AEC
http://autodesk.de/bimblog

Inhalt

Allgemein

Warum Visuelle Programmierung?

In der heutigen Zeit wird das Interesse an individueller, auf die eigenen Bedürfnisse angepasster Software immer größer – dies gilt natürlich auch für BIM-Software. Der Weg führt hier in der Regel über eigene Applikationen und Erweiterungen.
Die typischen Nutzer von BIM-Software haben jedoch in den seltensten Fällen so gute Programmiererfahrungen, um selbst in eine Programmierschnittstelle einzugreifen und mit Programmiersprachen wie C# ihre eigenen Werkzeuge zu entwickeln.

Visuelle Programmierung erfordert keine Programmiererfahrung und ermöglicht die Erstellung von eigenen Skripts bereits nach einer kurzen Einarbeitungsphase, wobei eine analytische Denkweise und eine grundlegende Programmiererfahrung natürlich durchaus von Vorteil sind. Es gilt allerdings auch: jeder, der Interesse mitbringt und sein Problem logisch beschreiben kann, kann es höchstwahrscheinlich mit Dynamo lösen.
Durch das Verknüpfen von vorgefertigten Blöcken wird in Dynamo der eigentliche Programmcode im Hintergrund erzeugt.

Dynamo ist ein **openSource Projekt** – dies bedeutet, dass der Programmcode frei verfügbar ist und jeder zur Weiterentwicklung beitragen kann.
Dynamo ist aktuell als kostenfreies Revit-Plugin oder auch als ein eigenständiges Programm, Dynamo Studio, erhältlich.

Installation

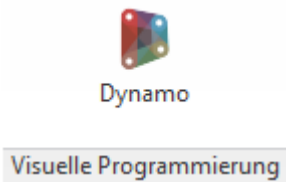
Bei Revit 2016 R2 oder später ist eine zusätzliche Installation von Dynamo für Revit nicht mehr notwendig. Dynamo ist ab dieser Version standardmäßig vorinstalliert und muss nur bei Bedarf aktualisiert werden. Hierzu erfolgt eine Einladung direkt im Dynamo Fenster:

Besitzer einer Revit 2015- oder 2016-Version können Dynamo für Revit unter http://dynamobim.org/download/ kostenlos herunterladen.

Dynamo für Revit

Erste Schritte

Benutzer von Revit 2017 finden Dynamo unter dem Tab *Verwalten*. In Revit 2015 und 2016 befindet sich Revit unter dem Tab *Zusatzmodule*.



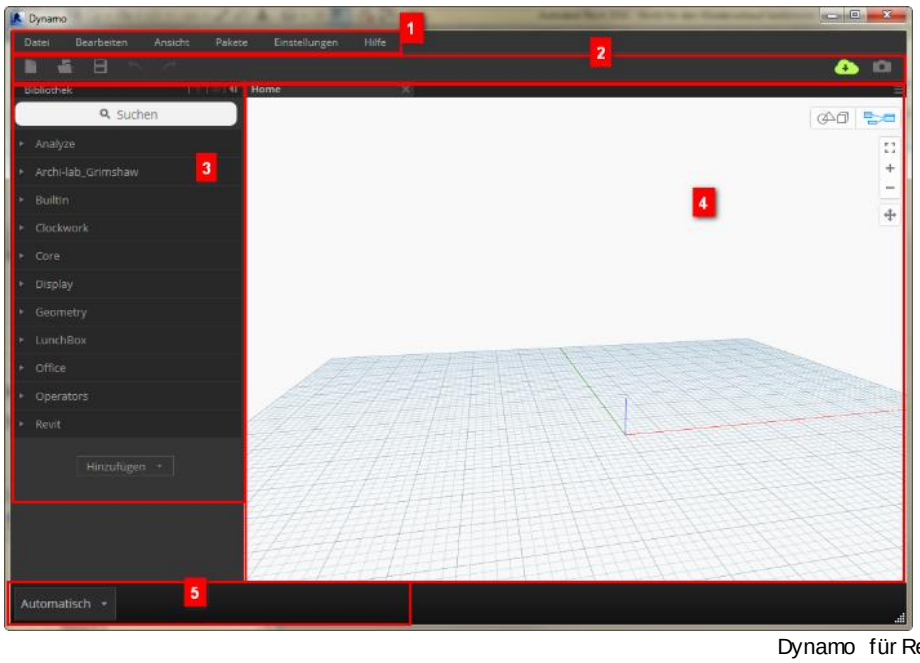
Nach einem Klick auf Dynamo öffnet sich ein neues Fenster. Hier können zuletzt geöffnete, neue oder benutzerdefinierte Skripte geöffnet werden. Beim Wechsel zwischen Dynamo und Revit ist es nicht nötig, das Dynamo Fenster zu schließen – dDynamo kann im Hintergrund laufen und während des Arbeitens in Revit einfach minimiert werden.

Skripte besitzen die Dateiendung .dyn, während vordefinierte Blöcke mit .dyf enden. Im Kapitel „Professionelles Arbeiten“ wird genauer auf vordefinierte Blöcke eingegangen.

Die Benutzeroberfläche

Die Oberfläche von Dynamo ist in 5 Bereiche aufgeteilt.

1. Menüleiste
2. Werkzeugleiste
3. Bibliothek
4. Arbeitsebene
5. Ausführungsleiste



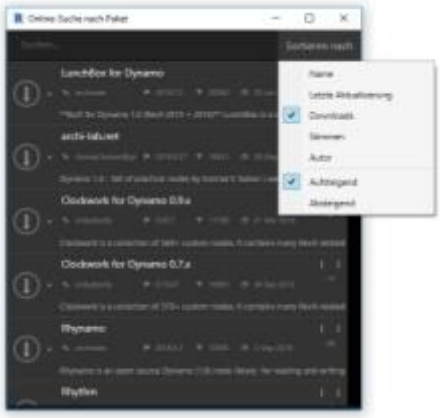
Die Tabs in der **Menüleiste** enthalten neben üblichen Funktionen wie *Neu*, *Speichern*, *Kopieren* und *Importieren* zusätzlich auch die Möglichkeit, das Arbeitsfeld als Bild oder das erstellte Modell als STL-Datei zu exportieren.

Unter *Bearbeiten* können Blöcke übersichtlich in Gruppen und Blockgruppen zusammengefasst und Kommentare eingefügt werden.

Ansicht enthält Navigations- und Darstellungsfunktionen. In diesem Tab wird im Hintergrund der Arbeitsebene eine 3D-Vorschau oder die Revit-Vorschau ein- und ausblendet. Die Hintergrundvorschau in Dynamo sollte hier standardmäßig ausgewählt sein.

Im *Pake*te-Tab können zusätzliche Pakete heruntergeladen werden.

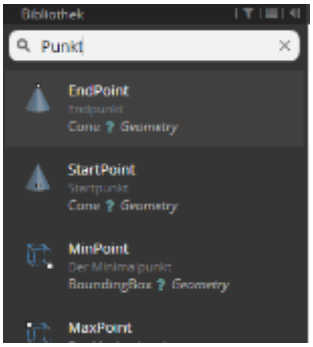
Pakete enthalten eine Reihe von Blöcken zu einem bestimmten Thema, die innerhalb der Dynamo Community ausgetauscht werden. Zu den am meisten verwendeten Paketen gehören *Clockwork*, *Lunchbox*, *Archi-lab_Grimshaw und SpringNodes.*



Es empfiehlt sich, die Pakete bei der Suche nach den Downloads zu sortieren, so tauchen die populärsten Pakete automatisch ganz oben auf.

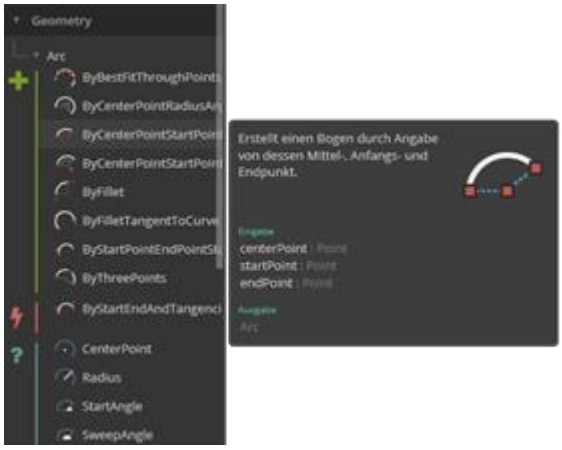
Für Anfänger befinden sich im Tab *Hilfe* kurze Beispiele, die Schritt für Schritt die Bedienung mit Blöcken und einfache Skripte zeigen.

Die **Bibliothek** besitzt eine Suchleiste, über die nach Blöcken gesucht werden kann.
Bei der Suche werden nicht nur die Namen der Blöcke, sondern auch Beschreibungen berücksichtigt. Ist der genaue Name und der Aufenthaltsort des Blocks nicht bekannt, kann dieser durch die Eingabe englischer und teilweise deutscher Schlüsselwörter gefunden werden.
In der Dynamo Grundeinstellung sind Blöcke in acht Kategorien und weiteren Unterkategorien unterteilt. Heruntergeladene Pakete werden ebenfalls hier aufgelistet.



Je nach Funktion sind in den Unterkategorien Blöcke in drei Eigenschaften unterteilt:

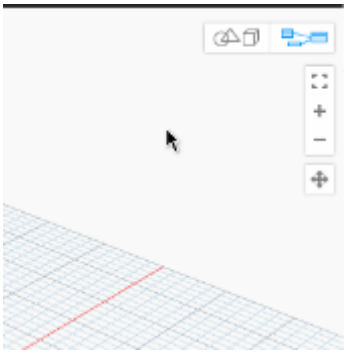
Erstellen Aktion Abfrage



Wird der Mauszeiger über einem Element platziert und gehalten, wird als Hilfestellung die Funktionsweise des Blocks genauer beschrieben.

Zusätzlich werden hier auch die Eingaben und Ausgaben des Blocks und die zugehörigen Datentypen aufgelistet.

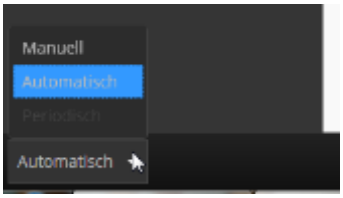
Die am häufigsten verwendeten Kategorien sind *Core*, *Geometry* und *Revit*. In der **Arbeitsebene** von Dynamo kann zwischen einer Diagrammansicht und der Hintergrund-3D-Vorschau gewählt werden. Die Diagrammansicht ist eine 2D-Umgebung, in der Blöcke abgelegt und miteinander verbunden werden können. Die Hintergrund 3D-Vorschau stellt 3D-Abbildungen von allen Blöcken dar, die in der Diagrammansicht eine Geometrie beschreiben.



Mit den Navigationsfunktionen in der oberen rechten Ecke der Arbeitsebene kann zwischen den beiden Ansichten navigiert werden.
Noch einfacher kann durch das Drücken und Halten der ESC Taste im Hintergrund navigiert werden.
Die Navigation mit der mittleren Maustaste (Drehen) und dem Scrollen (Zoomen) erfolgt in beiden Ansichten gleich.

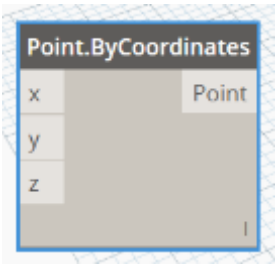
In der Diagrammansicht können mit der linken Maustaste Blöcke ausgewählt, mit der rechten Maustaste das Kontextmenü geöffnet und mit einem Doppelklick auf den Hintergrund ein sogenannter *Code Block* erstellt werden. Im Kontextmenü befindet sich u.a. die aus der Bibliothek bekannte Suchfunktion.

Die **Ausführungsleiste** hat zwei Funktionen: manuell und automatisch. Letzte Option führt das Skript nach jeder Änderung automatisch aus. Bei komplexen und rechenintensiven Skripten empfiehlt es sich daher, den manuellen Modus zu wählen.



Blöcke verbinden

Um Skripte zu erstellen, müssen Blöcke miteinander verbunden werden. Dafür ist es zunächst einmal notwendig zu verstehen, wie ein Block aufgebaut ist.

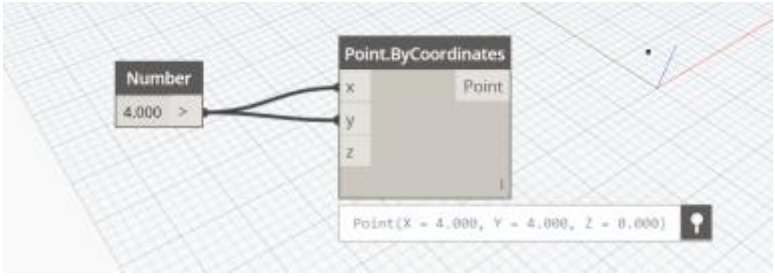


Ein **typischer Block** ist z.B. *Point.ByCoordinates*. Dieser befindet sich in der Bibliothek unter *Geometry → Point → ByCoordinates (x, y, z)*.

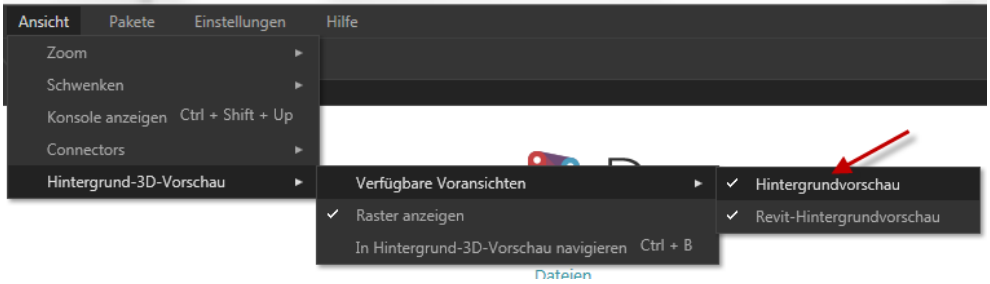
Auf der linken Seite eines Blocks befinden sich die Eingabevariablen, die bestimmte Werte als Eingabe erwarten. Diese Werte werden verarbeitet und in der Ausgabevariablen auf der rechten Seite ausgegeben.
In diesem Beispiel muss also an x, y und z jeweils eine Zahl übergeben werden, damit ein Punkt im 3D-Raum erzeugt werden kann.

Für die Eingabe von Zahlen eignet sich der Block *Number*.

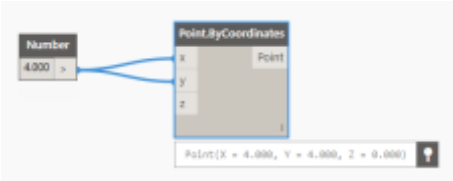
Zum **Erstellen einer Verbindung** zwischen den beiden Blöcken kann die Ausgabe des Number Blocks mit beliebig vielen Eingabevariablen verbunden werden.



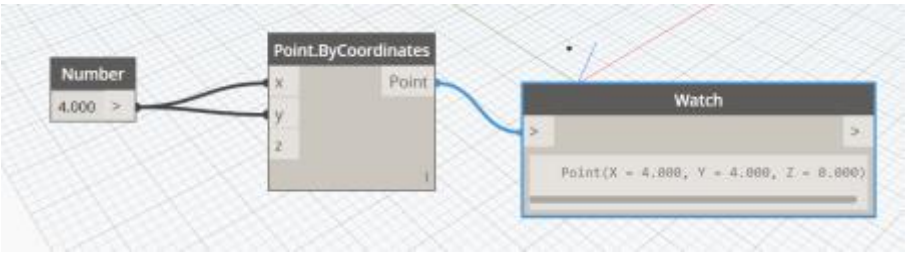
Falls keine 3D Vorschau im Hintergrund zu sehen ist, muss diese in der Menüleiste unter → *Ansicht* → *Hintergrund-3D-Vorschau* → *Verfügbare Ansichten* aktiviert werden:



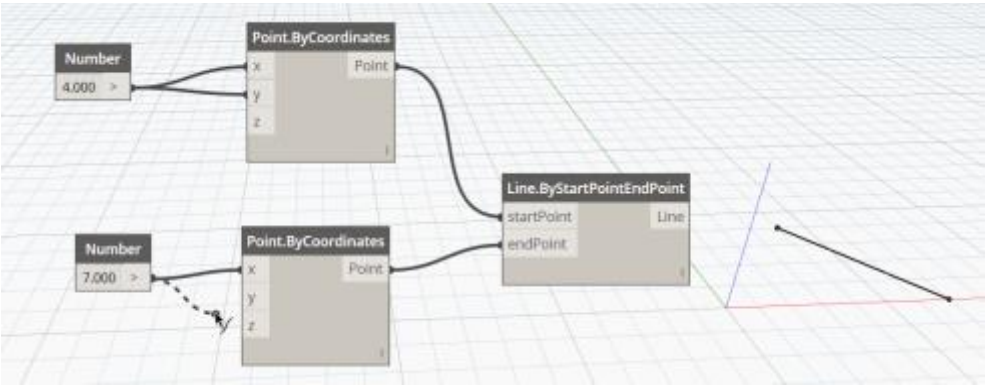
Wenn Sie nun mit der Maus über die rechte untere Ecke des Punkt-Blocks fahren, erscheint sein Ausgabewert. Klicken Sie auf das Symbol auf der rechten Seite, welches an einen Pin erinnert, um die **Ausgabe dauerhaft sichtbar** zu machen.



Alternativ kann auch ein *Watch*-Block verwendet werden:

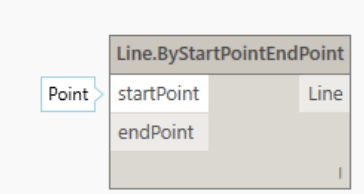


Der *Line.ByStartPointEndPoint*-Block aus der Kategorie *Geometry* → *Line* benötigt zwei verschiedene Punkte, also Werte vom Typ „Point“. Diese können mit den Ausgabevariablen des *Point.ByCoordinates*-Blocks verbunden werden.
Durch *Strg-C* + Strg-V können ausgewählte Blöcke kopiert werden. Werden zwei unterschiedliche Punkte an den *Line.ByStartPointEndPoint*-Block gegeben, erzeugt dies eine Linie zwischen den beiden Punkten:

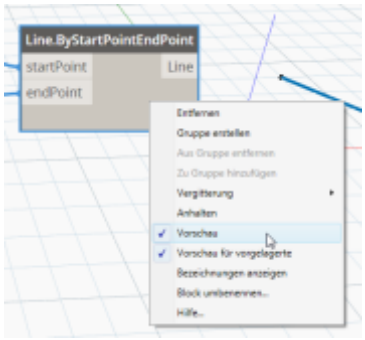


Zum **Entfernen einer Verbindung** wird auf das Ende einer Verbindung (die Ausgabevariable) und danach auf eine freie Stelle in der Arbeitsebene geklickt.

Bei der Zuweisung von Werten an eine Eingabevariable ist der richtige **Dateityp** zu beachten. Diese werden sichtbar, wenn die Maus über die Variable platziert wird:



Beim Auswählen eines Blocks wird in der Hintergrund-3D-Vorschau das entsprechende geometrische Element blau markiert.

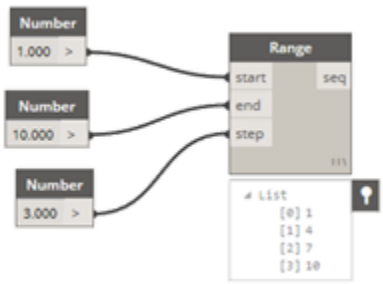


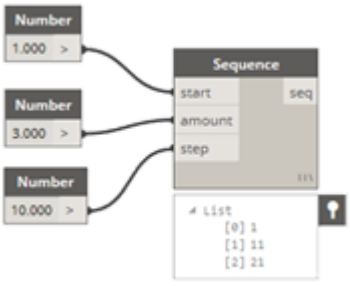
Einzelne **Elemente** können in der Hintergrund-3D-Vorschau **ausgeblendet** über das Kontextmenu ausgeblendet werden.

Listen

Listen sind wichtige Elemente in Programmiersprachen, da mit ihnen gleichartig strukturierte Daten verarbeitet werden können.

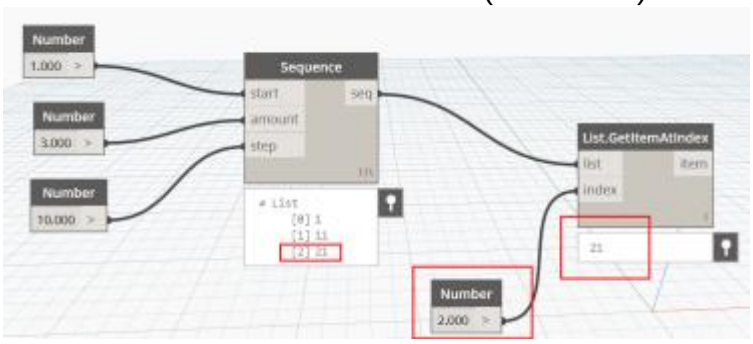
Listen aus Zahlen können auf verschiedenen Wegen erstellet werden:





Der *Range*-Block (*Core → List*) erstellt die Zahlenreihe mithilfe einer Unter- und Obergrenze sowie einer Schrittweite.
Beim *Sequence*-Block hingegen wird der Startwert und die Schrittweite (*step*) ermittelt, sowie die Anzahl der zu ermittelenden Werten bestimmt durch *amount* (Menge).

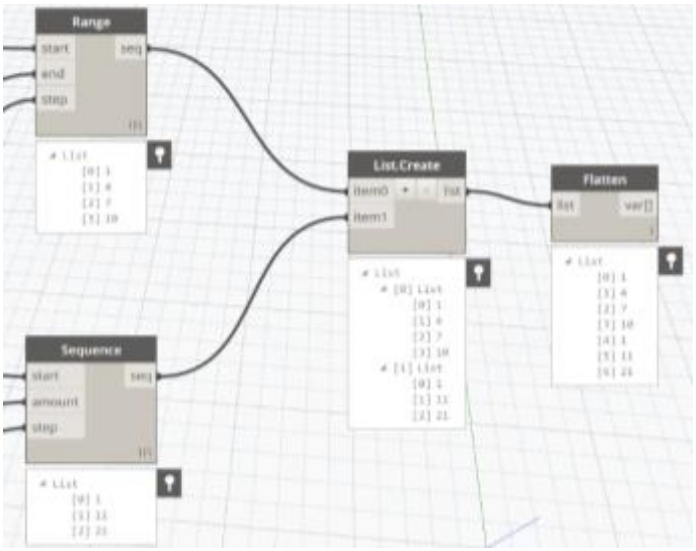
Um ein bestimmtes **Element aus einer Liste** aufzurufen, wird dessen Stelle (Index) in der Liste benötigt. Dieser kann mit dem *List.GetItemAtIndex*-Block aus der *Core*-Kategorie aufgerufen werden. Hierbei ist zu beachten, dass wie bei den meisten Programmiersprachen das erste Element einer Liste den Index 0 hat (und nicht 1).



Listen können allerdings nicht nur aus Zahlen bestehen, sondern auch aus anderen Datentypen.

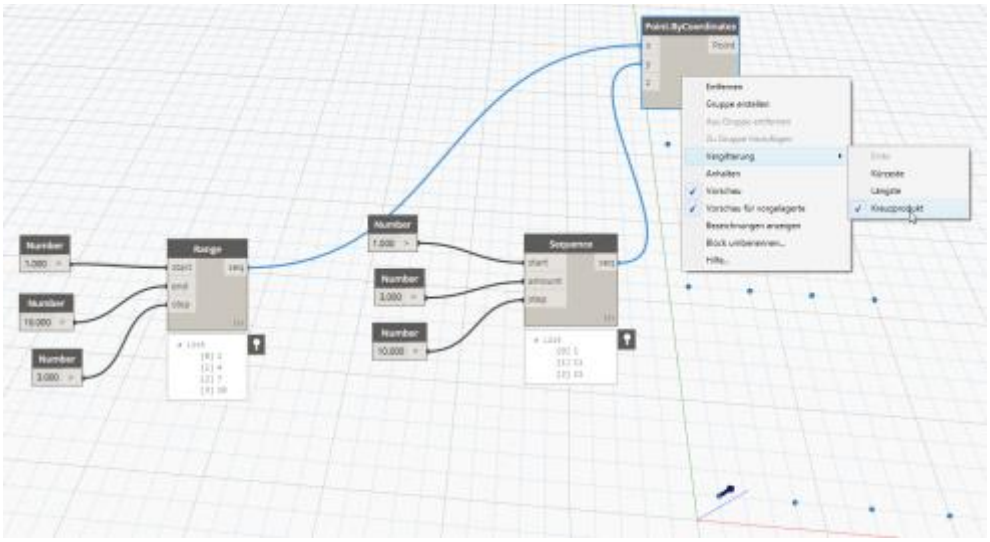
Mit ***List.Create*** können solche Listen erstellt werden, dabei kann die Anzahl der Eingabevariablen frei definiert werden. Durch das Klicken auf das + Symbol in dem Block werden zusätzliche Eingabevariablen angelegt. Diese können mit der – Funktion wieder entfernt werden.

Als Eingabe können entweder einzelne Elemente oder ganze Listen genutzt werden – im letzten Fall sind das Ergebnis **verschachtelte Listen.** Solche Listen werden oft bei der Interaktion mit Tabellen verwendet, da hier jede Unterliste eine Spalte definiert.
Mit dem *Flatten*-Block verschachtelte Listen zu einer einfachen Liste vereinfacht werden, was oft für die Durchführung bestimmter Berechnungen notwendig ist.



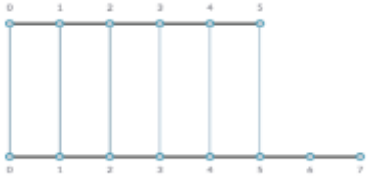
Beispiel: Punktegitter

Zum **Erstellen eines Punktegitters** werden zwei Zahlenlisten für die Definiton der X und Y Werte benötigt – wahlweise könnte natürlich auch eine Z-Variable definiert werden. In diesem Beispiel werden die die Listen aus dem *Range*- und dem *Sequence*-Block genutzt und mit den X und Y Variablen des *Point.ByCoordinates*-Blocks verbunden.
Anschließend ist es wichtig, die Vergitterung des Blocks zu definieren – diese Bezeichnet die Logik, nach der die beiden Listen miteinander verbunden werden. Im Kontextmenu des Blocks kann *Vergitterung* → *Kreuzprodukt* gewählt werden.

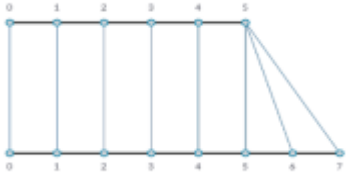


Folgende Grafik zeigt, was die Unterschiede zwischen den einzelnen Vergitterungsarten sind:

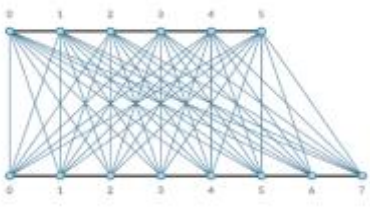
Vergitterung: Kürzeste



Vergitterung: Längste



Vergitterung: Kreuzprodukt

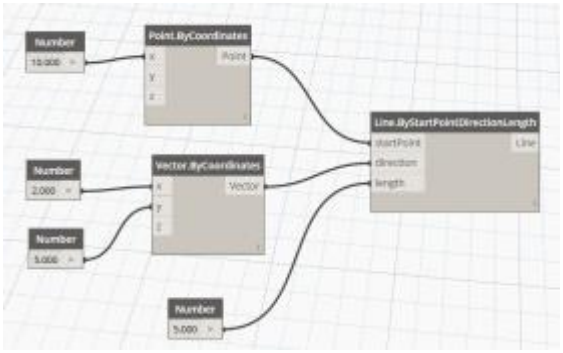


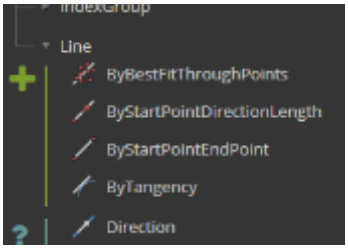
Erzeugen von Geometrien

2D-Geometrie

Dynamo bietet verschiedene Blöcke für die Erstellung von Geometrie an, zu finden in der entsprechenden Kategorie in der Bibliothek.

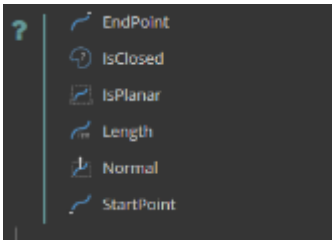
Eine Linie kann dabei nicht nur mit Hilfe von Start- und Endpunkt beschrieben werden, sondern auch mit Hilfe der Blöcke: *ByStartPointDirectionLength*, *ByTangency* und *ByBestFitThroughPoints*.



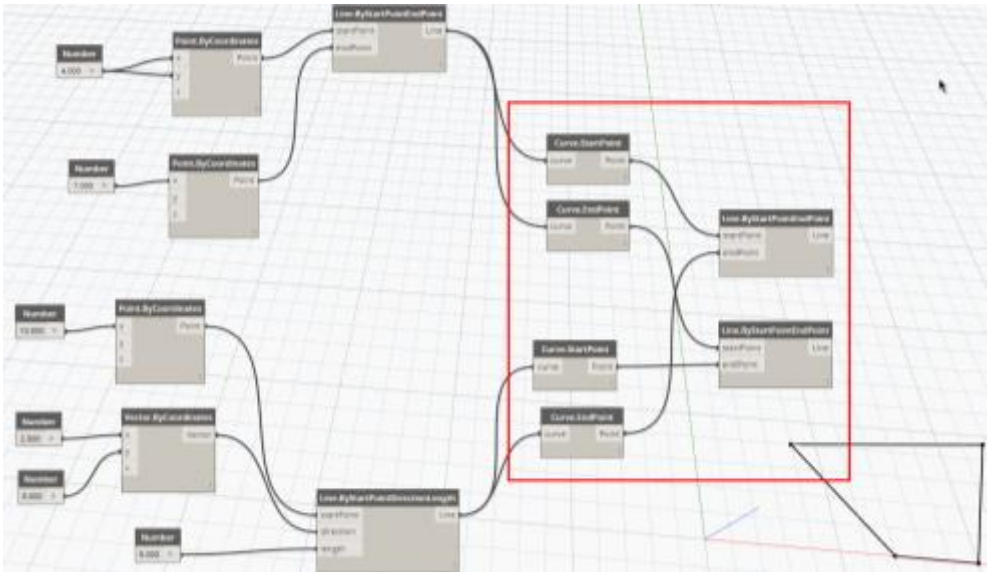


Line.ByStartPointDirectionLength erwartet drei Variablen erwartet: einen Startpunkt, eine Richtung und eine Länge. An die erste Variable wird ein definierter Punkt übergeben. Als Richtung muss ein **Vektor** vorgegeben werden.
Die *Vector*-Blöcke befinden sich ebenfalls in der *Geometry*-Kategorie unter *Vector*. Ihr Aufbau ähnelt zwar dem der Punkt-Blöcke, jedoch beschreiben Vektoren nur Richtungen und Pfade und sind deshalb auch nicht in der Hintergrundvorschau sichtbar.

Mit Dynamo können allerdings nicht nur Elemente erzeugt werden, sondern auch Abfragen gestellt werden, z.B.: „*Wie lang ist eine Kurve?*“, „*Welchen Richtungsvektor besitzt eine Linie?*“ oder „*Ist eine Kurve geschlossen?*“



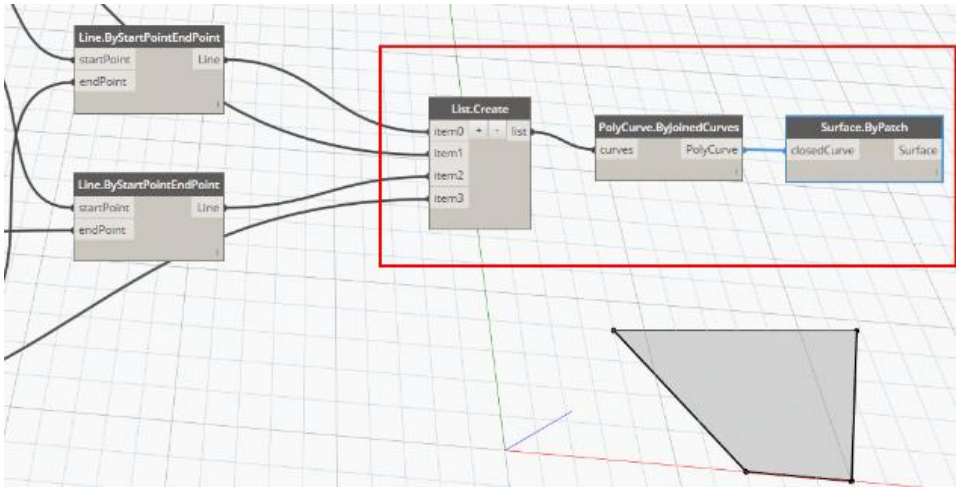
Im folgenden Beispielen wird ein Viereck erstellt, indem die Start- und Endpunkte zweier im ersten Schritt erstellten Kurven jeweils miteinander verbunden werden.
Diese können mit den Abfragen *StartPoint* und *EndPoint (Geometry → Curve)* ermittelt werden. Die Ausgabe dieser Blöcke ist jeweils ein Punkt, zwischen denen mit dem *Line.ByStartPointEndPoint*-Block neue Linien erstellet werden können.



Im nächsten Schritt soll eine **Oberfläche** mit den vier Linien als Grenzen gespannt werden. Dafür wird der *Surface.ByPatch*-Block aus der Kategorie *Geometry → Surface* verwendet. Dieser erwartet eine geschlossene Kurve als Eingabe.

In diesem Fall sind die vier Linien zwar optisch geschlossen, bestehen jedoch aus mehreren Kurven, die erst zu einer verbunden werden müssen. Dies geschieht mit Hilfe einer **Aktion** – wird in dem Suchfeld „join“ eingegeben, kommen mehrere Ergebnisse als Resultat. In diesem Fall eignet sich der Block *ByJoinedCurves*. Dieser Block empfängt Kurven vom Typ „Curve[]“. Die rechteckigen Klammern sind ein Zeichen dafür, dass der Block eine Liste aus Kurven erwartet.
Mit einem *List.Create*-Block können verschiedene Linien in einer Liste gesammelt werden, um anschließend an den *PolyCurve.ByJoinedCurves*-Block übergeben zu werden.

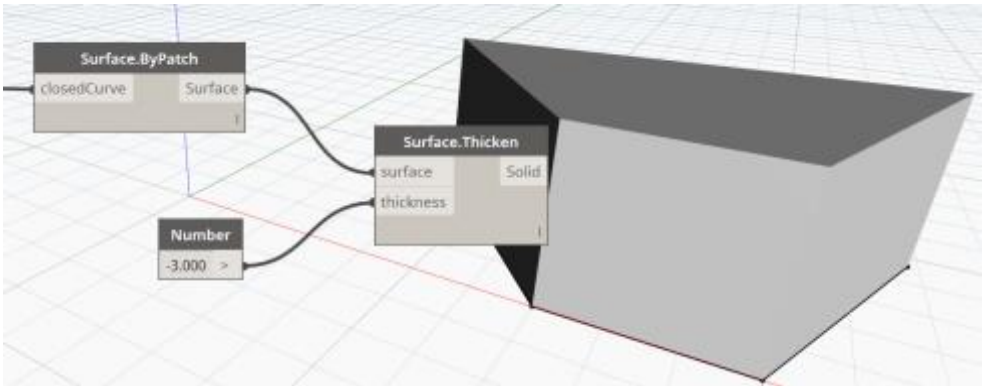
Die Ausgabe *PolyCurve* ist eine Kurve und kann schließlich mit dem *Surface.ByPatch*-Block verbunden werden:



3D-Geometrie

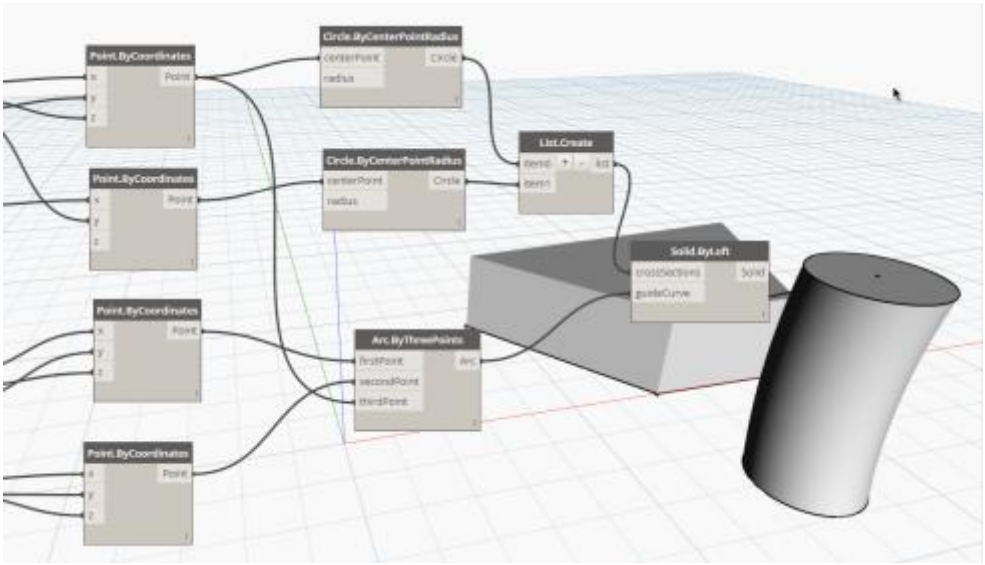
3D-Körper können in Dynamo direkt erzeugt werden – beispielsweise ein Zylinder (*Geometry → Cylinder*) oder ein Würfel (*Geometry → Cuboid*). Für komplexere Formen ist jedoch der Weg über Flächen oder Kurven unvermeidbar.

Die **Extrusion einer Fläche** kann mit dem *Surface.Thicken*-Block erzeugt werden. Dazu wird der Block mit einer fertigen Oberfläche und einem Zahlen-Block verbunden. Je nach Richtung der Oberflächennormale muss positive oder negative Zahl als *thickness* eingeben werden, um in die gewünschte Richtung zu extrudieren.



Für schwierige geometrische Körper eignet sich der Einsatz von Kurven. Unter *Geometry* → *Solid* finden sich unterschiedliche Möglichkeiten, **Körper aus Kurven** zu erstellen.

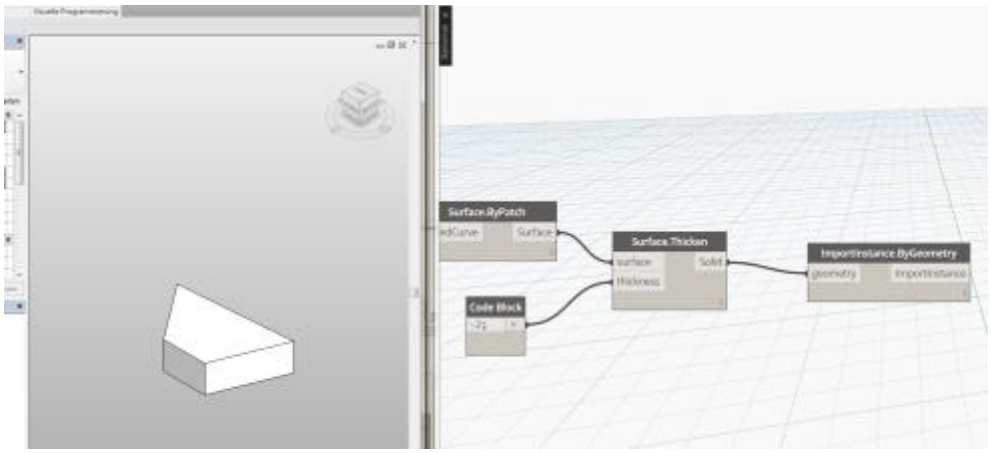
Die aus Revit bekannte Lofting-Funktion (Erhebungsform) lässt sich mit dem *Solid.ByLoft*-Block abbilden. Der Block besitzt zwei Eingabevariablen. Die erste erwartet eine Liste mit mindestens der geschlossenen Start- und Endkurve (es können auch Zwischenprofile vorgegeben werden, um komplexere Formen zu beschreiben), während in der zweiten Eingabevariable der Pfad über eine Kurve erstellt wird.



Interaktion mit Revit

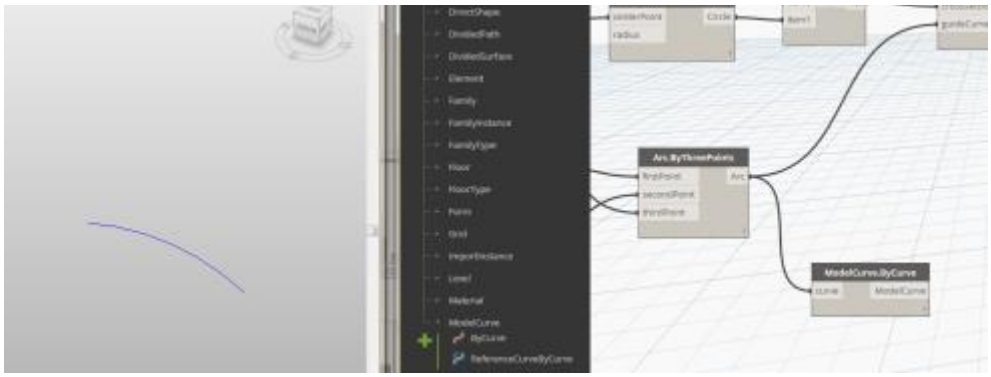
Geometrie nach Revit exportieren

Die in Dynamo erstellten **Körper** können natürlich anschließend nach **exportiert** werden. Hierzu dient der unter *Revit → ImportInstance* zu findende *ImportInstance.ByGeometry*-Block:



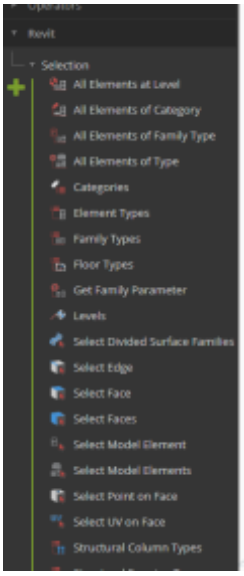
2D-Elemente wie Kurven müssen allerdings anders behandelt werden. Dynamo-Kurven können zu **Modelllinien** in Revit umgewandelt werden. Der *ModelCurve.ByCurve*-Block findet Sie unter *Revit → ModelCurve*.

Falls die Kurve nicht in ein Revit-Projekt sondern in den Familieneditor importiert wird, kann diese wahlweise als Modellinie oder Referenzkurve importiert werden.

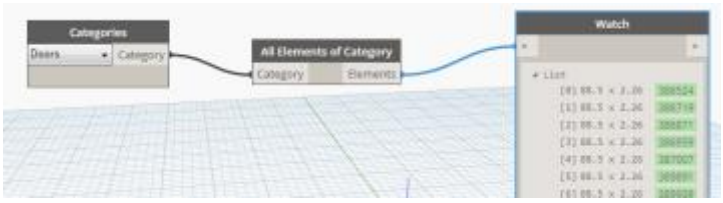


Revit-Elemente importieren

Mit Dynamo kann nicht nur Geometrie in Revit erzeugt werden, sondern auch diverse Abfragen durchgeführt werden. Hierzu werden i.d.R. Elemente im Revit-Dokument ausgewählt. Die Blöcke zum Aufrufen dieser Daten finden sich unter der Kategorie *Revit* → *Selection*.

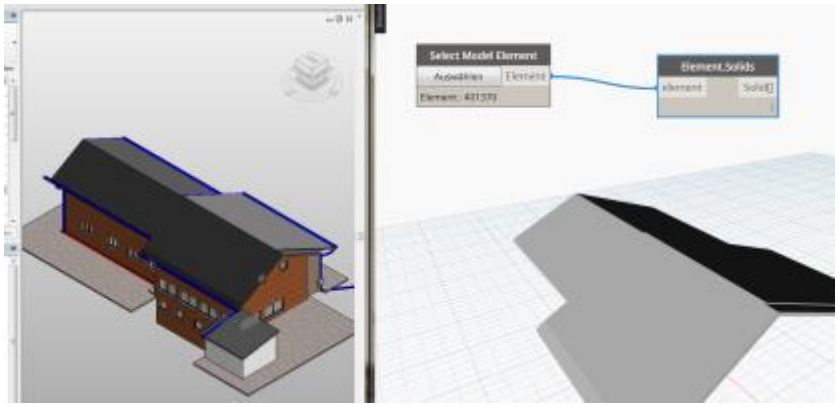


Typischerweise werden im ersten Schritt **alle Elemente** einer Ebene, einer Kategorie, eines Familientyps oder eines Elementtyps ausgewählt:



Alternativ können natürlich auch **einzelne Elemente** oder bestimmte Geometrien eines Elements aus dem Revit-Projekt nach Dynamo importiert werden.
Dies geschieht mit dem *Select-Model-Element*-Block: dieser Block besitzt einen Button, mit dem ein Element direkt ausgewählt werden kann. Falls mehrere Elemente ausgewählt werden, kann auch der Block *Select-Model-Element**s*** genutzt werden – hierbei muss anschließend in Revit ein Auswahlfenster aufgezogen werden.

Um mit der **Geometrie des Modellelements** in Dynamo weiterzuarbeiten, muss es in Dynamo-Geometrie umgewandelt werden. Für die Umwandlung von 3D Geometrie dient der *Element.Geometry*-Block (bzw. alternativ Element.Solid für 3D-Geometire bzw. Element.Curve für Kurven).



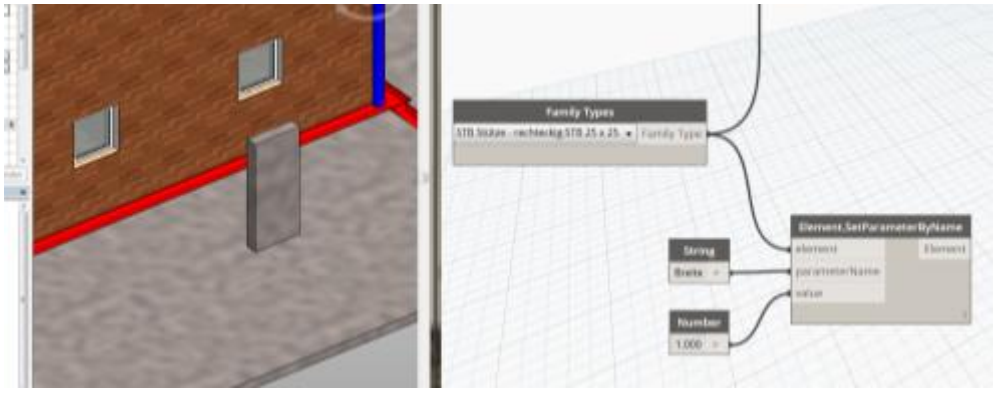
Revit-Familien

Mit Dynamo lassen sich auch **Parameter von Revit-Familien** aufrufen und bearbeiten.

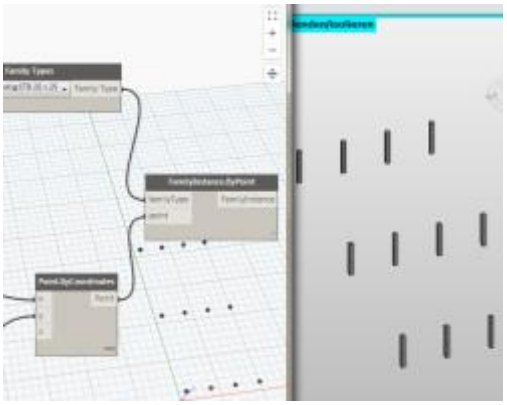
Dazu muss zunächst eine Familie mit dem *Family-Types*-Block ausgewählt werden. Mit dem *Get-Family-Parameter*-Block können anschließend **alle verfügbaren Parameter** in dieser Familie abgerufen werden.

Mit dem *Element.SetParameterByName*-Block kann der **Wert eines Parameters bearbeitet** werden, während der Block *Element.GetParameterValueByName* der **Wert ausgelesen werden** kann.

Dynamo ermöglicht gleichermaßen das Ändern von Typ- und Exemplarparametern.



Dynamo kann nicht nur Daten aus Revit-Familien auslesen, sondern auch **Familieninstanzen an Punkten platzieren**. Hierfür dient der Block *FamilyInstance.ByPoint* (Revit → Elements). Mit dem *FamilyInstance.SetRotation*-Block können Sie optional eine **Drehung der Familien** um einen bestimmten Winkel um die Z-Achse ausführen.

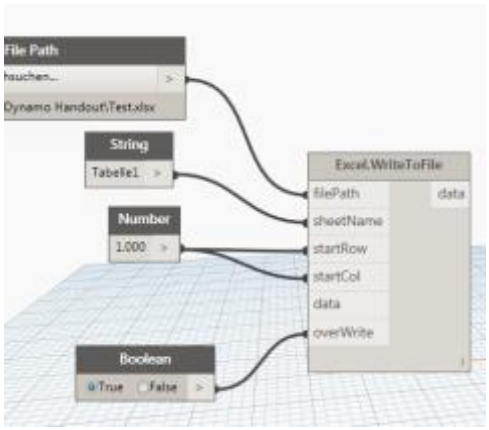


Ebenfalls können auch komplexe adaptive Komponenten mit Hilfe von Punktgruppen platziert werden. Ausführliche Anleitungen finden sich unter: http://dynamobim.org/learn/

Arbeiten mit Excel

Dynamo ermöglicht es, Daten aus Excel-Tabellen direkt zu exportieren bzw. zu importieren.

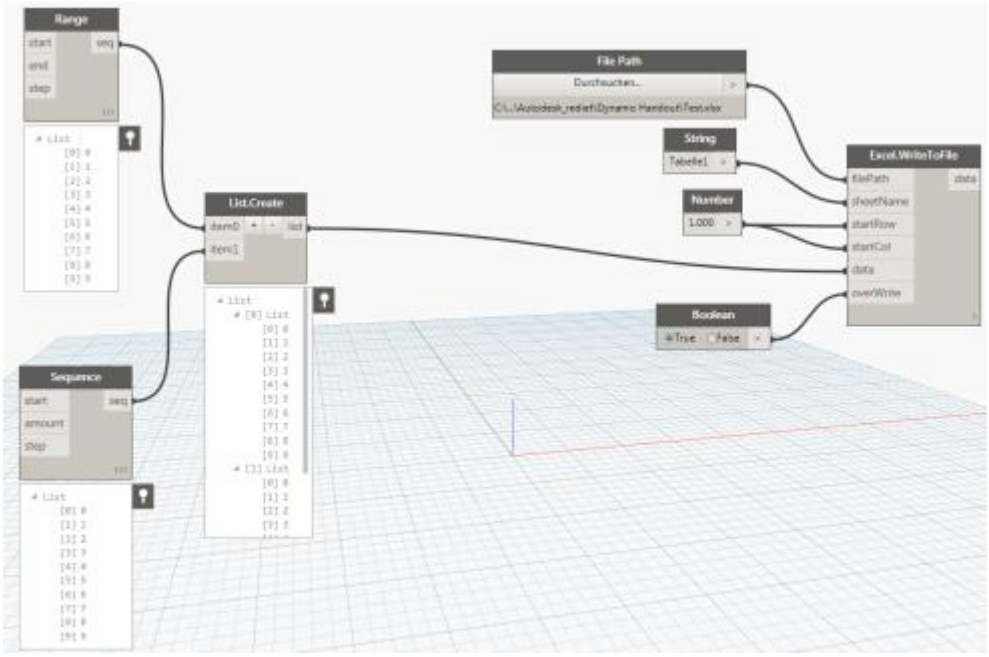
Export



Für den Export einer Excel-Tabelle wird der *Excel.WriteToFile*-Block genutzt. Dieser benötigt den Pfad zur Excel-Datei, der mit dem *FilePath*-Block gewählt werden kann.
Die Eingabe *sheetName* benötigt den Namen der Tabelle und *startRow* und *startCol* die Startnummern (Reihe und Zeile) der Tabelle – beide werden mit Zahlenangaben beginnend mit 0 definiert. Hierbei steht startRow 0 für die Zeile 1, startRow 1 für die Zeile 2, startCol 0 für die Spalte A, startCol 1 für die Spalte B usw.
Die letzte Eingabe *overWrite* ist eine boolesche Variable und will wissen, ob der gesamte Inhalt der Excel-Tabelle vor dem Beschreiben geleert werden soll. Sie erwartet daher die Aussagen wahr (*true*) oder falsch (*false, Vorgabewert*). Falls Sie hier keine Angabe machen, wird der Vorgabewert false genutzt, es werden also nur die definierten Zellen beschrieben.

An *data* übergeben wird eine zweidimensionale, alo einfach verschachtelte, Liste übergeben. Hierbei ist zu beachten, dass jede Unterliste eine Zeile definiert. I.d.R. müssen die mit Dynamo erstellten Listen mit *List.Transpose*-Block transponiert werden, um dieser Logik zu entsprechen.

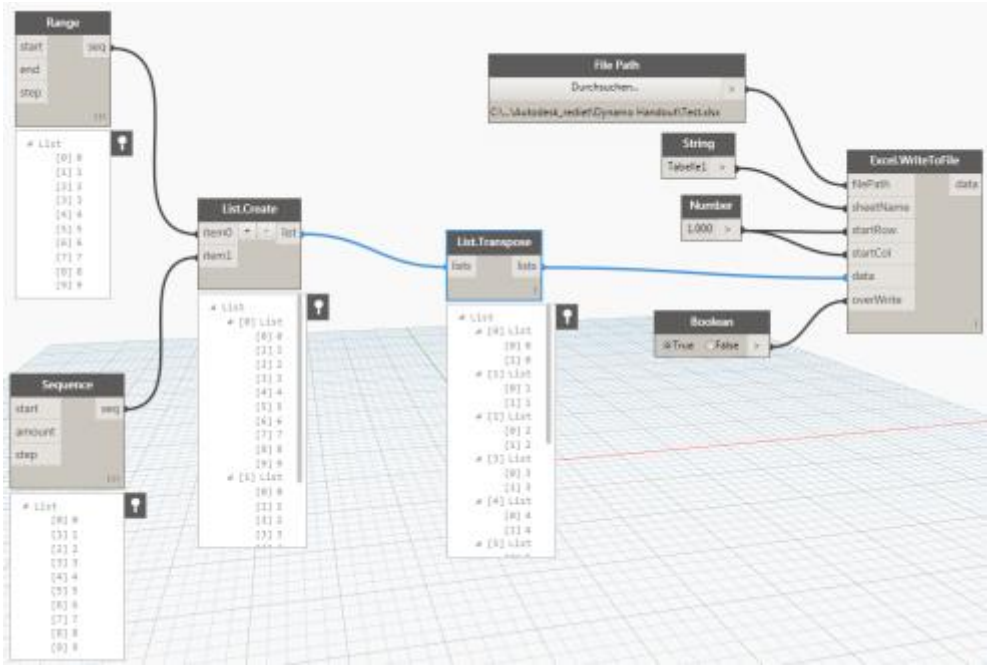
List.Transpose Beispiel 1:



Ausgabe in Excel:

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		0	1	2	3	4	5	6	7	8	9	
3		0	1	2	3	4	5	6	7	8	9	
4												

List.Transpose Beispiel 2:



Ausgabe in Excel:

	A	B	C	D
1				
2		0	0	
3		1	1	
4		2	2	
5		3	3	
6		4	4	
7		5	5	
8		6	6	
9		7	7	
10		8	8	
11		9	9	
12				

Für weitere Bearbeitung von Listen bietet Dynamo unter *Core → List* weitere hilfreiche Blöcke an, wie z.B. *AddItemToFront.*

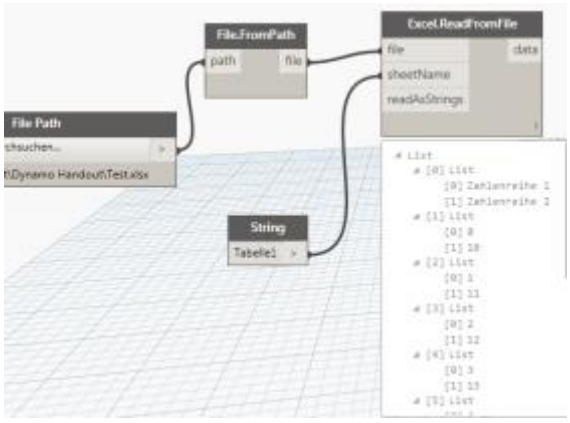
Import

Für das Einlesen von Excel-Tabellen wird der *Excel.ReadFromFile*-Block verwendet. Dieser befindet sich in der Bibliothek unter Office → Excel.

Im Gegensatz zum *Excel.WriteToFile*-Block benötigt dieser Block nicht den Pfad der Datei, sondern die Datei, weswegen zusätzlich der *File.FromPath*-Block verwendet wird. Der Name des Tabellenblatts wird durch den *String*-Block definiert.

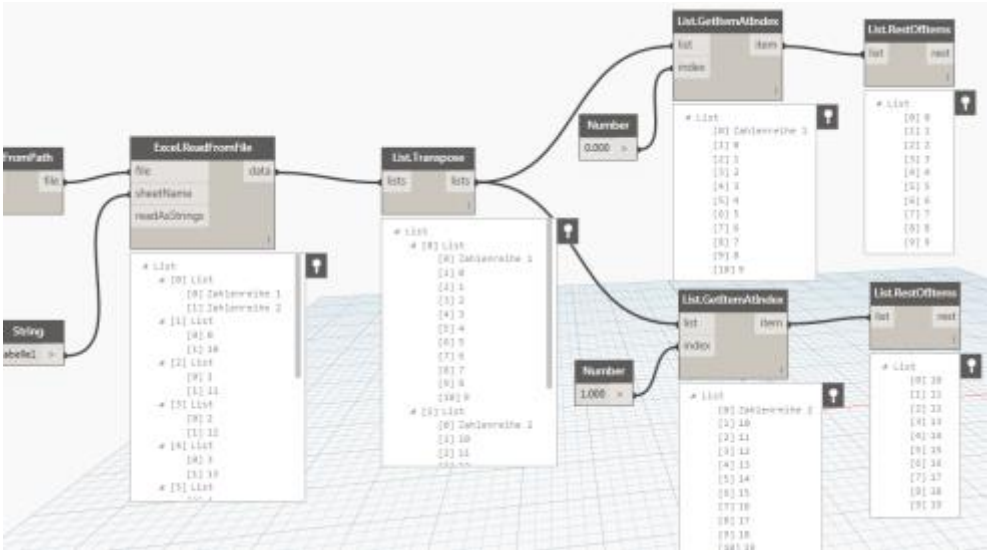
	A	B	C
1		Zahlenreihe 1	Zahlenreihe 2
2		0	10
3		1	11
4		2	12
5		3	13
6		4	14
7		5	15
8		6	16
9		7	17
10		8	18
11		9	19

→



Unter Umständen muss auch in diesem Fall die Liste vor der Weiternutzung in Dynamo mit Hilfe von *List.Transpose* umformuliert werden.

Mit dem *List.GetItemAtIndex*-Block können anschließend die Unterlisten gezielt ausgelesen werden.
Enthält die Excel-Tabelle am Anfang der Liste eine Beschreibung, die in Dynamo nicht benötigt wird, kann diese mit *List.RestOfItem*-Block entfernt werden:

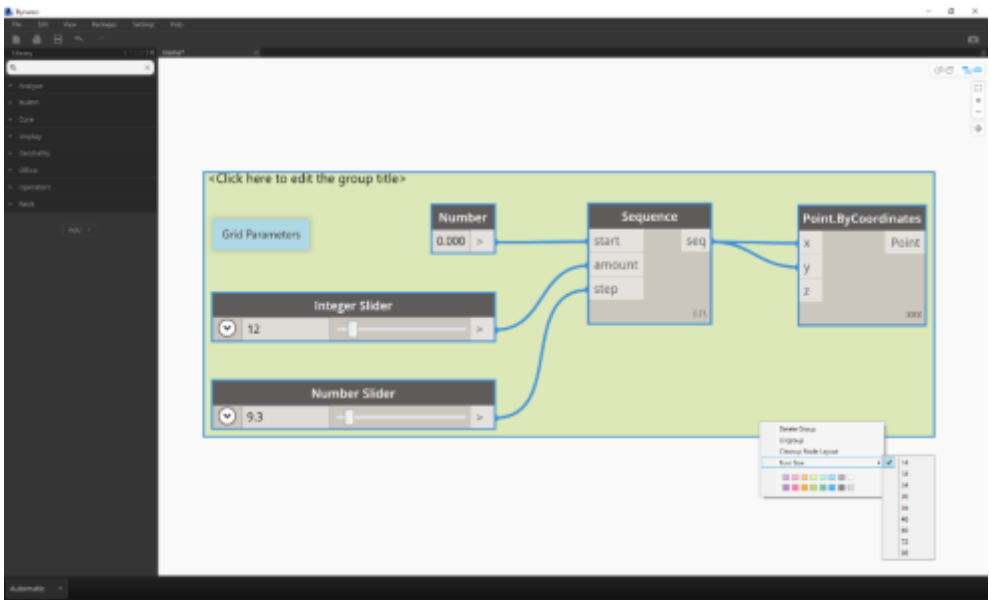


Skripte verwalten

Skripte sollten immer übersichtlich und verständlich sein, was durch die zunehmende Komplexität immer schwieriger wird. Daher bietet Dynamo zwei wichtige Funktionen an: Gruppen und Anmerkungen.

Zum **Erstellen einer Gruppe** werden die zu gruppierende Blöcke markiert und anschließend mit der rechten Maustaste auf die Arbeitsebene (nicht die Blöcke!) eine *Gruppe erstellt*.
Die Blöcke werden nun mit einer Farbfläche hinterlegt. Mit einem Doppelklick auf die Beschreibung kann der Name der Gruppe festgelegt werden. Mit einem Rechts-Klick auf die Gruppe können die Schriftgröße der Beschreibung geändert bzw. eine individuelle Farbe gewählt werden.

Oft ist auch das **Erstellen einer Anmerkung** hilfreich, um bestimmte Schritte zum späteren Zeitpunkt nachzuvollziehen oder zu erklären. Dieser Befehl befindet sich in der Menüleiste unter *Bearbeiten → Anmerkung erstellen* (Tastaturkürzel Strg W).
Die Anmerkungen sind frei bewegliche, blau hinterlegte Felder, die ebenfalls in die Gruppen eingebunden werden können.



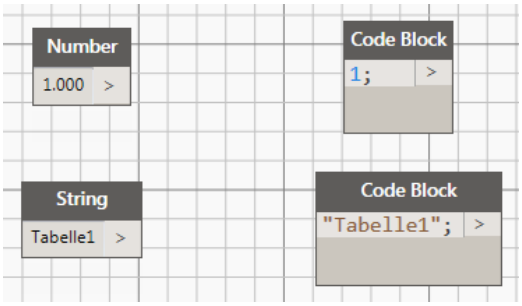
Professionelles Arbeiten mit Dynamo

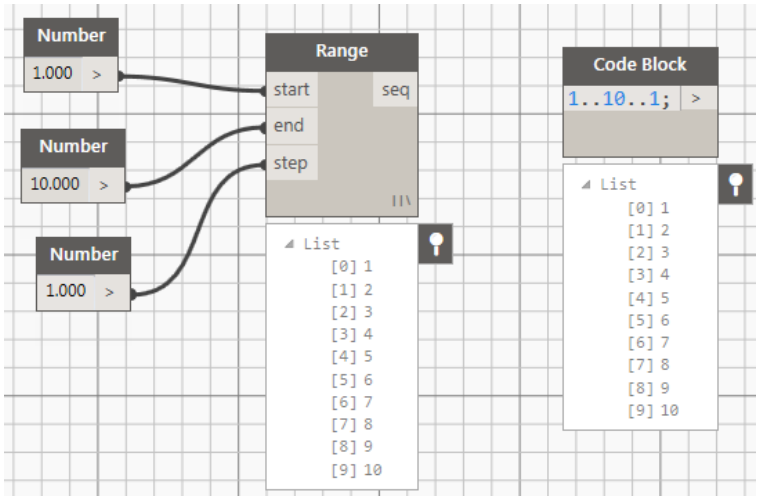
Codeblocks

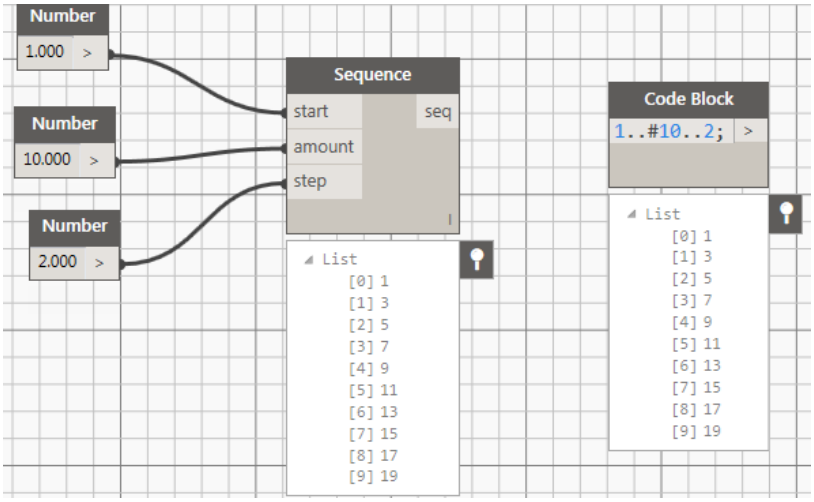
Code-Blöcke bieten die Möglichkeit, Listen einfacher zu definieren.
Ein Codeblock wird durch einen Doppelklick auf die Arbeitsfläche erstellt.

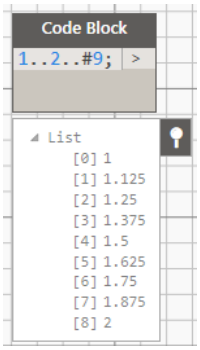
Hinweis: In einem Code-Block können Befehle untereinander stehen, jede Zeile muss allerdings mit einem Semi-Colon (;) enden.

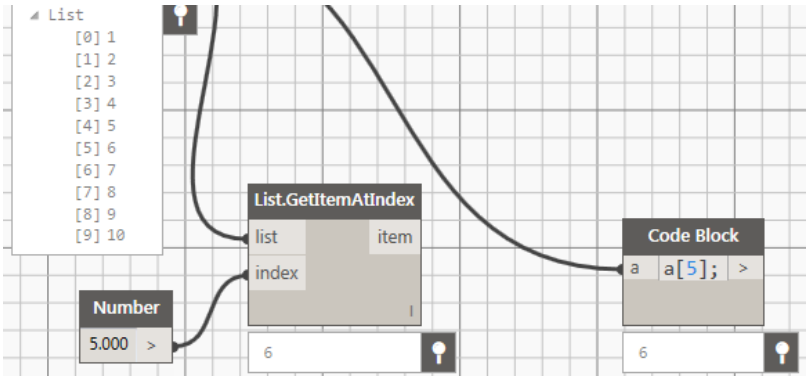
Beispiele:

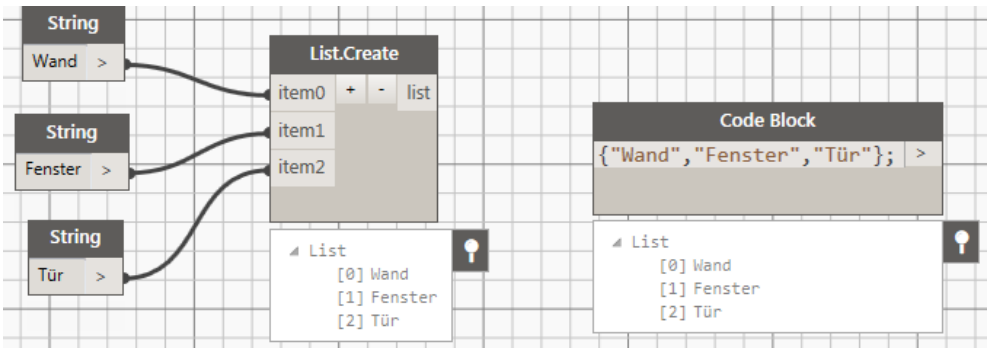





Number
1.000
Number
10.000
Number
2.000
Sequence
start
seq
amount
step
List
[0] 1
[1] 3
[2] 5
[3] 7
[4] 9
[5] 11
[6] 13
[7] 15
[8] 17
[9] 19
Code Block
1..#10..2;
List
[0] 1
[1] 3
[2] 5
[3] 7
[4] 9
[5] 11
[6] 13
[7] 15
[8] 17
[9] 19


Code Block
1..2..#9;
List
[0] 1
[1] 1.125
[2] 1.25
[3] 1.375
[4] 1.5
[5] 1.625
[6] 1.75
[7] 1.875
[8] 2

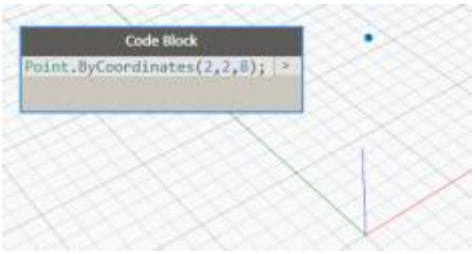

List
[0] 1
[1] 2
[2] 3
[3] 4
[4] 5
[5] 6
[6] 7
[7] 8
[8] 9
[9] 10
List.GetItemAtIndex
list
item
index
Number
5.000
6
Code Block
a
a[5];
6


String
Wand
String
Fenster
String
Tür
List.Create
item0
+
-
list
item1
item2
List
[0] Wand
[1] Fenster
[2] Tür
Code Block
{"Wand","Fenster","Tür"};
List
[0] Wand
[1] Fenster
[2] Tür

Designscript

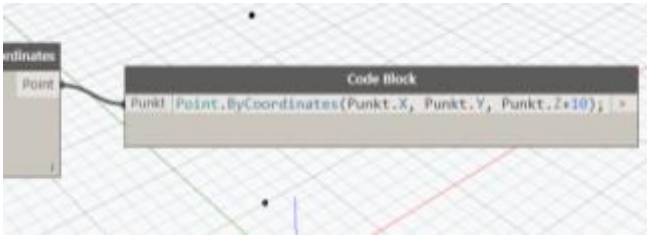
Codeblocks in Dynamo ermöglichen eine direkte Eingabe von Designscript Befehlen. Designscript ist eine einfache Skriptsprache, die speziell für Designzwecke entwickelt wurde.

In Codeblocks können alle in Dynamo verfügbaren Blöcke durch einfache Eingabe Ihrer Namen aufgerufen werden, was oft die Erstellung von wesentlich einfacheren und kompakteren Skripten ermöglicht.

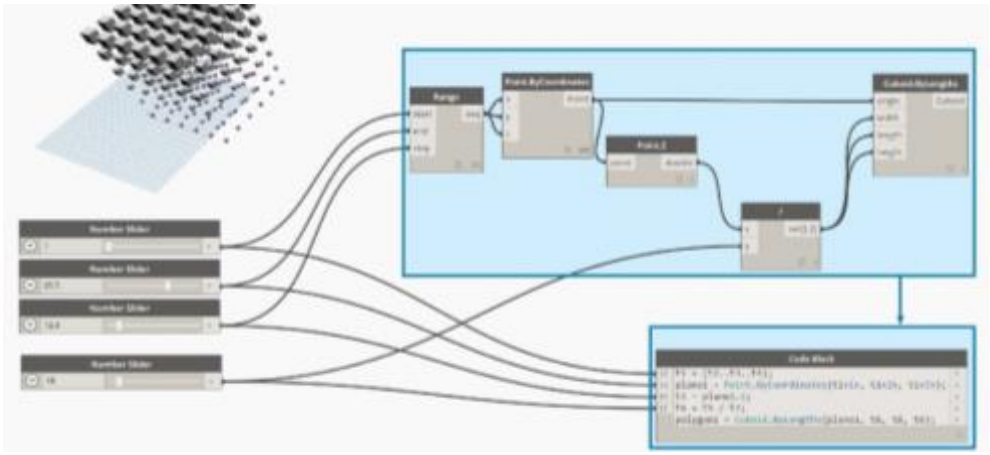


Auf diese Weise können komplexe Zusammenhänge in wenigen Zeilen beschrieben werden:

Nicht definierte Variablen erscheinen automatisch als Eingabevariable auf der linken Seite des Blocks und können somit mit Eingabewerten verknüpft werden.

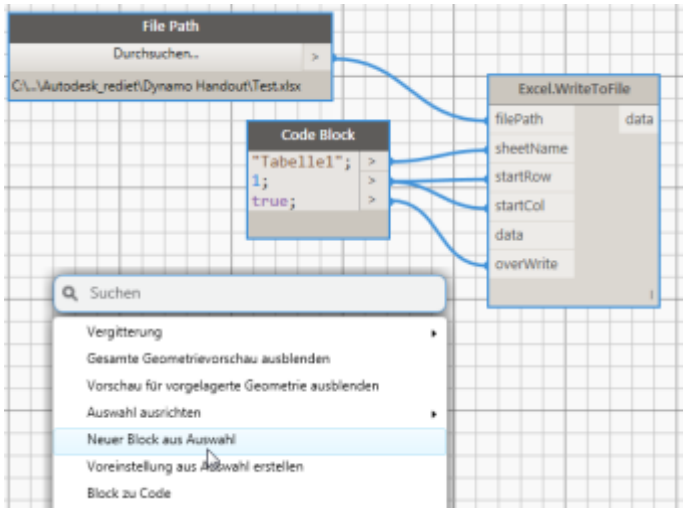


Dynamo bietet auch die Möglichkeit mehrere S*tandard*-Blöcke in einen Codeblock umzuwandeln. Dazu werden die Blöcke markiert anschließend das Kontextmenu durch einen Rechtsklick auf die Arbeitsebene aufgerufen. Hier befindet sich der Befehl „*Block zu Code*“:

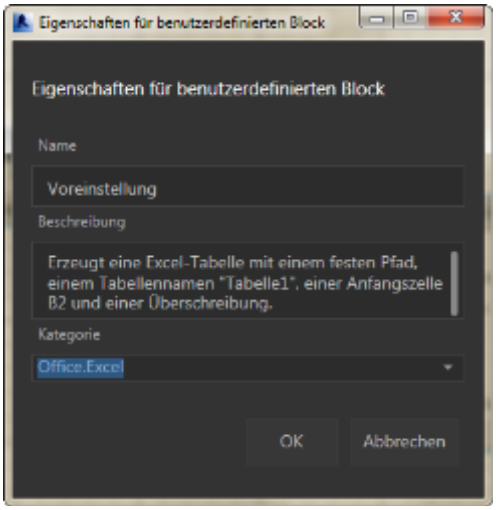


Benutzerdefinierte Blöcke

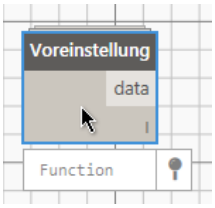
Bei wiederkehrenden Blockkombinationen macht es Sinn, benutzerdefinierte Blöcke anzulegen. Ähnlich wie dem Befehl „*Block zu Code*“ werden die gewünschten Blöcke ausgewählt und im Kontextmenu „Neuer Block aus Auswahl“ gewählt:



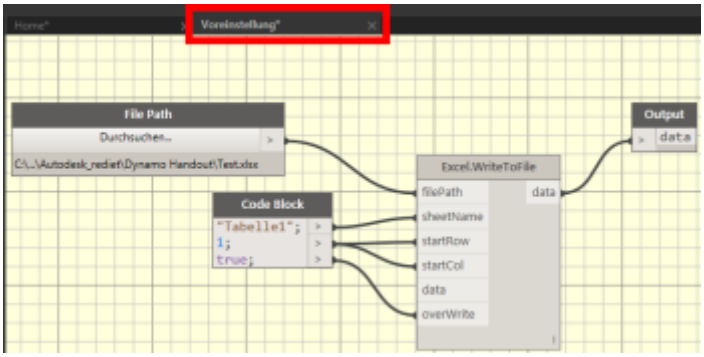
Hierbei wird in der Bibliothek ein eigener Block abgelegt, wobei der Name, die Beschreibung sowie die Kategorie, unter der der Block anschließend erscheint, frei wählbar sind:



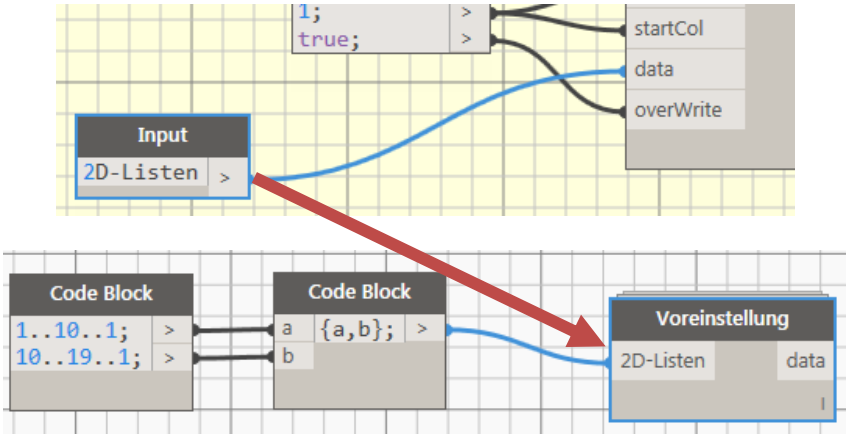
Mit einem Doppelklick wird der benutzerdefinierte Block in einer separaten Registerkarte geöffnet und kann jederzeit bearbeitet werden:



→

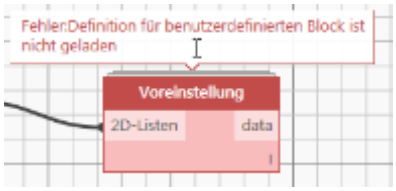


Der Block kann mit beliebige Eingabe. und Ausgabevariablen ausgestattet werden. Hierzu während in der Block-Bearbeitungsansicht (erkennbar an der gelben Arbeitsfläche) die Blöcke „input“ bzw „output“ hinzugefügt. Diese Blöcke müssen an den gewünschten Stellen im Block verbunden werden und können durch einfache Textangaben mit Namen versehen werden, die später die Variablen des Blocks bezeichnen:



Schließlich sollte der neue Block gespeichert werden. Dynamo speichert die Blockdefinitionen automatisch mit der Endung *.dyf* (anstatt wie Skripte mit *.dyn*).

Vorsicht: Wenn Skripte mit benutzerdefinierten Blöcken weitergegeben werden, müssen die benutzerdefinierte Blöcke mitgeschicken. Ansonsten kommt es zu Fehlermeldungen wie dieser:

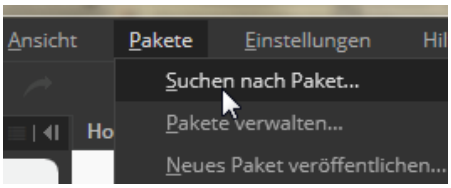


Den **Pfad** für die benutzerdefinierten Blöcke kann unter unter *Einstellungen → Pfad für Blöcke und Pakete verwalten…* definiert werden. Nur benutzerdefinierte Blöcke, die unter den hier angegebenen Pfaden abgelegt sind, werden in Dynamo automatisch geladen.

Pakete

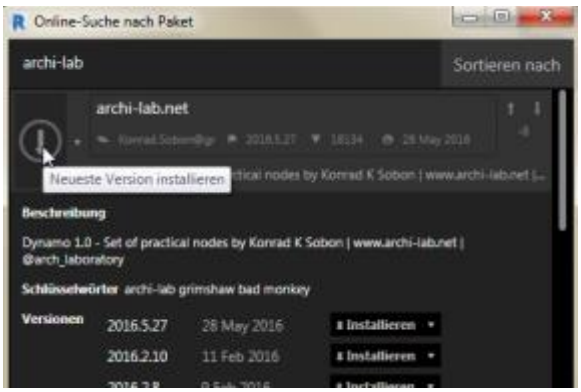
Benutzerdefinierte Blöcke können auch in Form von eigenen Paketen veröffentlicht werden.

Bereits veröffentlichte Pakete finden sich unter *Pakete – Suchen nach Paket….*

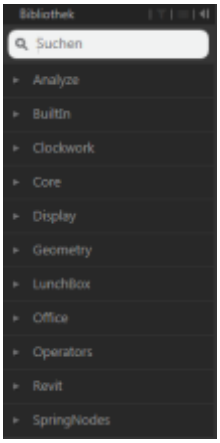


Zu den am meisten verwendeten Paketen gehören:

- Clockwork for Dynamo x.x.x
- Lunchbox for Dynamo
- Spring nodes
- Archi-lab.net



Nach der Installation von Paketen werden diese i.d.R. als neue Kategorien in der Bibliothek geladen:

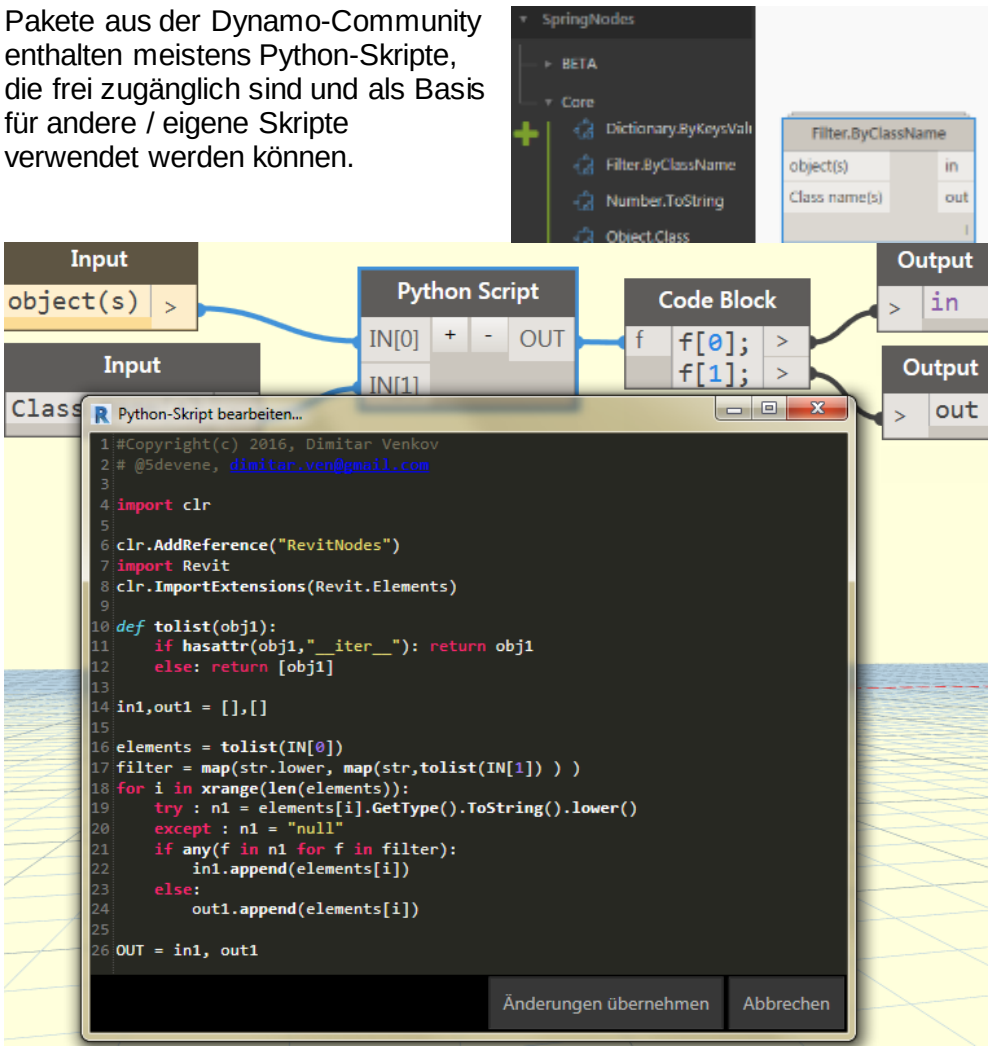


Python-Skript

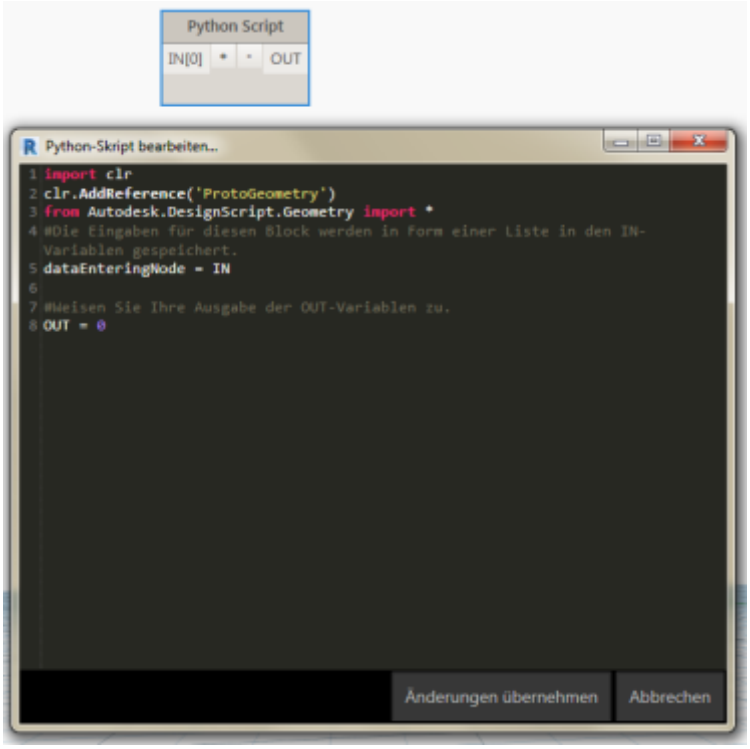
Dynamo basiert auf der Programmiersprache Python. Sie ist einer der am weitesten verbreiteten Sprachen, da sie besonders einfach zu ist. Dynamo bietet erfahrenen Benutzern die Möglichkeit, Python-Skripte direkt einzugeben.

Während bei der Nutzung der *Standard*-Blöcke in Dynamo die Voraussetzung gegeben ist, dass eine bestimmte Funktion bereits als ein vorgefertigter Block existiert, können mit Python beliebige Funktionen oder API-Interaktionen selbst definiert werden. Außerdem bietet Python Vorteile bei der Erstellung von komplexen, bedingten Anweisungen (wenn …, dann …) oder Schleifen.

Pakete aus der Dynamo-Community enthalten meistens Python-Skripte, die frei zugänglich sind und als Basis für andere / eigene Skripte verwendet werden können.



Der Pythonskript kann durch den speziellen *Python*-Block (unter *Core – Scripting) definiert werden*.
Mit einem Doppelklick auf eine freie Stelle des Blocks (mittelgrauer Hintergrund) öffnet sich das Bearbeitungsfester:



Um über Dynamo auf Funktionen in Revit zugreifen können, müssen die verschiedenen Bereiche aus der Revit-Bibliothek importirt werden:

```
# aus IronPython
import clr

# Import von Core Blöcken
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# Import von RevitNodes
clr.AddReference("RevitNodes")
import Revit

# Import von Revitelementen
from Revit.Elements import *

# Import des Dokument-Managers
clr.AddReference("RevitServices")
import RevitServices
from RevitServices.Persistence import DocumentManager

import System
```

Links

Grundlagen

Offizielle deutschsprachige Infoseite mit Blogs, Links und Lernmaterialien:
http://dynamobim.de/

Offizielle englischsprachige Dynamo-Seite mit Video-Tutorials
http://dynamobim.com/learn/

Englischsprachiges Dynamo-Forum
https://forum.dynamobim.com/

Deutschsprachiges / mehrsprachiges Online-Handbuch:
http://dynamoprimer.com/

Dynamo Paket-Manager
https://dynamopackages.com/

Weiterführende Informationen

Programmiersprache Python - Tutorial
https://py-tutorial-de.readthedocs.io/de/python-3.3/index.html
(© Copyright 2010 – 2013, Michael Markert et al. Revision 1a3f93b564cc+)

Scripting Autodesk Revit with RevitPythonShell
https://daren-thomas.gitbooks.io/scripting-autodesk-revit-with-revitpythonshell/content/index.html