TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

form follows code
Anwendung von Code zum Entwerfen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs / Diplom-Ingenieurin unter der Leitung von

Ivica Brnic
Dr. techn. Dipl. Arch. ETH SIA

E253

Architektur und Entwerfen

eingereicht an der Technischen Universität Wien

Fakultät für Architektur und Raumplanung

von

Mohammad Elzahaby

01126649

Wien, am 18.05.2022 **eigenhändige Unterschrift**

With the advent of new technologies in architecture, such as computer-aided design (CAD), the work of the architect has changed and is in a state of permanent flux. From CAD, the digital translation of an analog design act, to the latest development and use of building information modeling (BIM) to generate a virtual building model.

The design process is characterized by constant change. Apparently superficial decisions require profound adjustments. Innovations are neglected during the design process in order to escape the resulting effort. What expression does the use of coding in the context of BIM give to design?

In the software environment Revit with the plug-in Dynamo, an attempt is made to design a pavilion composed of BIM objects exclusively through code written in DesignScript. The aim is to avoid „manual operations" to avoid redundant steps and to work exclusively through the „source code". The intention is to show the potentials of this method.

An important factor is the complexity of the design.
The intention is to be able to explore and evaluate multiple design variants within the same typology. Asymptomatic versions are dropped from the selection. The result is described and presented in a way that provides an approach to this method.

Keywords: algorithmic, automatic, code, design, Building Information Modelling, DesignScript, Revit

Mit dem Einzug neuer Technologien in die Architektur, wie computerbasiertes Design (CAD), hat sich die Arbeit der ArchitektInnen verändert und befindet sich in einem Zustand des permanenten Wandels - vom CAD, der digitalen Übersetzung eines analogen Entwurfsakts, bis hin zur neuesten Entwicklung und Verwendung parametrischer Bauteile zum Generieren eines virtuellen Gebäudemodells, also Building Information Modelling (BIM).

Dabei ist der Entwurfsprozess von ständiger Veränderung geprägt. Scheinbar Oberflächliche Entscheidungen bedingen tiefgreifende Anpassungen. Innovationen werden beim Entwurf vernachlässigt, um dem daraus resultierend Aufwand zu entkommen. Welchen Ausdruck verleiht das Verwenden von Coding im Zusammenhang mit BIM dem Entwerfen?

In der Software-Umgebung Revit mit dem Plugin Dynamo wird versucht ausschließlich durch den in DesignScript geschriebenen Code einen Pavillon zu entwerfen, welcher sich aus BIM Objekten zusammensetzt. Ziel ist es dabei auf „manuelle Handgriffe" zu verzichten, um redundante Schritte vorzubeugen und ausschließlich über den „source-code" zu arbeiten. Die Absicht ist es, die Potentiale dieser Methode aufzuzeigen.

Ein wichtiger Faktor ist die Komplexität des Entwurfes.
Beabsichtigt ist die Möglichkeit mehrere Entwurfsvarianten innerhalb derselben Typologie zu erkunden und zu evaluieren. Asymptomatische Versionen fallen aus der Auswahl heraus. Das Ergebnis wird so beschrieben und dargestellt, dass es eine Herangehensweise für diese Methode bietet.

Schlagwörter: algorithmisch, automatisch, code, entwerfen, Building Information Modelling, DesignScript, Revit

Inhalt

Einleitung 6
Geschichtlicher Rückblick 7
State of the Art 11
Hypothese & Problemstellung 12
Die Methode 15
Algorithmus 17
Beispiel 18
Source Code 24
Entwurfsbeispiel 26
Bauteile 28
Boden 31
Stütze 33
Die Basis 35
Das Kapitell 37
Balken 39
Algorithmus 40
Source Code 64
Variation 67
Raum A-2 72
Raum F-5 74
Raum D-1 76
Raum I-3 78
Conclusio 84
Abbildungen 86
Literatur 88

Zur besseren Lesbarkeit wird in der vorliegenden Arbeit auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Es wird das generische Maskulinum verwendet, wobei beide Geschlechter gleichermaßen gemeint sind.

Einleitung

> *„Code is everywhere, from the systems within our computers, the methods within our design software, the language to run our machines and the robots that manufacture our cars and houses."* (Leach, et. al. 2018: 165)

Code - damit ist ein Vokabular gemeint, das die Übersetzung oder Austauschbarkeit von Begriffen ermöglicht. So wird Code in der Informatik dazu verwendet, eine für den menschlichen Anwender freundlichere Sprache zu gestalten, mithilfe derer Anweisungen an einen Computer übermittelt werden. Natürlich steht dazwischen ein Übersetzer, der sogenannte Compiler. Dieser verwandelt den „menschlichen" Code in die „maschinelle" Sprache.

Code ist die grundlegende Sprache, die es der physischen Welt ermöglicht, mit der digitalen Welt zu kommunizieren. Diese Sprache veranlasst unsere Maschinen, Aufgaben durchzuführen, zu wiederholen, fundierte Entscheidungen zu treffen, Eingaben/Ausgaben abzufragen und Wissen oder Ausgaben zu verarbeiten. Ebenso, wie sich durch Coding Gesellschaften und diverse Disziplinen verändert haben, wird es auch die architektonische Praxis treffen. (Leach, et. al. 2018: 165)

Ähnlich zur Verwendung des Computers, ist die Architektur generell eine Branche, die sich vergleichsweise spät technologischen Errungenschaften annimmt. Die wohl gebräuchlichste Programmiersprache in der Architektur ist Grasshopper die mit Rhinoceros 3D verwendet wird, um Geometrien parametrisch zu erzeugen und zu manipulieren.

Geschichtlicher Rückblick

Seitdem die Menschheit zu Bauen begann, gibt es Entwurfsaufgaben. Obwohl diese Entwürfe nach heutigen Maßstäben primitiv sind, lässt die mit der Geschichte wachsende Komplexität und Anforderung an die Bauwerke darauf schließen, dass diese vorab gedacht, geplant und an die Mitwirkenden des Bauakts vermittelt wurden.

Vor allem mit dem Sesshaftwerden des Menschen wurde aus der spontan entstandenen Behausung, der Urhütte, schnell ein in Relation betrachteter, anspruchsvoller Tempel. Dieser Akt des Bauens muss in einem sozialen Gefüge vollzogen worden sein. (Brnic 2013: 33)

Der Akt, der nicht mehr einfach nur funktionaler Art war, nämlich das Schützen vor den Umweltgefahren und eine der Ursprünge des Kampfes des Menschen mit der Natur markiert, wird in einem sozialen Gefüge vollzogen. Intrinsisch müsste eine Form der Kommunikation des zu vollziehenden Aktes des Bauens entstehen.

So hatten antike Kulturen, eine Sprache zur visuellen Kommunikation und Planung von Gebäuden entwickelt. Im Laufe der Geschichte hat der Mensch mithilfe der verfügbaren Technologien monumentale technische Meisterleistungen entworfen und konstruiert. Viele, wenn nicht sogar jedes dieser Kunststücke nutzten irgendeine Form des Entwerfens.

Wie Koolhaas beschreibt (2018: LVIII), spielt technologische Entwicklung stets eine wichtige Rolle. Sie bietet ein Werkzeug an, das im Laufe der Geschichte den Zugang zu anspruchsvollerer Architektur möglich macht.

Abb.1: Die Urhütte, abgebildet auf dem Titelbild der zweiten Ausgabe von Laugiers Essay über Architektur

Die Urhütte stellt einen theoretischen Anfang der Architektur dar. Es ist Sinnbild der Bewältigung der Natureinflüsse. Wie ein ständiger Kampf trotzt der Mensch der Natur.

Die Architektur, wie wir sie heute kennen, wurde kurz vor der Entdeckung Amerikas, konzipiert. Genauer gesagt, in den 1450er, Jahren als der Renaissance Architekt Leon Battista Alberti eine Trennung zwischen der intellektuellen Aufgabe des Entwerfens und dem Handwerk des Bauens vorschlug. Bis dahin waren Architekten im Wesentlichen Handwerker und der Bauakt war eine Gesamtaufgabe dieser. Alberti schlug vor, dass Gebäude vor ihrer Errichtung in ihrer Gesamtheit geplant werden sollten und der Architekt als Autor fungieren soll. So war die Architektur als eine Kunst des Entwerfens geboren. (Eastman, et. al., 2018: 177)

Folglich wurde die Zeichnung, die es dem Architekten erlaubt zu planen und mit diversen Entwurfsvarianten zu experimentieren, zum wichtigsten Werkzeug. Dieser Entwicklung war die Technologie des Papiers vorangegangen, die Europa erreichte. Welche Infolgedessen durch Innovationen erschwinglicher wurde. Es diente als Medium zum Experimentieren und Kommunizieren mit den Bauherren.

Bis zum Ende des 20. Jahrhunderts wurden nahezu alle architektonischen Zeichnungen händisch angefertigt. Wenn diese nicht vom Architekten geschaffen wurden, dann von einem eingeschulten technischen Zeichner, welcher nicht unbedingt den Entwurf entwickelt hatte.

Diese Systematik hat sich mit dem Einzug von CAD nicht wesentlich verändert. Viele Architekten haben oft kein Wissen darüber, wie CAD Programme funktionieren und verlassen sich nach wie vor auf spezialisiertes Personal, um den Entwurf von einer Handskizze in einen digitalen Plan umzusetzen.

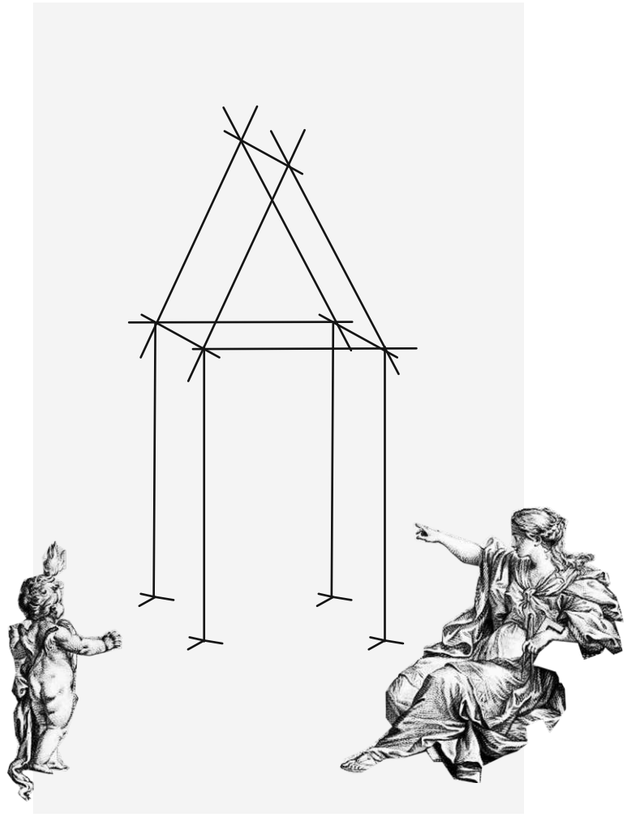


Abb. 2: Algorithmus der digitalen Urhütte

Die Essenz der Urhütte wird abgebildet. Unabhängig von Materialität und die damit einhergehenden Eigenschaften.

State of the Art

Eine eindeutige weitestgehend akzeptierte Definition von Building Information Modelling steht noch aus. (Campbell 2006)
Unter dem Akronym BIM versteht man meist einen Prozess bei dem Informationen ‚modelliert' werden. Dabei zielt der Prozess darauf ab, eine Informationsdatenbank für ein Bauwerk zu schaffen. (buildingSmart 2022)
Verschiedene am Markt vorhandene Softwares versuchen diese „Methodik" auf eigene Art und Weise zu implementieren. Sie kommen so auf diverse Arbeitsmedien, die dasselbe Ziel verfolgen, nämlich das Bauwerk in seiner Gesamtheit zu dokumentieren. Das Modell kann als Stütze dienen, um die durch die Abstraktion des Planes fehlenden Informationen zu ergänzen. (Eastman, et. al., 2018: 14)
Die Schwesterentwicklung von BIM, das „parametric design", entwickelte sich zeitgleich zu einer populären Entwurfsmethode. Dies leitet einen neuen digitalen Stil ein, den Parametrismus. Dabei ging es nicht vordergründig um das Verwenden von Algorithmen, sondern mehr um das Generieren von „weichen" Formen die oft zu komplex sind, um diese anderweitig zu definieren.
Dabei beschränkt sich die Anwendung von Code auf das Lösen von einzelnen, kleinen und gut begrenzten Aspekten eines Entwurfes wie eine parametrische Fassade. (Jabi 2013: 23)

Die Kombination aus BIM und Coding ermöglicht es potentiell, den Algorithmus auf alle Aspekte des Entwurfes auszuweiten. Dabei übernimmt der Algorithmus lediglich die Rolle des tektonischen Zusammenfügens der BIM Objekte zu einem zusammenhängenden Bauwerk. So kann die Urhütte in ihrer Essenz erfasst werden.

Hypothese & Problemstellung

In der konzeptuellen Entwurfsphase, die auch die kreativste Phase ist, werden Entscheidungen von enormer Bedeutung getroffen, welche tiefgründige Auswirkungen auf das Gesamtprojekt haben. Diese Entscheidungen bestimmen weitgehend die Kosten, die Komplexität der Konstruktion, die Technik und andere Aspekte. Diese konzeptuelle Phase stützt auf die Erfahrung und das Fachwissen des Entwerfers der intuitiv Entscheidungen trifft. (Eastman, et. al. 2018: 211)

Die ersten Versuche, diesen Prozess zu verbessern, geschahen mit Hilfe von BIM. Hierbei wird das Gebäude in einzelne Bauteile unterteilt. Diese Bauteile sind parametrisch, d.h. sie lassen sich auf die Bedürfnisse des Entwurfs abstimmen und anpassen. Beim Erstellen eines solchen virtuellen Gebäudemodells wird mit sehr viel Energie ein Mehrwert geschaffen, der oft aber auch wenig Spielraum für Veränderungen zulässt, denn jede Innovation am ursprünglichen Entwurfsgedanken zieht sehr viel Anpassungen an dem BIM-Modell nach sich. Daher wird der Einsatz von BIM oft hintenangestellt.

BIM entstand als Mittel, um die Komplexität moderner Gebäude zu bewältigen, da diese aufgrund von hohen Standards zu sehr komplexen Einheiten geworden sind. (Koolhaas, 2018: 374)

Oft entwickelt sich der Entwurf in den späteren Phasen zu einer Art Kompromisslösung aus dem ursprünglich Gewollten und dem, was zusätzlich als Auflage hinzukommt, da viele Informationen erst im späteren Verlauf zur Geltung kommen. Mit BIM hat sich das nicht großartig verändert, jedoch bietet das Verwenden von Algorithmen zum Entwerfen eine Möglichkeit, mehr Informationen in den konzeptionellen Entwurfsprozess einfließen zu lassen, wie zum Beispiel das Dimensionieren von Stützen und Trägern, abhängig von Spannweiten. Solche rein funktionale und andere gestalterische „Abhängigkeiten" lassen sich durch Algorithmen programmieren, wodurch bei der Konzeption darauf reagiert werden kann.

Vieles lässt sich mit der Anwendung von Code automatisieren und dadurch sowohl beschleunigen als auch die resultierenden Ergebnisse optimieren. Daher stellt sich die Frage, welchen Ausdruck das Verwenden von Coding im Zusammenhang mit BIM, dem Entwerfen verleiht.

Der in Code verfasste Algorithmus übernimmt dabei lediglich die Aufgabe des tektonischen Zusammenfügens von BIM-Objekten. Dabei muss dieser das Verhalten der Objekte steuern können - sowohl die Verortung im Raum als auch die Formveränderungen abhängig von Rahmenbedingungen, wie unter anderem der Entwurfsgedanke oder die statischen Anforderungen - sprich die Dimensionierung von Bauteilen. Dabei bekommen diese Abhängigkeiten gleich beim Entwerfen Geltung und helfen somit bei der Entscheidungsfindung.

Vergleichbare Methoden finden sich in der Entwicklung von Computerspielen wieder und lassen sich unter dem Begriff „Procedual Content Generation" sammeln.
(Shaker, et. al., 2016: 1)

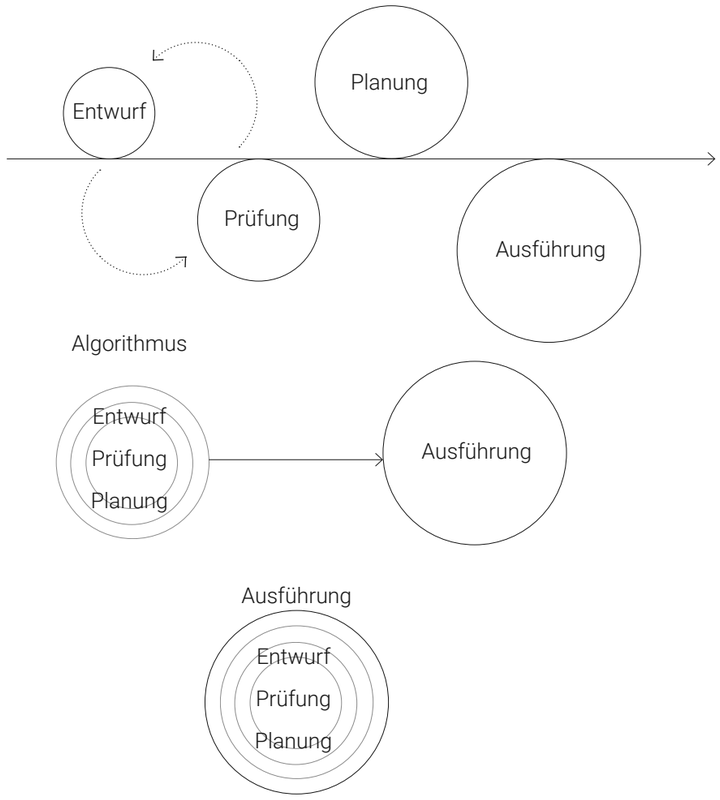


Abb. 3: Gegenüberstellung Linearer Entwurfsprozess nutzen von Algorithmen beim Entwerfen

Theoretisch kann ein Algorithmus so aufgebaut werden dass dieser alles erfasst, was das Bauwerk beinhaltet.

Die Methode

Im Laufe des Entwerfens nähert man sich der Komplexität des großen Ganzen schrittweise an. Dabei wirft die Lösung einer Problematik neue Fragen auf und so wiederholt sich der Ablauf. Man spricht dabei von iterativen Prozessen. Mit neuen Fragestellungen konfrontiert, kommen auch neue Lösungsansätze hervor. (Jabi 2013: 9)

Der Entwurfsprozess verläuft üblicherweise in einer linearen Abfolge: Entwerfen, Prüfen, Planen und abschließend wird das Entworfene ausgeführt. Wobei das Entwerfen und Prüfen ein iterativer Prozess ist.

Traditionell werden diese Arbeiten aufbauend abgewickelt, was sich auch in den Leistungsphasen des Architekten abbilden lässt. Oft ist es ein Wechselspiel zwischen den verschiedenen Phasen, da in der Anfangsphase sehr viele Informationen noch nicht bekannt sind bzw. auch bewusst nicht übernommen werden. So wird in all diesen Phasen kontinuierlich der Entwurf überarbeitet, da dieser auf neue Informationen reagieren muss. (Humppi, H., Österlund, T. 2016)

Beim Verwenden von Algorithmen können bereits in der ersten Phase des Entwurfs wesentlich mehr Informationen verarbeitet werden, was es theoretisch erlaubt, schon in der frühen Entwurfsphase auf statische Rahmenbedingungen oder materialspezifische Besonderheiten zu reagieren. Es können jedoch auch Entwurfsintentionen festgehalten werden, wie das Einhalten besonderer Achsen, Kanten oder Fugen.

Also lassen sich mit Algorithmen Abhängigkeiten zwischen Objekten festhalten, was mit BIM alleine eine Herausforderung darstellt. (Humppi, H., Österlund, T. 2016)

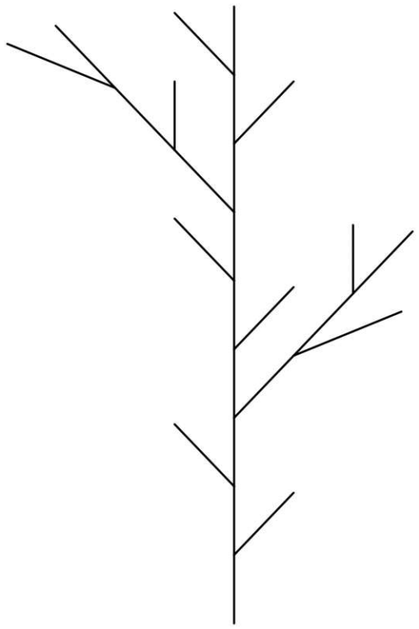


Abb. 4 Demonstration der Anwendung eines Lindenmayer-System Algorithmus:
1: F=F[+F]F[-F]F, Axiom: F, Kosntante: F, Iteration: 2, Winkel: 33

Eine Anweisung kann so einfach sein wie das Bewegen von Punkt A nach B. Dabei bilden Hindernisse auf diesem Weg „Probleme“ die in Form eines Algorithmus gelöst werden. Das Resultat kann dennoch oft unvorhersehbar sein.

Algorithmus

Für einen Algorithmus gibt es keine klare Definition, jedoch lässt er sich wie folgt beschreiben: Die sukzessive Anweisung zum Erhalten eines Resultats. Dabei ist vor allem das Umwandeln der für Menschen klar verständlichen Anweisungen in für Computer durchführbare Befehle eine Herausforderung. Hierfür ist das Verständnis von dem wie ein Computer arbeitet sowie das dazu benötigte semantische Denken nicht unwesentlich. Dieses Verarbeiten einer Handlung in Anweisungen und wiederum in Computer Code wird als Computational Thinking bezeichnet. (Jabi 2013:22)

Überträgt man dies auf den architektonischen Entwurfsprozess, so kann dieser ebenso aus definierten Anweisungen zum Bewältigen einer Bauaufgabe bestehen. Diese Anweisungen können auch Regeln sein, welche die Essenz des Entwurfs beschreiben. Oft sind verschiedene Entscheidungen im Zusammenhang mit natürlichen Gegebenheiten verknüpft. Aber auch Entwurfsintentionen können nach Bedingungen verlangen, die zu Zusammenhängen führen. So kann im Wohnbau die Regel sein, dass ein Aufenthaltsraum stets ein Fenster haben muss oder dass die Stütze stets zentral auf einer Achse stehen soll.

Anhand eines einfachen Beispiels wird nun erläutert, wie Algorithmen mit BIM angewendet werden können. Dabei wird ein Raum aus 4 Stützen, einem Boden und einer Decke modelliert. Der Code wird in der Programmiersprache „DesignScript“ geschrieben. Diese ermöglicht sowohl textliche als auch visuelle Programmierung. Die Schnittstelle zu BIM erfolgt durch „Dynamo“ - integriert in „Revit“. Im ersten Schritt wird die Essenz in Form von Regeln definiert. Dannach werden diese Regeln in From von Code in den Algorithmus übertragen.

Beispiel

Diese Methodik wird nun anhand eines einfachen „Bauwerks“ demonstriert. Zunächst werden die Regeln des Algorithmus bzw. der Entwurfsgedanke beschrieben.

1. Der Boden wird durch ein quadratisches Rechteck beschrieben.
2. An jeder Ecke werden Stützen platziert.
3. Auf den Stützen liegt die Decke auf.

Nun werden diese Regeln, nach und nach in Code übersetzt. Somit bilden sie logischen Anweisungen.

Der Boden wird durch ein quadratisches Rechteck beschrieben.
Um eine „Bodenplatte“ zu erzeugen wird ein Polygon benötigt, dieses lässt sich durch 4 Eckpunkte in „DesignScript“ erstellen.

Der erste Schritt ist es die Maße dieses Rechtecks als Variable zu definieren. Diese sind vom Typ „Zahl“.

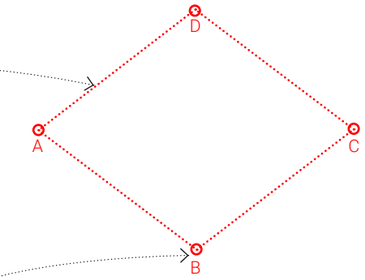
Name der Variable — Wert der Variable

```
length = 200;
width = 200;
```

Nun wird mithilfe dieser Variablen jeder Eckpunkt als eigenständige Variable des „Typs“ Punkt erstellt. Diese beinhaltet ein Objekt mit den X,Y und Z Koordinaten des Punktes.

Funktion zum erstellen eines Punktes im 3D Raum

```
A = Point.ByCoordinates(-length,width);
B = Point.ByCoordinates(length,width);
C = Point.ByCoordinates(length,-width);
D = Point.ByCoordinates(-length,-width);
```



Vorab muss bewusst entschieden werden, welche Bauteile verwendet werden. Hier kommt die Erfahrung des Anwenders ins Spiel

Diese Punkte werden nun in eine Liste von Punkten zusammengefasst. Diese Liste wird auch in einer Variablen gespeichert, sogenannte „corners“ die alle Punkte fasst.

```
corners = [A,B,C,D];
```
Erstellen einer Liste mit 4 Punkte

Im nächsten Schritt wird mit Hilfe dieser Variable ein Polygon erstellt. In dem Fall ist das Polygon ein Rechteck. Dazu wird die vorhandene Funktion „Polygon.ByPoints(...)“ verwendet.

Auswählen des Typen für Geschoßdecke

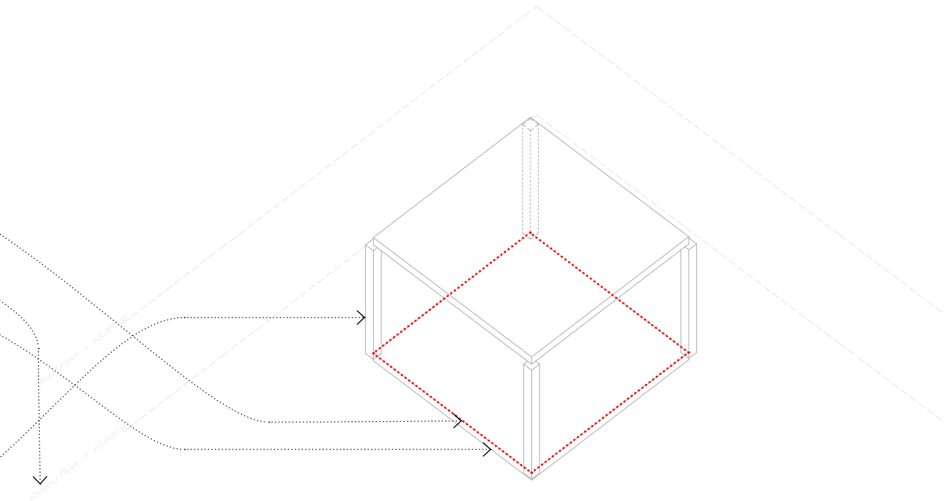
Geschoßdecke auf basis des Rechtecks erstelalen

```
polygon = Polygon.ByPoints(corners);
polygonCurve = PolyCurve.Curves(polygon);
floorType = FloorType.ByName(„STB 20“);
groundLevel = Level.ByElevationAndName(
     0,“Ground Floor“);
Floor.ByOutlineTypeAndLevel(
     polygonCurve, floorType, groundLevel);
```
Rechteck Zeichnen

Anschließend wird mit Hilfe dieses Rechtecks der Umriss der Bodenplatte definiert, dass schließlich ein BIM Element ist. Nun werden die Stützen an den Ecken platziert.

```
columnFamily = Family.ByName(
     „STB Stütze - rechteckig“);
columnType = FamilyType.ByFamilyAndName(
     columnFamily,“STB 200 x 200“);
columns = FamilyInstance.ByPointAndLevel(
     columnType, corners, groundLevel);
```

Hierfür greifen wir auf dieVariable „corners“ zurück, und nutzen diese Eckpunkte als Ursprungspunkte unserer Stütze.

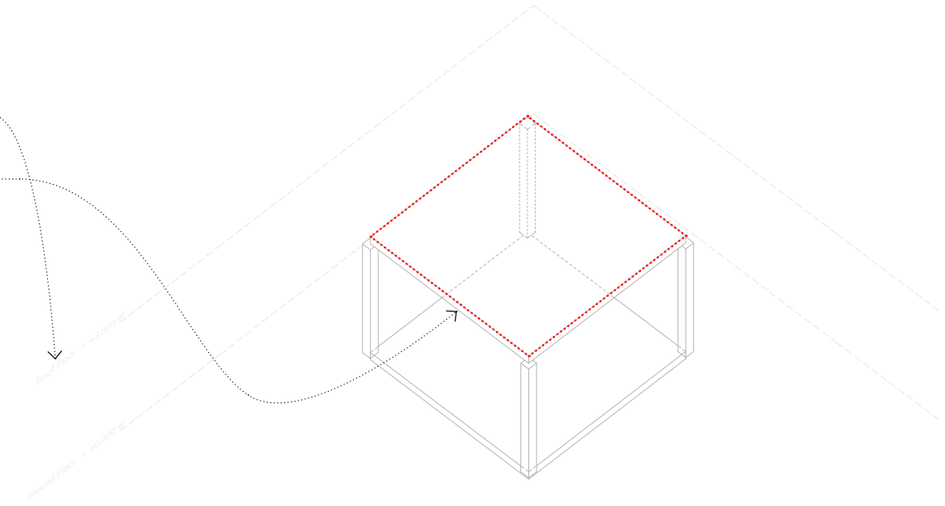


Auf diese Stützen kommt nun die Decke. In diesem einfachen Fall kann das Polygon des Bodens wieder verwendet werden um die Decke zu erstellen.

```
height = 300;

roofLevel = Level.ByElevationAndName(
     height,"Roof Floor");
roofType = RoofType.ByName(„STB 20“);
Roof.ByOutlineTypeAndLevel(
     polygonCurve, roofType, roofLevel);
```

Eine Ebene auf der gewünschten Höhe von 3 Meter als Referenz für die Decke wird erstellt. Anschließend wird mit Hilfe dieser Ebene und der Auswahl des Decken-Typus eine Decke mit dem Umriss des ursprünglichen Polygons erstellt.

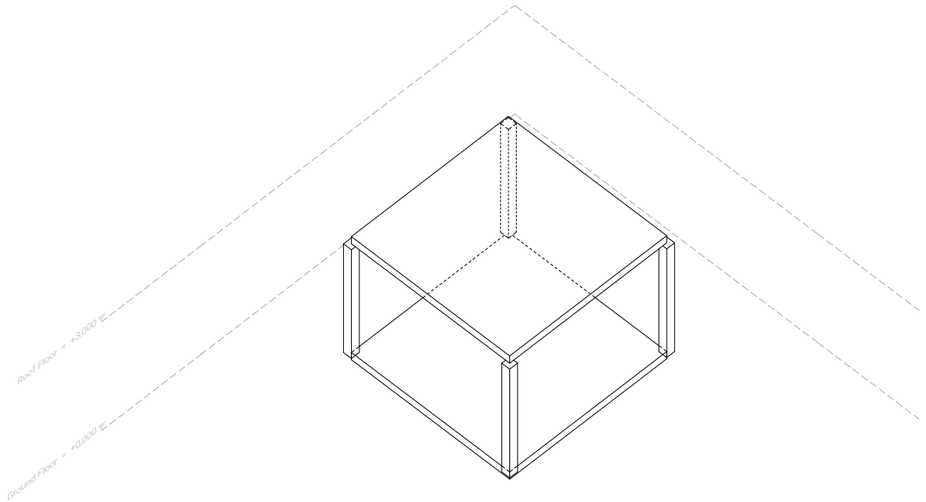
```
Element.SetParameterByName(columns,"Oberste
Ebene",roofLevel);
Element.SetParameterByName(columns,"Versatz oben",0);
```



Source Code

```
length = 200;
width = 200;

A = Point.ByCoordinates(-length,width);
B = Point.ByCoordinates(length,width);
C = Point.ByCoordinates(length,-width);
D = Point.ByCoordinates(-length,-width);
corners = [A,B,C,D];

polygon = Polygon.ByPoints(corners);
polygonCurve = PolyCurve.Curves(polygon);
floorType = FloorType.ByName(„STB 20“);
groundLevel = Level.ByElevationAndName(0,“Ground Floor“);
Floor.ByOutlineTypeAndLevel( polygonCurve, floorType, groundLevel);

columnFamily = Family.ByName(„STB Stütze - rechteckig“)
columnType = FamilyType.ByFamilyAndName(columnFamily,“STB 200 x 200“);
columns = FamilyInstance.ByPointAndLevel(columnType, corners,
groundLevel);

height = 300;

roofLevel = Level.ByElevationAndName(height,“Roof Floor“);
roofType = RoofType.ByName(„STB 20“);
Roof.ByOutlineTypeAndLevel( polygonCurve, roofType, roofLevel);

Element.SetParameterByName(columns,“Oberste Ebene“,roofLevel);
Element.SetParameterByName(columns,“Versatz oben“,0);
```

Dieser Code kann nun kopiert werden und in einer eigenen Instanz von „Revit & Dynamo“ eingefügt und ausgeführt werden. Dabei muss auf die Bezeichnung der verwendeten Familien- und Typen für BIM Bauteile geachtet werden.



Entwurfsbeispiel

Diese Methode des algorithmischen Auflösens eines Entwurfsgedanken wird nun am folgenden Entwurfsbeispiel angewendet. Anhand einer Skizze werden die Regeln für den Entwurf festgehalten, aus welchem anschließend der Algorithmus erstellt wird.

1. Das Bauwerk wird durch ein Raster in Längs- wie auch Querrichtung unterteilt.

2. In die daraus resultierenden Flächen wird eine Bodenplatte platziert.

3. Die Kreuzungspunkte des Rasters bestimmen die Position der Stützen.

4. Kreuzpunkte können „übersprungen“ werden.

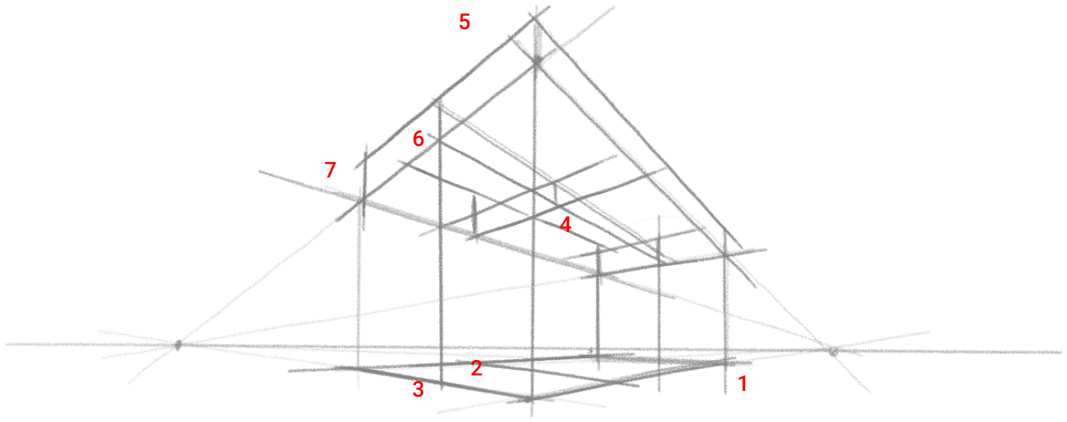
5. Die Dachkonstruktion besteht aus Trägern.

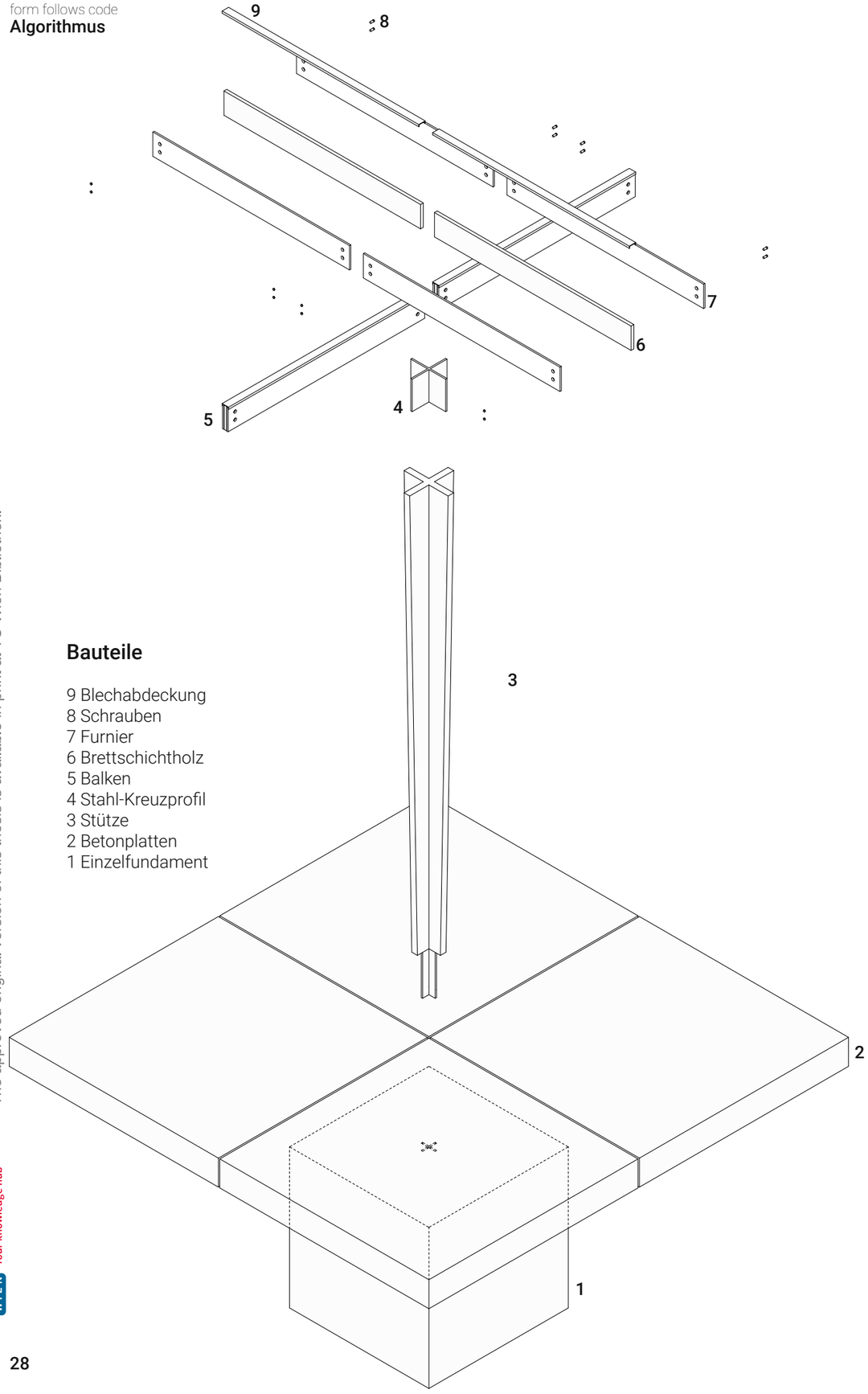
6. Die Träger liegen an ihren Enden auf Stützen auf.

7. Die Trägerhöhe wird durch die Stützweite bestimmt.

Der Denkprozess, also diese Anweisungen dem Computer zu übermitteln, wird auch als „Computational thinking“ bezeichnet. Dabei ist es wichtig zu verstehen wie der Computer arbeitet bzw. die virtuelle Umgebung die verwendet wird.

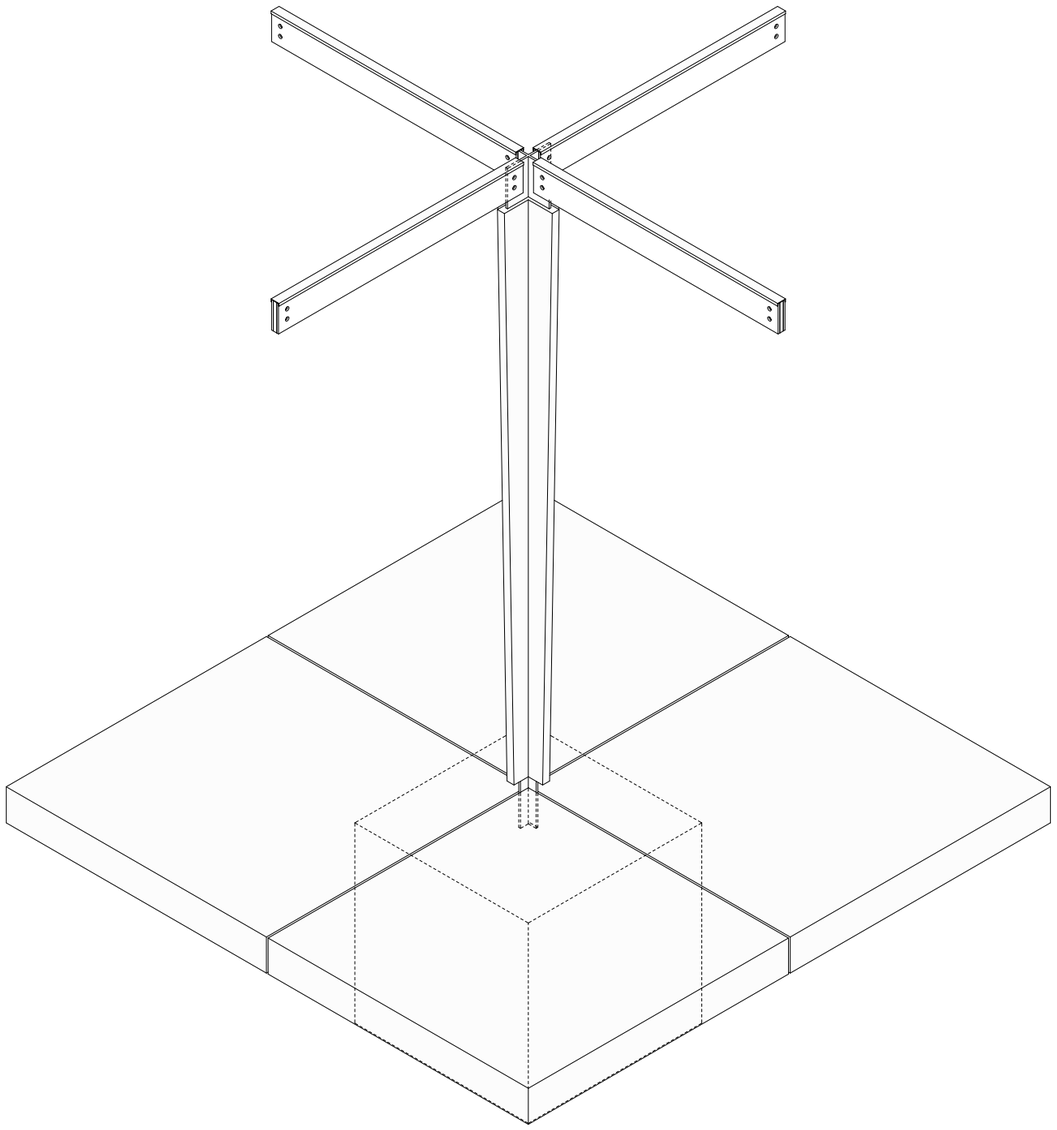
Das Bauwerk besteht somit aus Platten für den Boden, Stützen für das Tragen der Dach-Konstruktion und Trägern, welche die Dachkonstruktion bilden.

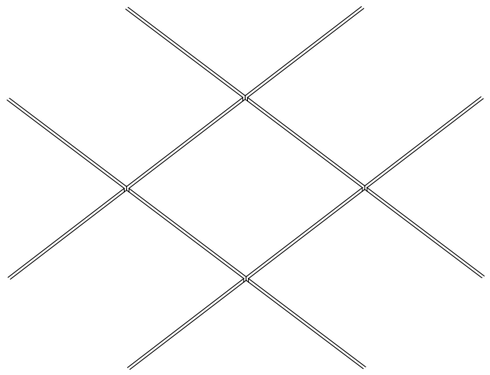




Bauteile

9 Blechabdeckung
8 Schrauben
7 Furnier
6 Brettschichtholz
5 Balken
4 Stahl-Kreuzprofil
3 Stütze
2 Betonplatten
1 Einzelfundament

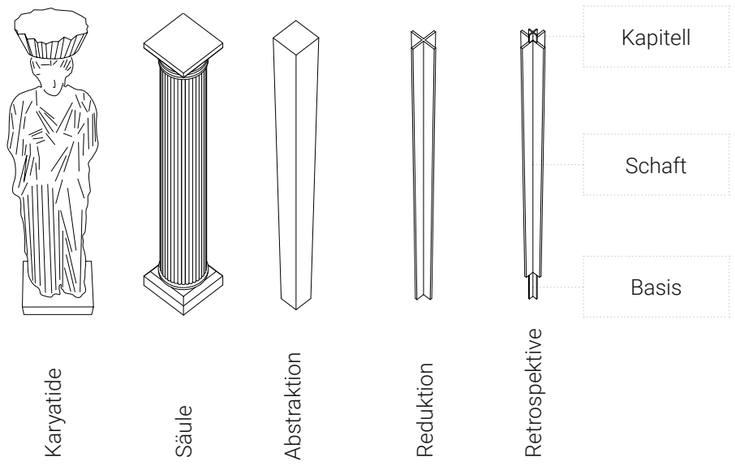




Boden

> *...the floor as an elevated platform has been a universal technology for separating the clean from the dirty...*
> (Koolhaas, R. 2018: 25)

Aus Platten wird eine ebene Fläche geschaffen, die abgehoben vom Erdreich, den Boden bildet.
Diese werden in einem orthogonalen Raster angeordnet. Die Fugen zwischen den einzelnen Platten bilden das Raster. In der Kreuzung zweier Fugen wird in regelmäßigen Abständen eine Stütze platziert. Die Größe der Platten bzw. des Rasters ist ein Parameter, der den Algorithmus steuert. Somit spiegelt sich das Raster im Boden wieder. Der Gestaltungswille ist es in den Kreuzpunkten der Fugen Stützen zu platzieren. So wird die Fläche der Platte nicht gestört.



Stütze

Mit dem weiteren Ausformulieren des Entwurfes kommen immer mehr Anforderungen auf die Stütze zu. Sie soll nun nicht nur die tragende Funktion erfüllen, sondern auch mit allen Bauteilen in Harmonie stehen. So wurde aus dem quadratischen Profil, ein kreuzförmiges, um den Querschnitt zu optimieren.
Die Dimensionierung der Stütze hängt mit der Spannweite zusammen, die wiederum die Größe der Balken beeinflusst und damit das zu tragende Gewicht erhöht. Hierfür wird eine Konstante verwendet, die abhängig von der Spannweite die Längs- und Querrichtung der Stütze verändert, um den Eindruck, der durch die statischen Anforderungen entstandenen, Veränderungen zu simulieren.

Da jedoch die Stütze sowohl am Boden als auch an der Decke an andere Bauteile anschließt, benötigt diese einen Übergang. Diese Problematik war bereits in der Antike vorhanden und die Antwort darauf war wahrscheinlich die Schaffung der Übergänge am Boden in Form einer Basis. Und wiederum das Anknüpfen an der Decke in Form eines Kapitells.

Andererseits wird aber der Antike zugesprochen, dass der menschliche Körper mit seinen Proportionen zur Gestaltung der Stütze führte. (Brnic, 2019: 43)

Ob der Mensch als Vorlage für die Formfindung und Ausbildung der Säulen diente bleibt dahingestellt. Was jedoch definitiv gesagt werden kann, ist dass das Arbeiten mit Bauteilen, wie es bereits bei der „Urhütte“ geschah, stets zu Knotenpunkten führt. Diese verlangen auch nach einem Übergang zum anküpfenden Teil.

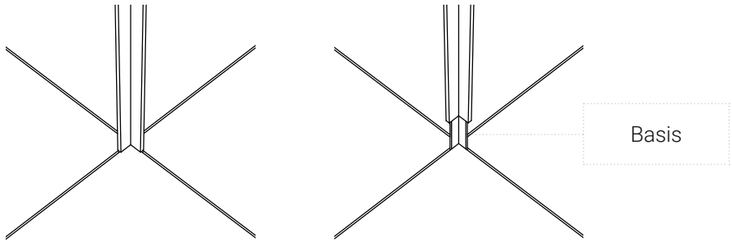
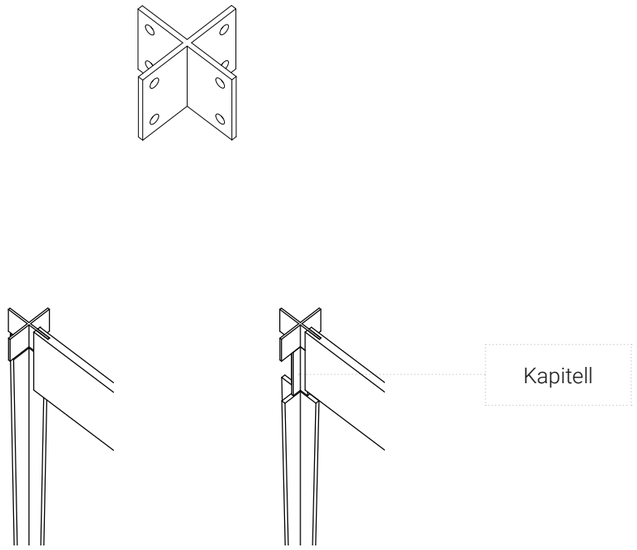


Abb. 6 Deutscher Pavillon, Barcelona, 1929, Querschnitt der Stütze.
Zwar fehlen Basis und Kapitell bei Mies völlig, dennoch verlangt das zusammenfügen von Bauteilen wie 4 L-Profile nach verbindenden Elementen

Die Basis

Da die Stütze sich in der Fuge zwischen den Platten befindet, wird ein Element benötigt, dass das Zusammentreffen diverser „Linien“ aufnimmt und in eine neue Form, der Stütze, überträgt.

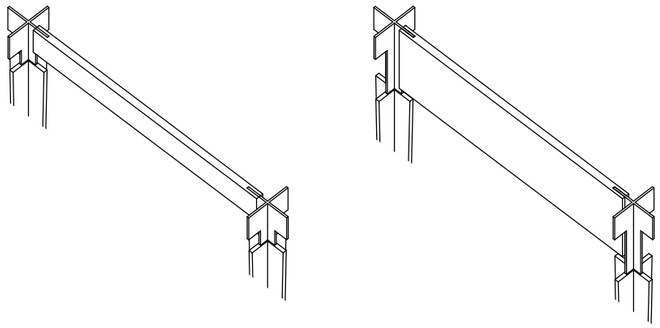
Hierfür ist eine Basis aus Stahl nötig, die nicht nur dazu dient, die statischen Kräfte aufzunehmen und in das Fundament weiterzuleiten, sondern auch das Holz vor Bodenfeuchte zu schützen.



Das Kapitell

Vor der Ausgestaltung wurden die grundsätzlichen algorithmischen Ansätze programmiert. Darunter fallen die Punkte für die Positionierung der Stützen, welche auch das Raster bilden.

In dieser Phase stoßen mehrere Bauteile an einem Punkt zusammen und verursachen „Konflikte“. Diese Konflikte, die wahrscheinlich auch in der Antike der Auslöser für die Ausbildung der klassisch griechischen Säulenordnung waren, führten auch hier zu einer Gliederung der Stütze in Basis, Schaft und Kapitell. Ein anpassbares Kapitell ist nötig, um die unterschiedlichen Balkenhöhen aufzunehmen. Dies ermöglicht die Verbindung zur Stütze.



Balken

> *... the most basic concept of ceiling as underside of the floor above, with structural members exposed and unadorned.*
> (Koolhaas, R. 2018: 218)

Entsprechend dem einfachen Konzept einer Decke bildet hier der Balken den oberen Abschluss des Bauwerks.

Um statische Eigenschaften in den Algorithmus einbeziehen zu können, müsste man die vorherrschende Herangehensweise für Statik, überdenken. Diese basiert darauf, dass das Bauwerk zuerst entwickelt wird und anschließend berechnet bzw. geprüft wird. Um die geforderten Werte zu erreichen, werden entsprechende Elemente des Entwurfs angepasst bzw. statische Maßnahmen getroffen. Diese wirken sich direkt auf den Entwurf aus.
Man könnte meinen, es sind fremde Einflüsse, die zu einem Kompromiss zwischen dem Entwurfsgedanken und der physikalischen Anforderungen der Natur, der Gravitation führt.

Beim algorithmischen Arbeiten kann Statik bereits im Entwurfsprozess einbezogen werden. Um die statischen Veränderungen zu simulieren, wird eine Konstante verwendet, welche sich ausschließlich auf die Höhe des Balkens auswirkt. Idealerweise müsste abhängig von Material, Verbindung und Spannweite eine komplexe Berechnung stattfinden. Folglich können bereits im frühen Entwurfsprozess die Auswirkungen der Statik wahrgenommen werden und entsprechende Entscheidungen getroffen werden.

Algorithmus

Der verantwortliche Code für die tektonische Zusammensetzung der Bauteile lässt sich in 4 Teile gliedern.

1. Dem Eingang (Input) und die Verarbeitung der Variablen bzw. Parameter des Entwurfs.
2. Die Platzierung der Bauteile.
3. Die Manipulation der einzelnen Bauteile, um statische Veränderungen zu simulieren.
4. Das Erstellen von Plangrafiken in Form von Grundriss, Schnitt, Ansicht und Perspektive.

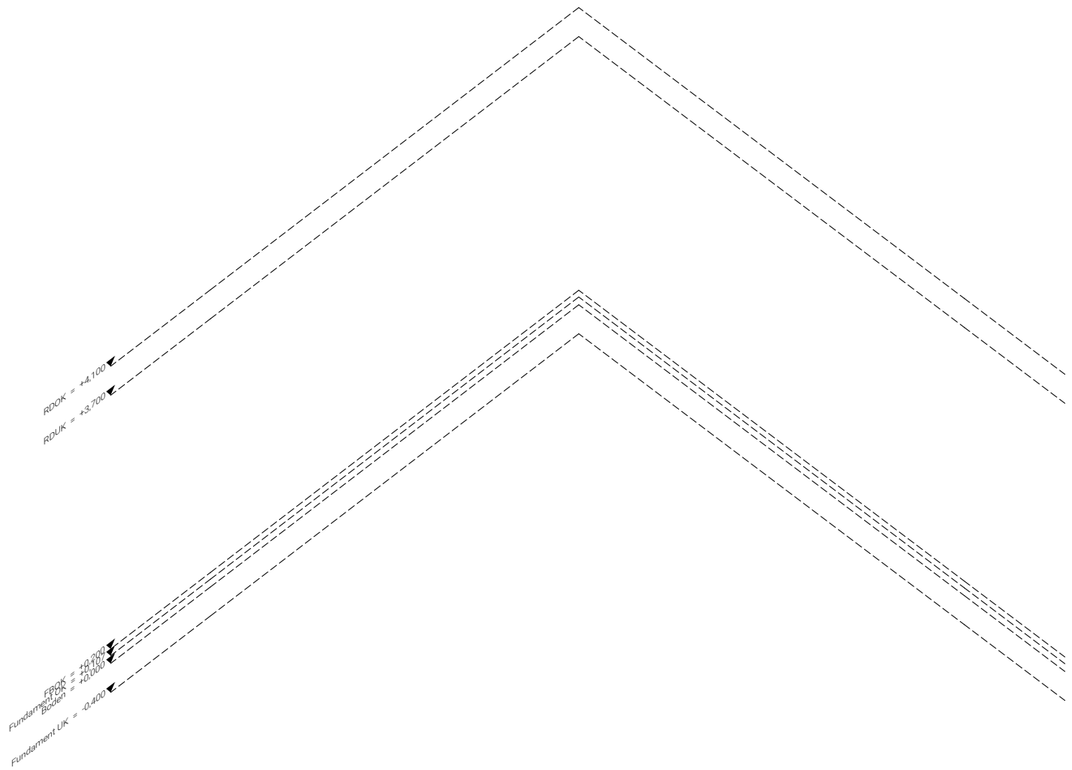
Im ersten Schritt werden die Ebenen definiert, auf die sich die Bauteile des Bauwerks beziehen sollen.

Die Höhen der Ebenen werden teilweise wechselseitig von Bauteilen definiert. So kann sich die Tiefe des Fundaments mit steigenden Lasten ändern.

Die Höhe des Raumes definiert die Unter- & Oberkante des Daches. Die Raumhöhe ist eine Variable, die für die Generierung diverser Varianten von Bedeutung ist.

```
Boden = 0;
FBOK = 10 + tileHeight
RDUK = 350 + FBOK
RDOK = RDUK + beamheight
Fundament = xxxx

levelRDOK = Level.ByElevationAndName(RDOK, „RDOK“);
levelRDUK = Level.ByElevationAndName(RDUK, „RDUK“);
levelFBOK = Level.ByElevationAndName(FBOK, „FBOK“);
levelFloor = Level.ByElevationAndName(Boden, „Boden“);
levelFUK = Level.ByElevationAndName(Fundament, „Fundament UK“);
levelFOK = Level.ByElevationAndName(FBOK-tileHeight, „Fundament OK“);
```

In der realen Welt ist das abstrakte Konstrukt der Ebenen nicht vorhanden. Auch mit Algorithem lassen sich diese wegrationaliseren.



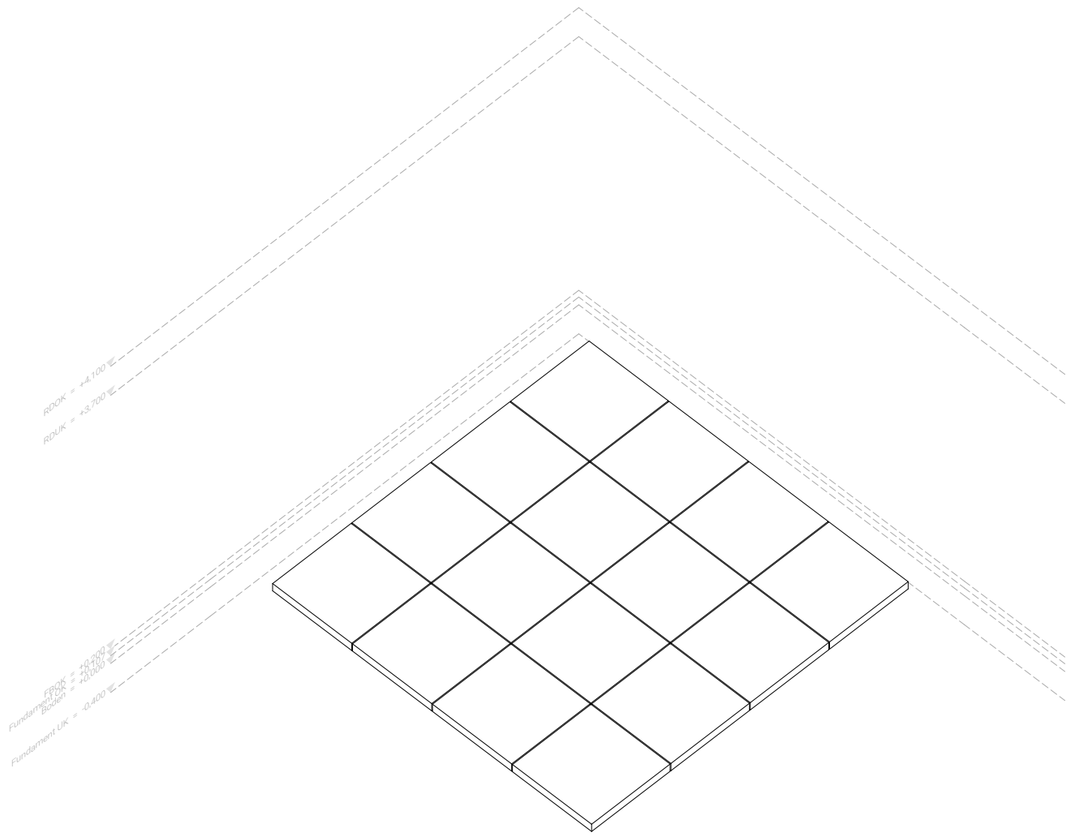
Zunächst werden die nötigen Punkte für die Platzierung der Bodenplatte generiert. Hierfür werden die grundsätzlichen Variablen des Bauwerks herangezogen, diese sind:

- Länge/Breite/Höhe
- Teilungen in Längs/Querrichtung

Aus diesen Variablen werden alle anderen Werte berechnet, die das Bauwerk definieren.

Zur Generierung der Punkte wird die Funktion einer „Reihe“ verwendet, bei der die erste Zahl, die letzte und die Abstände festgelegt, in der eine Zahlenfolge erstellt werden soll. Dies wird für die Längs- als auch Querrichtung verwendet. Mit den beiden Reihen lassen sich nun ein Raster von Punkte erstellen.

An jedem Punkt wird nun das BIM-Element platziert. Die Länge/Breite des Elements wird durch die Teilung der Gesamtlänge berechnet.



Das Raster ist ein Werkzeug das in der Architektur immer wieder auftaucht. Je freier man ist, umsomehr kehrt dieses zurück.

Code:

Durch // gekennzeichnete Zeilen sind Kommentare

Deklarieren einer Funktion

Argumente oder Parameter der Funktion

Ausgabe Wert der Funktion

```
// Funktion zum ermitteln der Platten Größe
def    calc_tile_size(totalLength,relativeLength)    {
    return  Math.Round(totalLength/  (relativeLength+1));
};

// Funktion zum berechnen der Punkte
def calc_coordinate(start,end,distance,gap) {
    return start..end+start+distance..distance+gap;
}

tileLength = calc_tile_size(totalLength,relativeLength)
tileWidth = calc_tile_size(totalLength,relativeLength)
tileHeight = tileLength/tileWidth * 10;

x = calc_coordinate(start,end,tileLength,gap);
y = calc_coordinate(start,end,tileWidth,gap);
points = Point.ByCoordinates(x,y,0);
```

Auswahl der Familie und Typ des Objekts

```
tileFamily = Family.ByName(„tile“):
tileType = columnType = FamilyType.ByFamilyAndName(
    tileFamily,“tile“);
tiles = FamilyInstance.ByPointAndLevel(
    plattenTyp,points,levelFBOK);

//Eindeutige Bezeichnung jeder platte vergeben
id = List.IndexOf(points,points);
Element.SetParameterByName(tiles,“Kennzeichen“,id);

//Größe der Platte festlegen
Element.SetParameterByName(tiles,“Length“,tileLength);
Element.SetParameterByName(tiles,“Width“,tileWidth);
Element.SetParameterByName(tiles,“Height“,tileHeigth);

//Höhenversatz von Referenzeben der Platte definieren
Element.SetParameterByName(tiles,“Höhe von Ebene“,0);
```

Informationen in die Paramter des Objektes eintragen

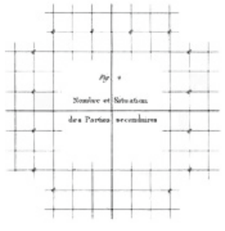
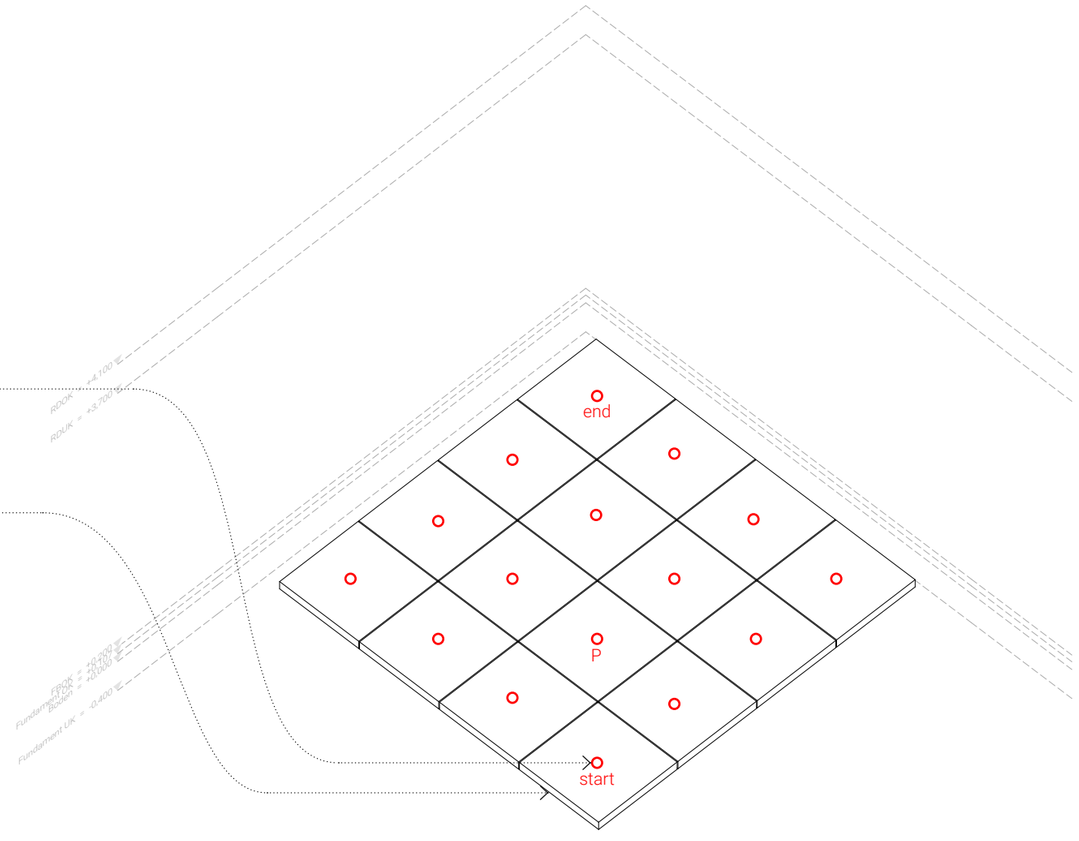
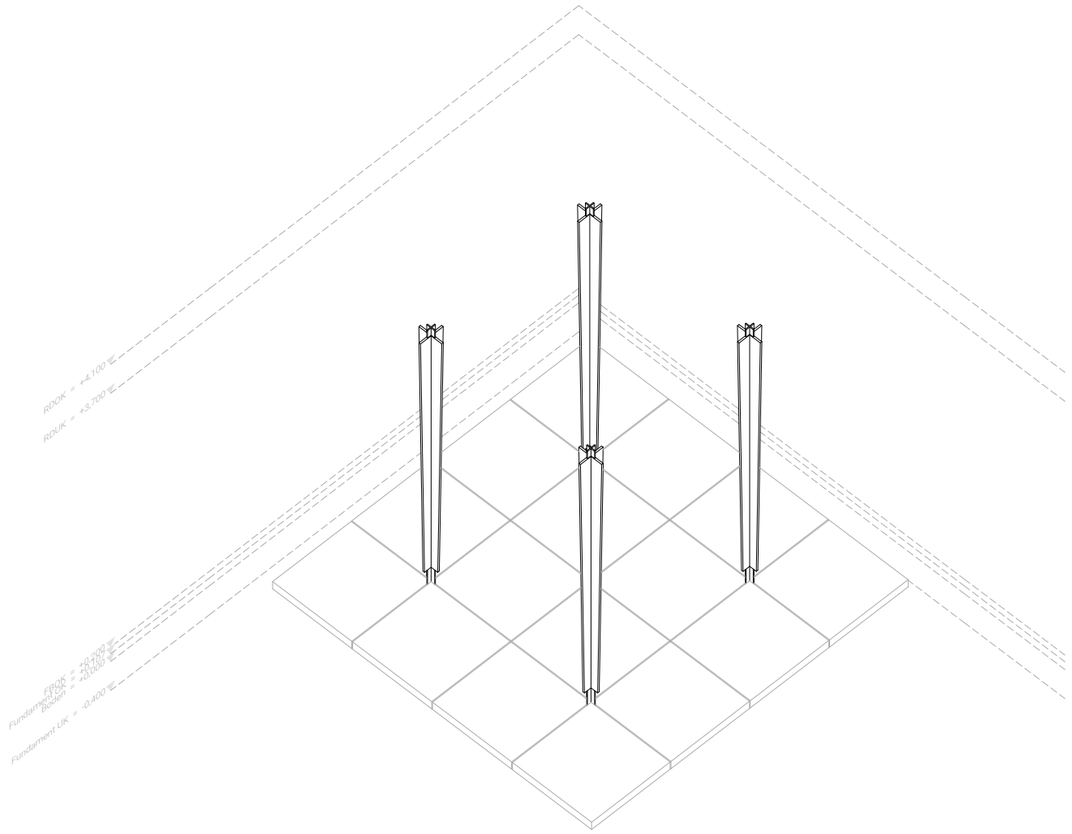


Abb. 6 „Leitfanden für die Komposition eines beliebigen Projektes“ Durands bestreben den architektonischen Entwurf zu standardisieren führen zur Bildung eines Rasters als Basis für Formgebung. (Picon, 2018: 35)



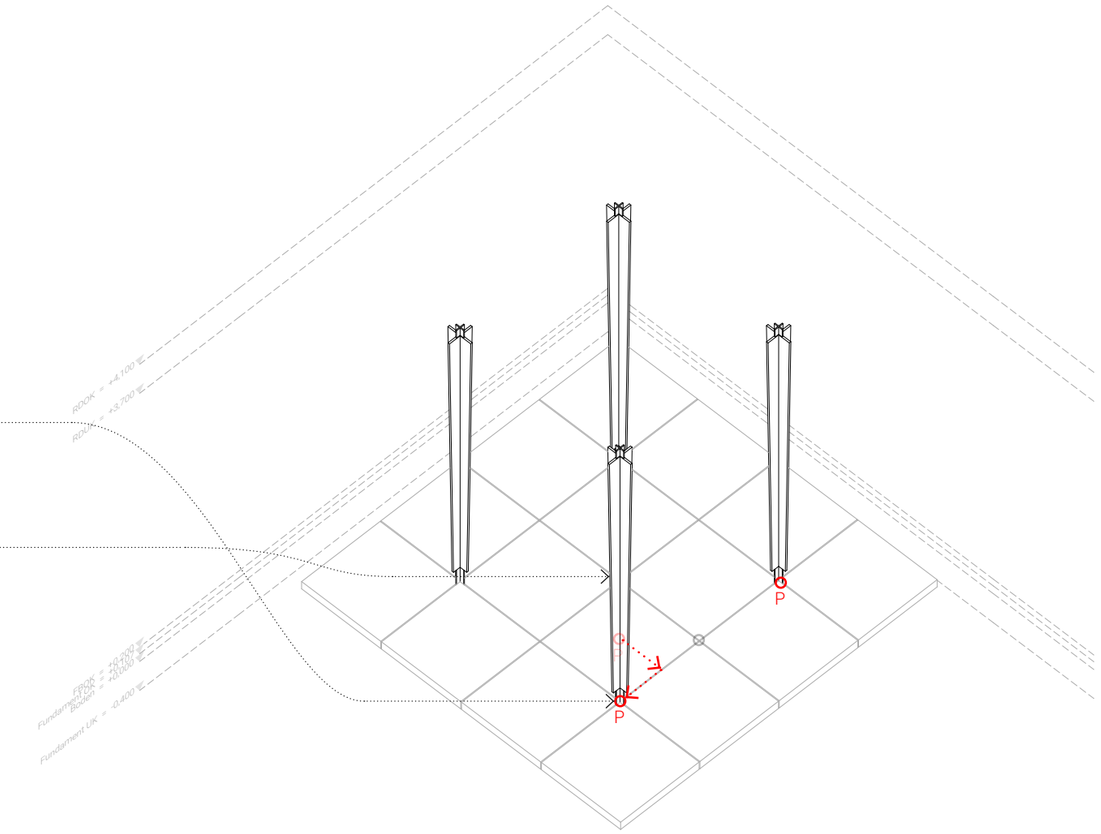
Auf Basis der zuvor generierten Punkte werden die Stützen platziert. Hier kommt eine neu definierte Variable hinzu. Es soll nur jeder X-Kreuzpunkt verwendet werden, um eine Stütze zu platzieren. Dies wird sowohl in Längs- als auch Querrichtung verwendet.

Für die Platzierung der Stützen werden nun die Ursprungspunkte der Platten um die Hälfte versetzt, um genau den Punkt an der Kreuzung zweier Fugen zu berechnen. Dieses neu erstellte Raster wird nun verwendet, um die Stützen zu platzieren.

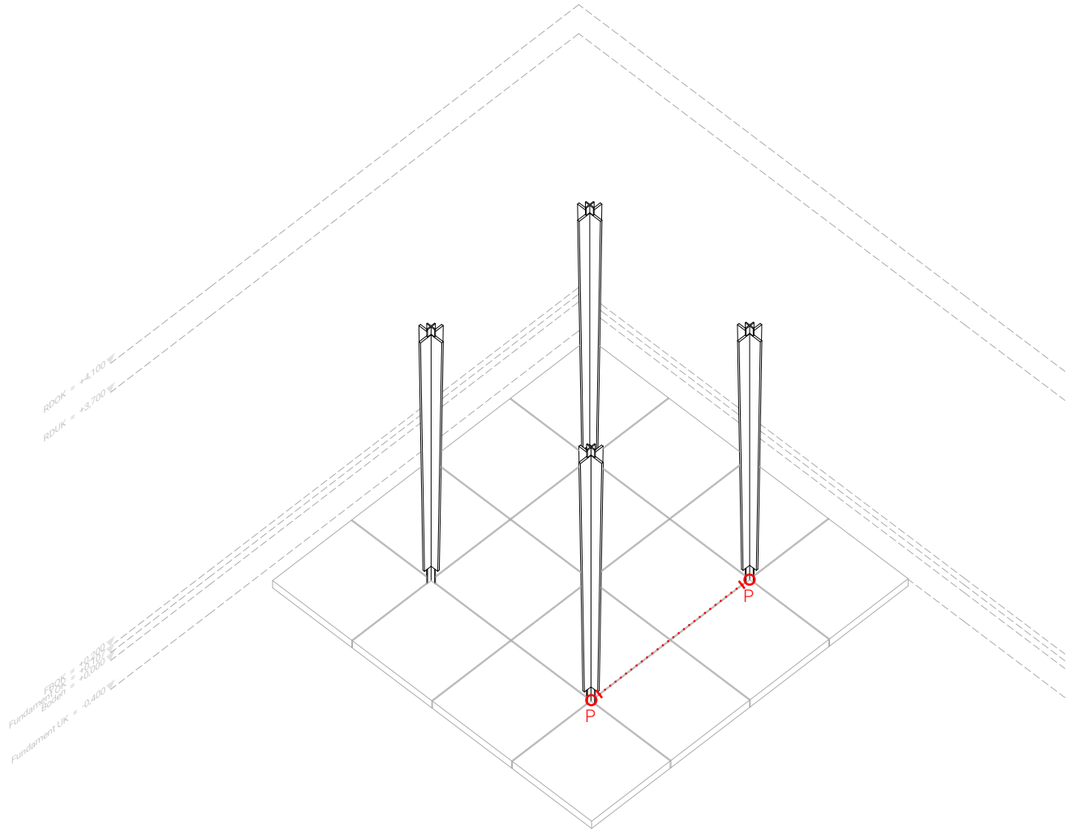


Code:

```
// Funktion zum ermitteln der Platten Größe
def calc_tile_size(totalLength,relativeLength) {
    return Math.Round(totalLength/ (relativeLength+1));
};

// Funktion zum berechnen der Punkte
def calc_columnPoint(start,end,distance,gap) {
    size = (distance+gap)*skip;
    return start+((size+gap)/2)..end+start..distance;
}

columnX = calc_coordinate(start,end,tileLength,gap);
columnY = calc_coordinate(start,end,tileWidth,gap);
points = Point.ByCoordinates(x,y,0);

columnFamily = Family.ByName(„column“);
columnType = FamilyType.ByFamilyAndName(
    columnFamily,“column“);
column = FamilyInstance.ByPointAndLevel(
    columnType,points,levelFBOK);

//Größe der Platte festlegen
Element.SetParameterByName(platten,“Length“,tileLength);
Element.SetParameterByName(platten,“Width“,tileWidth);
Element.SetParameterByName(platten,“Height“,tileHeigth);

//Höhenversatz von Referenzeben der Platte definieren
Element.SetParameterByName(platten,“Höhe von Ebene“,0);
```



Für die Berechnung der Spannweite werden die Fußpunkte der Stützen benötigt. Hierfür wird die Liste in vertikale und horizontale Punkte geteilt und anschließend der Abstand zwischen den zwei Punkten berechnet. Das Ergebnis wird anschließend für die Dimensionierung der Balken verwendet.

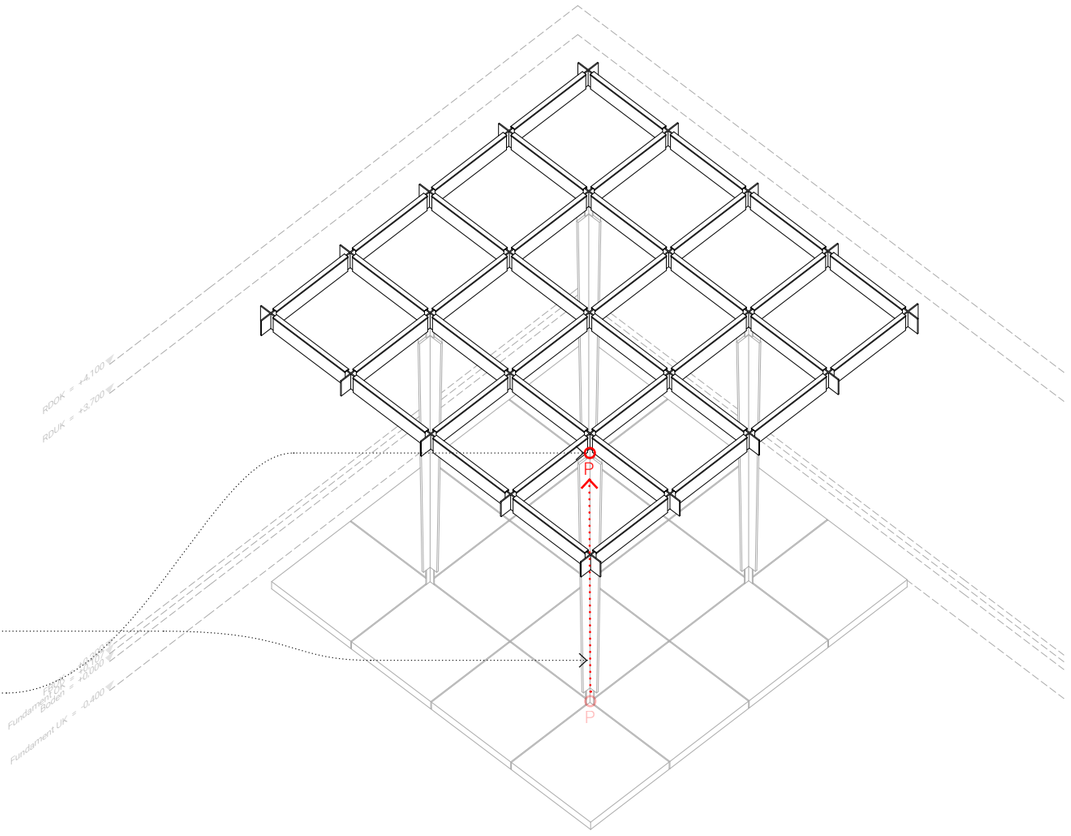


Nach demselben Prinzip der Stützen werden die Punkte für die Balken generiert. Die Punkte der Stützen werden um die Höhe des Raumes verschoben, um die Balken auf die räumlichen Positionen zu bringen. Hierfür wird auf die zuvor definierte Ebene zurückgegriffen.

Als Verbindung zwischen 4 Trägern und der Stütze wird ein Verbindungstück verwendet, das erlaubt, unterschiedliche Trägerhöhen aufzunehmen.

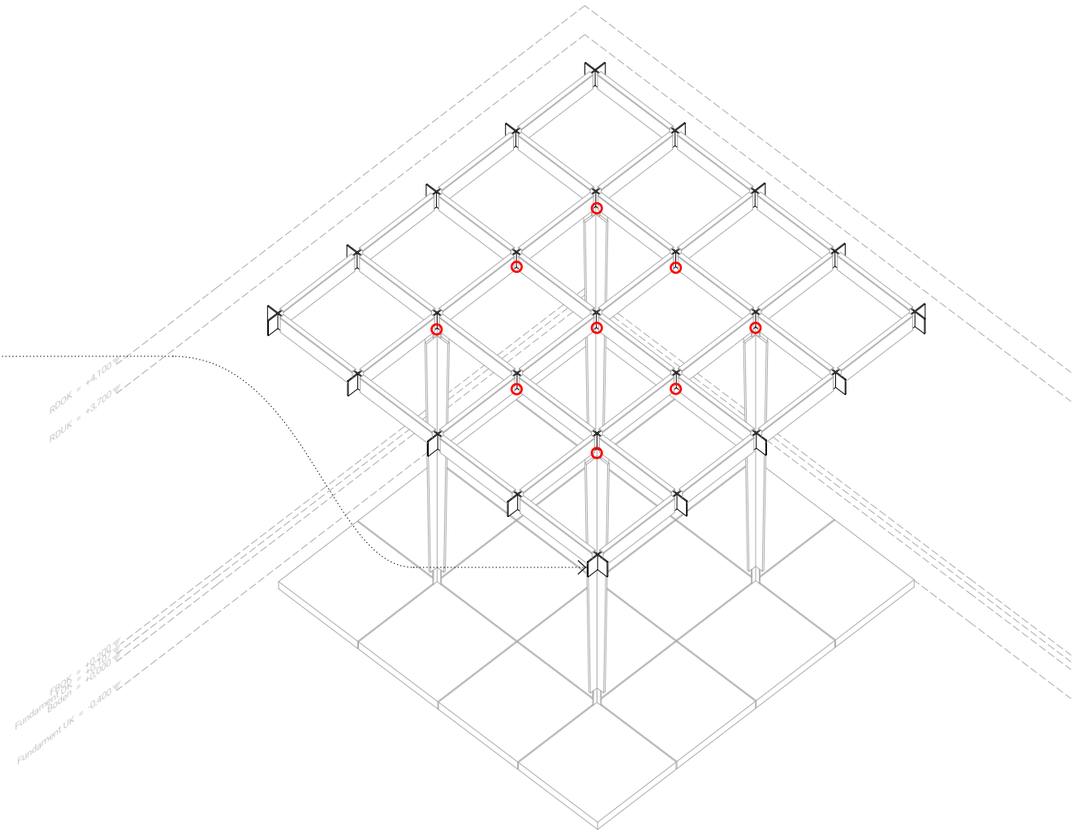
Code:

```
// Funktion zum berechnen der Punkte
def calc_roofPoints(start,end,distance,gap) {
     size = (distance+gap)*skip;
     return start+((size+gap)/2)-(distance+gap)
                    ..end+start+(distance+gap)
                    ..(distance+gap);
}

roofX = calc_roofPoints(start,end,distance,gap)
roofY = calc_roofPoints(start,end,distance,gap)
roofZ = Level.Elevation(LevelRDOK);

roofPoints = Point.
ByCoordinates(roofX,roofY,roofZ);
```



Code:

```
connectionFamily = Family.ByName(„connection“);
connectionType = FamilyType.ByFamilyAndName(
    connectionFamily,“connection“);
connections = FamilyInstance.ByPointAndLevel(
    connectionType, connectionPoints,levelRBOK);

//Eindeutige Bezeichnung jeder Verbindung vergeben
id = List.IndexOf(connectionPoints,connectionPoints);
Element.SetParameterByName(platten,“Kennzeichen“,id);

//Größe der Platte festlegen
Element.SetParameterByName(platten,“Length“,tileLength);
Element.SetParameterByName(platten,“Width“,tileWidth);
Element.SetParameterByName(platten,“Height“,tileHeigth);

//Höhenversatz von Referenzeben der Platte definieren
Element.SetParameterByName(platten,“Höhe von Ebene“,0);
```



Anders als bisher werden für die Träger zwei Punkte benötigt. Der Grund liegt auf der Hand: Träger sind gerichtete Bauteile. Diese lassen sich auf zwei Arten erstellen.

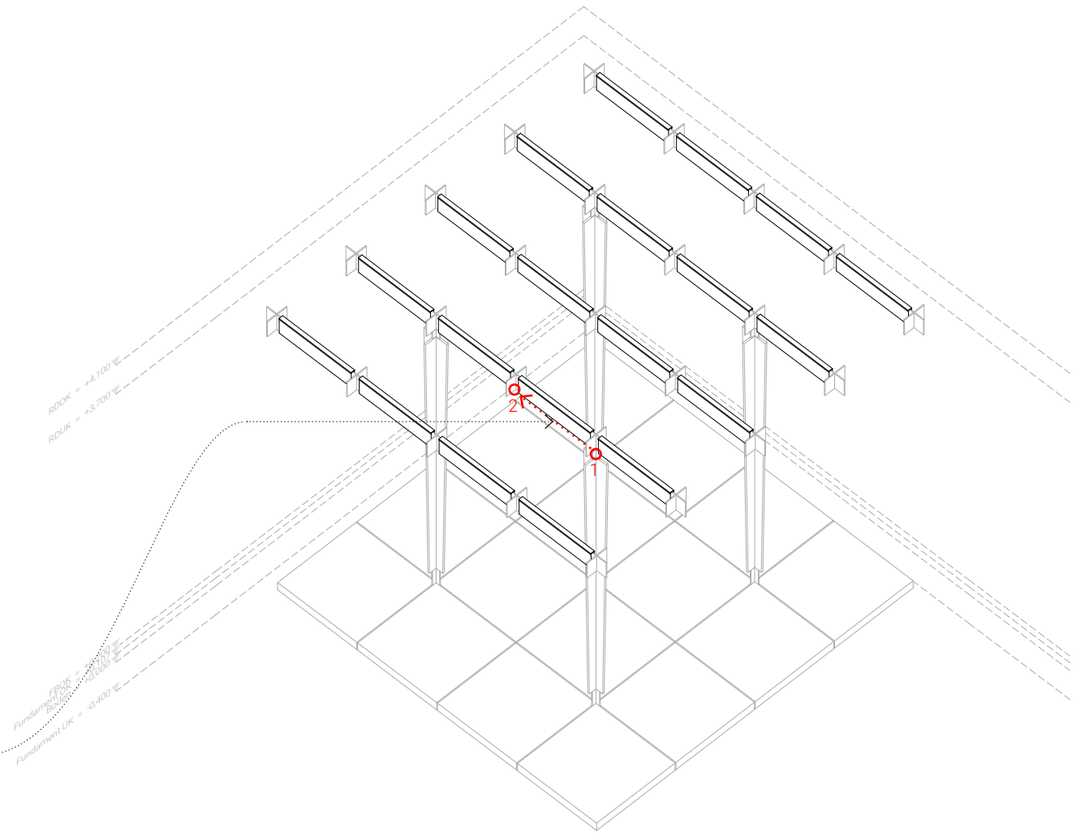
1.Durch Platzierung am Ursprungspunkt und anschließendes Drehen zum Erreichen der gewünschten Ausrichtung

2.Durch Erstellung eines Vektors, aus zwei Punkten, der somit die Ausrichtung des Trägers bestimmt.

Also wird ein Anfangs- und ein Endpunkt benötigt. Hierfür wird das Raster so aufbereitet, dass es jeweils eine Liste von Punkten in Querrichtung beinhaltet und eine Liste von Punkten in Längsrichtung.

Code:

```
def getVectors(points) {
      //Liste mit Anzahl index für Reihen
      cols = List.Count(points);
      index = 0..#cols-1..1;
      point_1 = list[index];
      point_2 = list[index+1];
      line = Line.ByStartPointEndPoint(
             point_1,point_2);

      //Anfang und Ende Kürzen
      line = Curve.ExtendStart(line, -5);
      line = Curve.ExtendEnd(line, -5);
      return line
}

vectors_1 = getVectors(points);
```



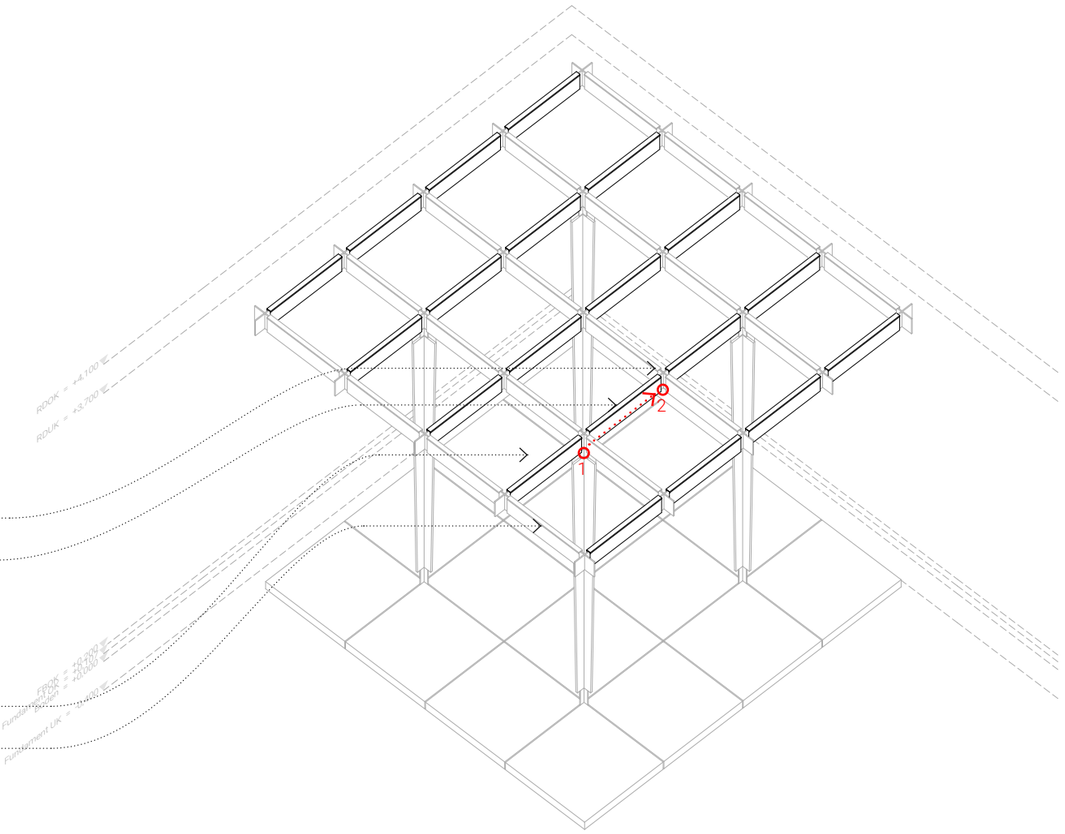
Ähnliches wird nun in Querrichtung durchgeführt.

Der Algorithmus wurde relativ allgemein geschrieben. Daher lassen sich auch andere „Rasterformen“ verwirklichen.

Die Höhe der Träger ist abhängig von der Spannweite. Hierfür wird der Abstand zwischen zwei Stützen berechnet und das Ergebnis für die Berechnung der Träger in Stützrichtung verwendet. Dabei werden alle Träger in der jeweiligen Reihe auf die größte ermittelte Höhe definiert.

Code

```
//Liste verwandeln um „Reihen“ mit „Spalten zu
tauschen
points_2 = List.Transpose(points);
//neue Liste für die funktion verwenden
vectors_2 = getVectors(points_2);

//platzieren der Träger
beamFamily = Family.ByName(„beam“);
beamType = FamilyType.ByFamilyAndName(
    beamFamily,“beam“);
beams_1 = FamilyInstance.ByCurve(
    beamFamily, vectors_1,levelRBOK);
beams_2 = FamilyInstance.ByCurve(
    beamFamily, vectors_2,levelRBOK);
```



```
room(length, width, height,
xTiles, yTiles, xSkip, ySkip)
```

Verwendung der Funktion zum erstellen eines Raumes

```
room(300, 300, 350, 2, 2, 2, 2)
```

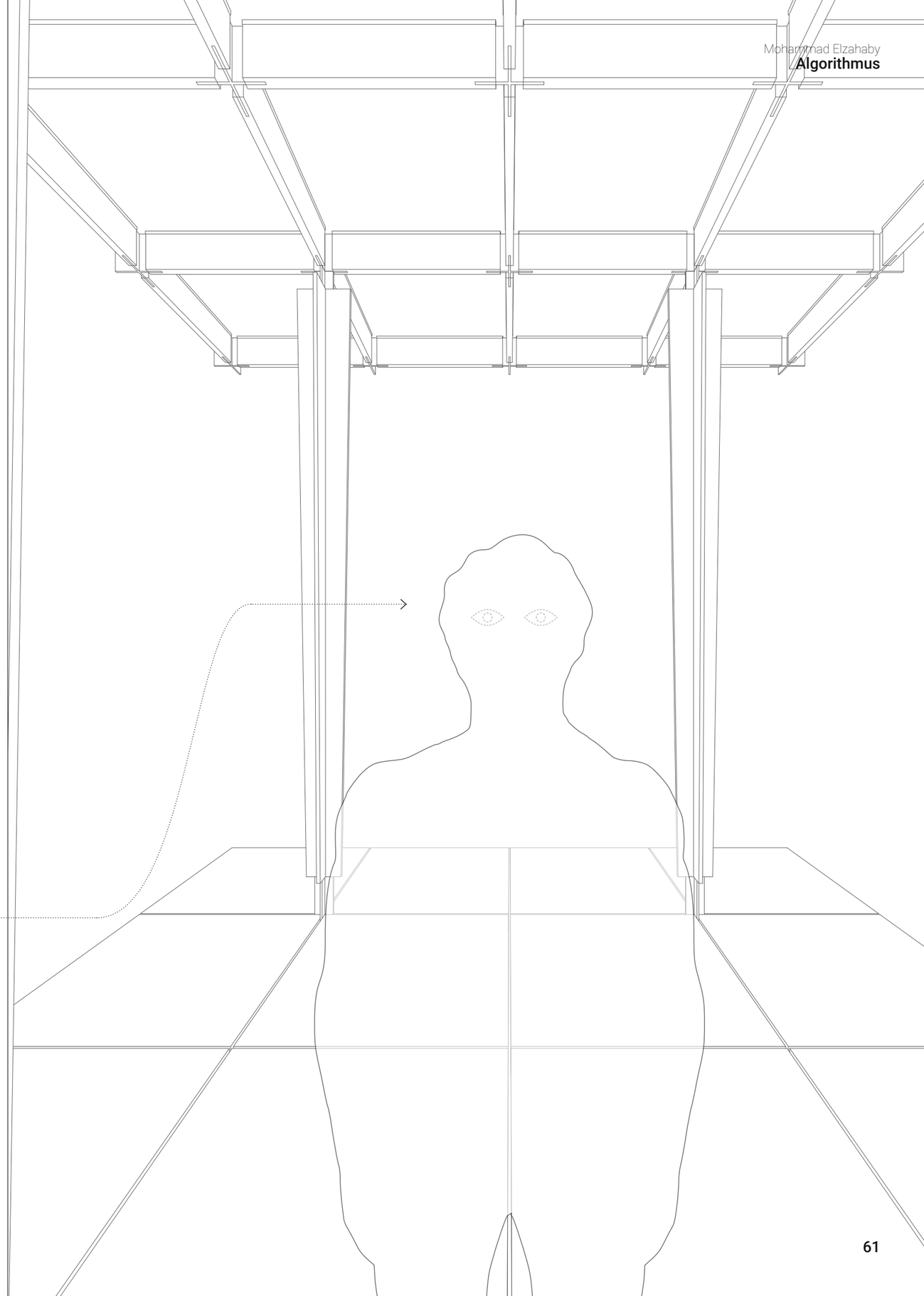
Dieses Zusammenspiel der Bauteile kann nun verwendet werden, um verschiedenste Varianten zu generieren. Hierfür wird eine Perspektive im Inneren des Raumes verwendet, um eine „Vergleichbarkeit“ zu schaffen.

Die Perspektive wird auch durch die Verwendung von Code generiert. Hierfür werden die Gesamtmaße des Bauwerks verwendet, um eine „Kamera“ mittig und am äußeren Rand einer Seite platzieren.

Da alle Teile des Bauwerks zusammenhängen, wandert auch die Kamera mit der Veränderung mit und muss nicht ständig angepasst werden.

```
// Walk 1 Meter into the space
position = 100;
x = width+(gap*(tile+1)) + width/(tile+1);
y = ((width+start)/2)+offset;
// Eye height;
z = 165;
eyePoint = Point.ByCoordinates(x, y, z);
targetPoint = Point.ByCoordinates(x, y, z);
// Get the Last Pillar
element = List.LastItem(List.Flatten(columns));
PerspectiveView.ByEyePointAndTarget(
    eyePoint,targetPoint, element, name,false);
```

Platzierung einer Kamera zum ersteellen der Perspektivischen Darstellung des Raumes



Genauso wie die Perspektive, lässt sich auch der Schnitt generieren:

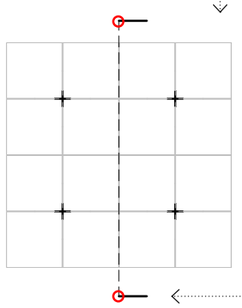
„Zeichnen" einer Schnitt linie

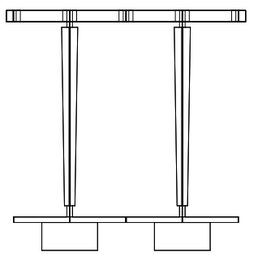
```
// Defining the start and end point
startPoint = Point.ByCoordinates(x, y, 0);
endPoint = Point.ByCoordinates(x, y+length, 0);
line = Line.ByStartPointEndPoint(
    startPoint, endPoint);
vN = Vector.Normalized(Line.Direction(line));
v Y             =             V e c t o r . Z A x i s
cs = CoordinateSystem.ByOriginVectors(
    startPoint,
    vN,
    vY,
    Vector.Cross(vN,vY));
//Creating an A3 Sized Window
minPoint = Point.ByCoordinates(-4200/2,-2970/3,0);
maxPoint = Point.byCoordinates(4200/2,2970/3*2,0);
section = SectionView.ByCoordinateSystem(
    cs, minPoint, maxPoint);
Element.SetParameterByName(
    section,"Ansichtsname","Schnitt");
```

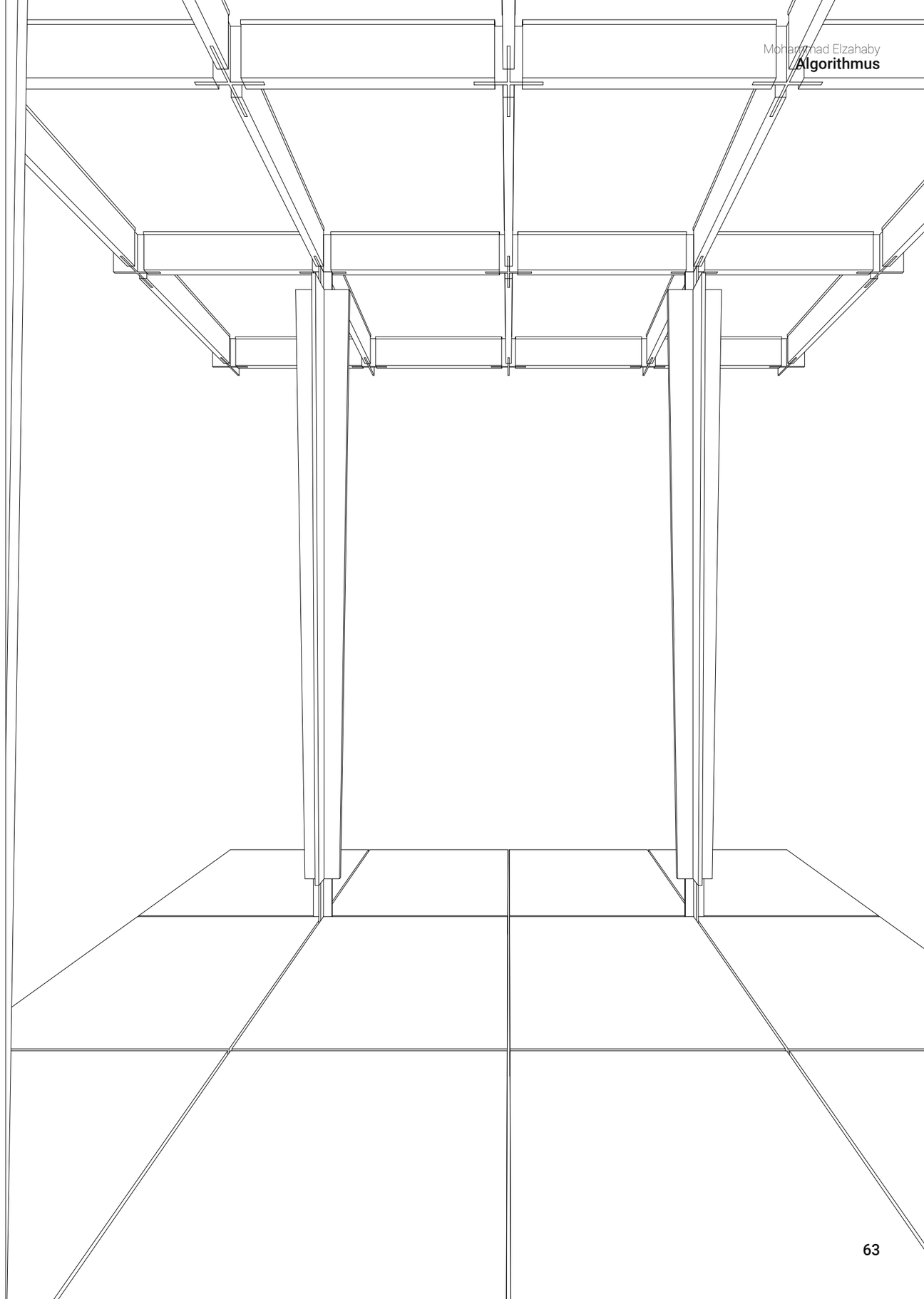
Erstellung eines Schnittes in der Mitte des Raumes entlang der Schnittlinie

Mit dieser Herangehensweise lässt sich auch der Grundriss erstellen und sämtliche weitere für die Vermittlung des Entwurfs notwendigen Darstellungen.

```
// Code zum erstellen eines Grundriss aus Ebene
FloorPlanView.ByLevel(FBOK);
```







Source Code

Dieser Algorithmus bildet den gesamten Code des Pavillons. Also, die Anweisungen für den Computer zum Fügen und Manipulieren der einzelnen Bauteile. Es wird keine „Geometrie" geschaffen.

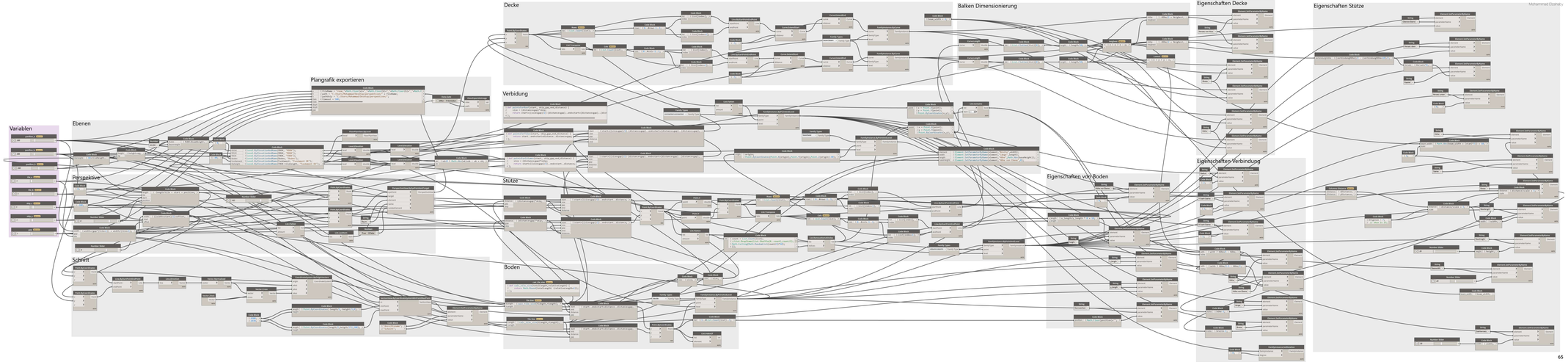
Gestaltung kann in zwei Arten unterteilt werden, dem Zusammenfügen von Bauteilen; Tektonik oder dem Formen von Geometrie, Plastik. Beides hat seine Legitimität. Jedoch wirkt sich diese grundsätzlichen Herangehensweisen unterschiedlich auf dem Entwurf aus.

Zum einem wird von einer Masse ausgegangen, die frei formbar ist. Diese Entwicklung der Formfindung ist Koralle zweier Technologien zu verdanken. Dem Stahlbeton und dem Parametrismus der es erlaub diese „freien" Formen, Herr zu werden.

Bei der Plastik kann ganz auf alle Verbindungen verzichtet werden. Hier wird die Form in den elementaren Bestandteilen, die wir als Boden, Wand, Decke wahrnehmen, nahtlos ineinander übergehen kann. Dieses wird oft als Freiheit empfunden, mit ihr geht aber gleichzeitig die Möglichkeit einher, komplexe Formen zu schaffen die wiederum den Computer zum Denken benötigen.

Andererseits werden die vorher gedachten oder aus Erfahrung vorstellbare Bauteile zu einem großen Ganzen gefügt. Hierbei entstehen Verbindungen, die gelöst werden sollten.
Dieses tektonische zusammenfügen von Bauelemente und Lösen von Verbindungen, wird mit dem Code bewältigt.

Mohammad Elshahby



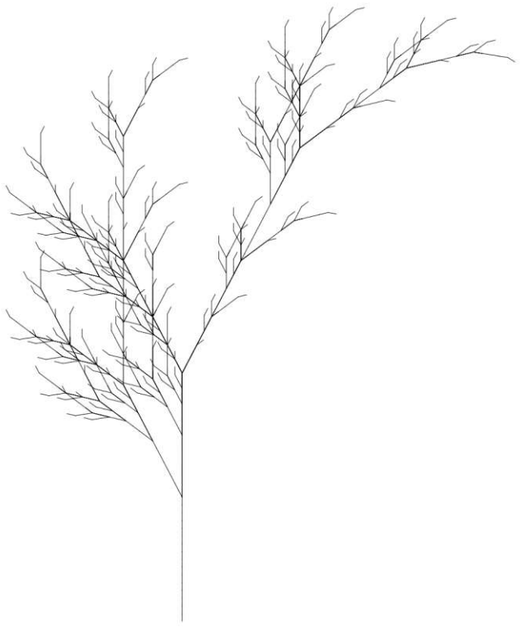
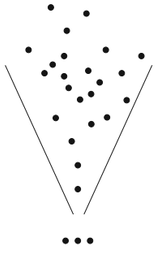


Abb. 7: Demonstration der Anwendung eines Lindenmayer-System Algorithmus:
1: X=C0F-[C2[X]+C3X]+C1F[C3+FX]-X, 2: F=FF, Axiom: X, Kosntante: X, Iteration: 5, Winkel: 25



Das Evaluieren kann wie ein Sieb sein. Der Algorithmus kann so die Ausgabe auf das Wesentliche reduzieren.

Variation

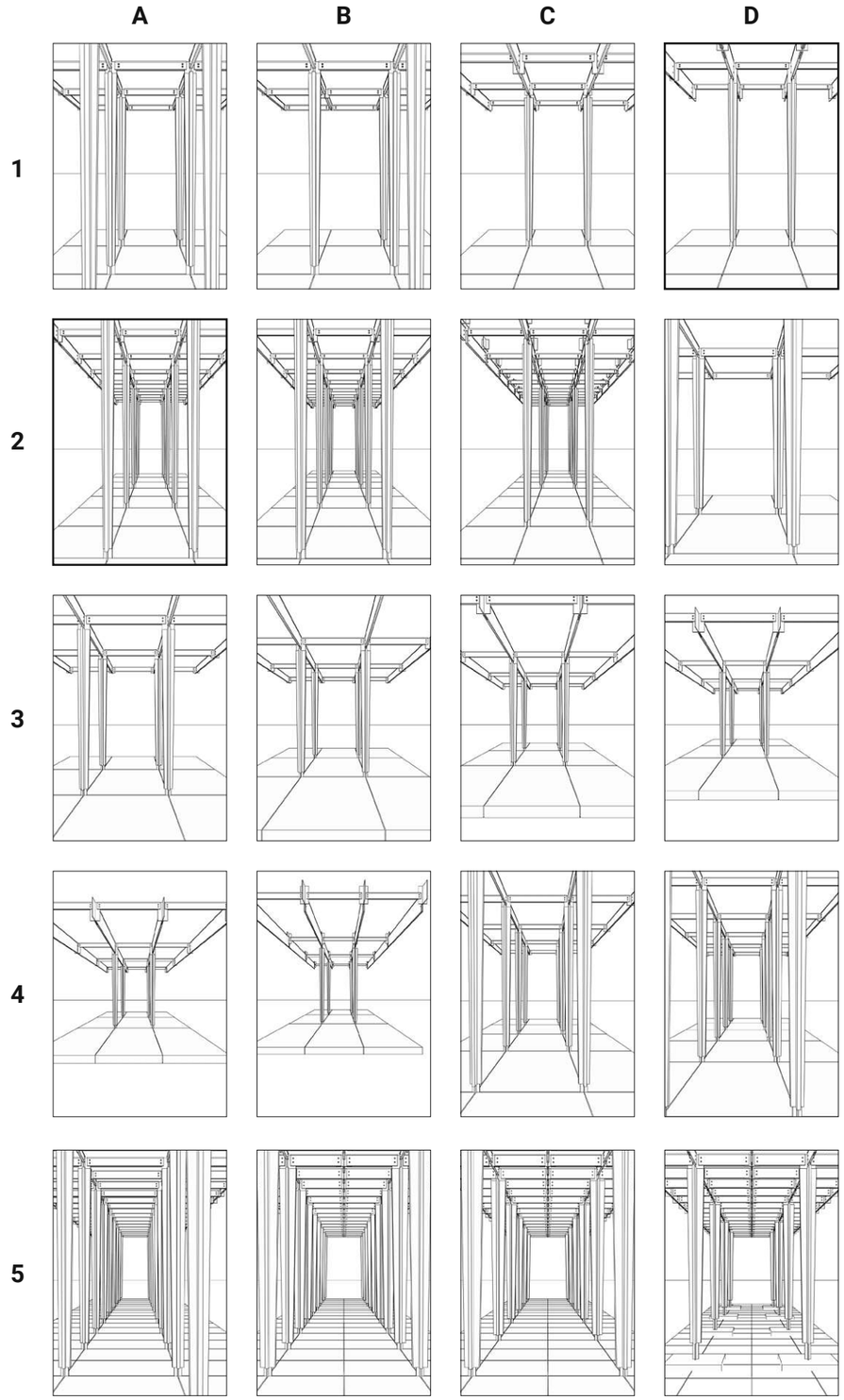
> *„Der Ausgang einer algorithmischen Prozedur ist offen. Der Input kann bekannt sein, die Handlungsanweisung kann bekannt sein, aber das Ergebnis, der Output, kann vorerst unbekannt sein“* (Wolf-Plottegg 1998)

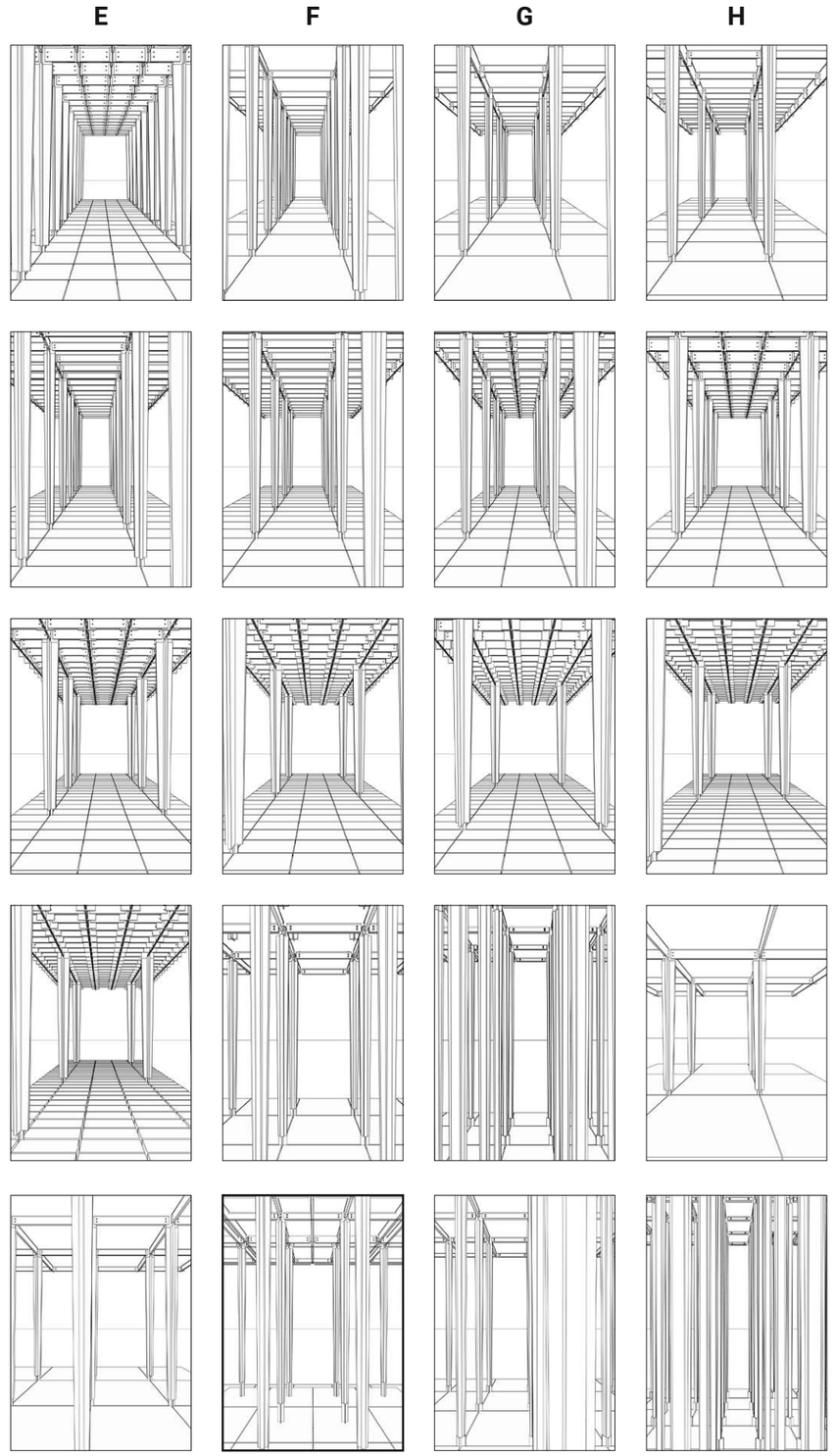
Auch wenn der Algorithmus eine klar definierte Vorgangsweise beschreibt, kann dieser dennoch zu unverhofften Resultaten führen. So ermöglicht das Arbeiten mit Algorithmen nicht nur die Erfassung von mehr Informationen in der Entwurfsphase, sondern auch potenziell auf Lösungen zu kommen, die jenseits des Denkbaren sind.

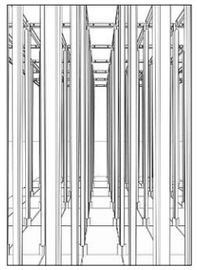
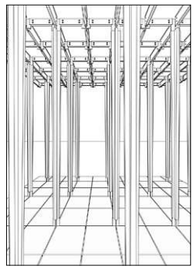
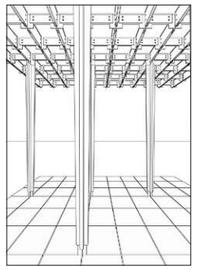
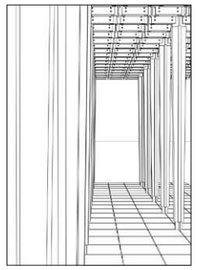
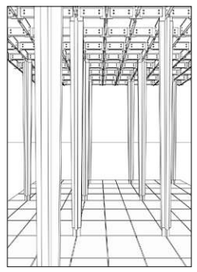
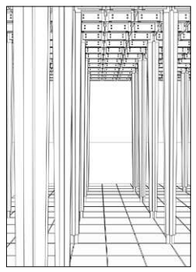
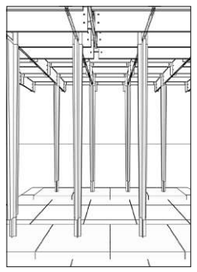
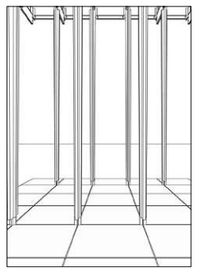
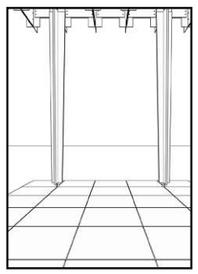
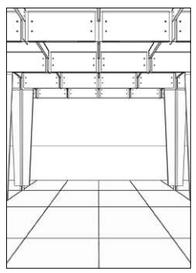
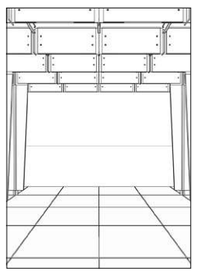
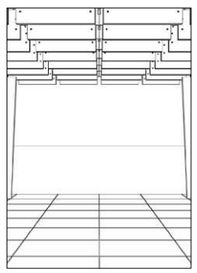
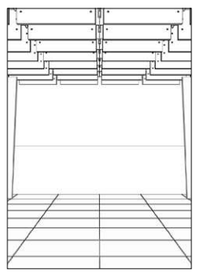
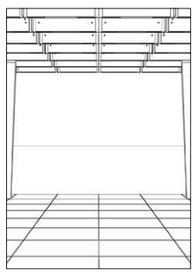
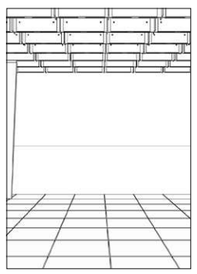
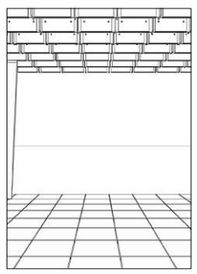
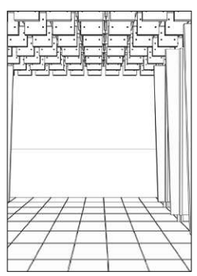
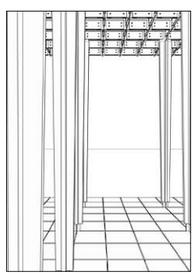
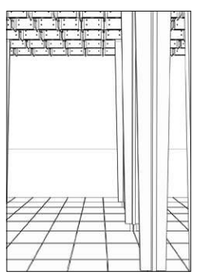
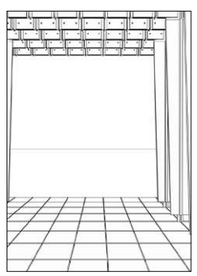
Durch dieses schnelle Generieren von Varianten lassen sich schneller und einfacher verschiedene Möglichkeiten entdecken, die dabei stets alle Regeln der Prozedur einhalten.

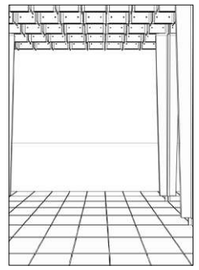
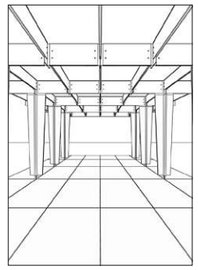
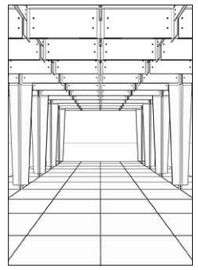
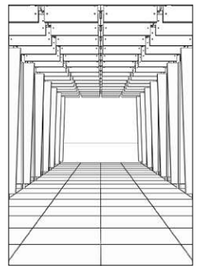
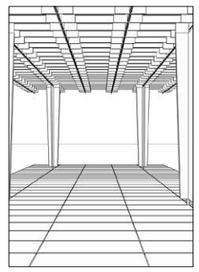
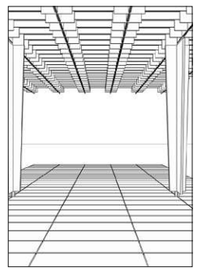
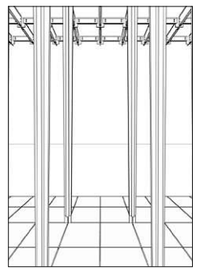
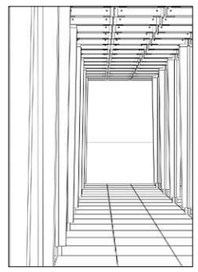
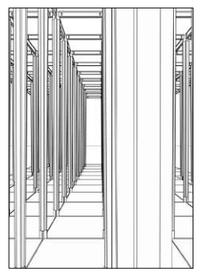
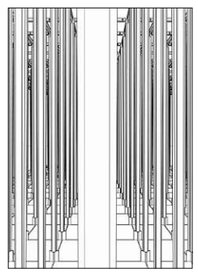
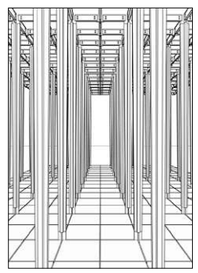
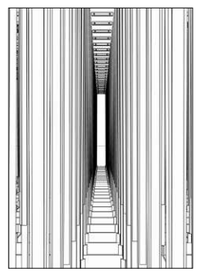
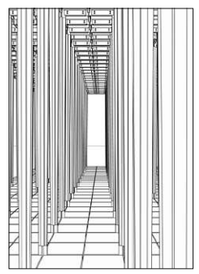
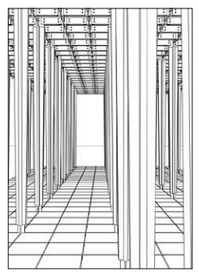
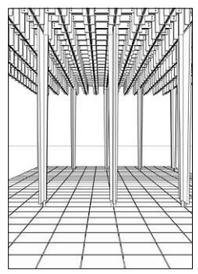
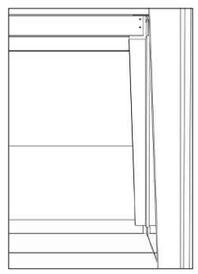
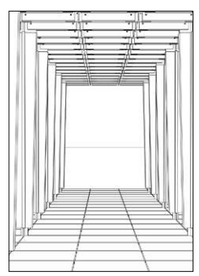
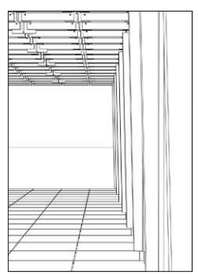
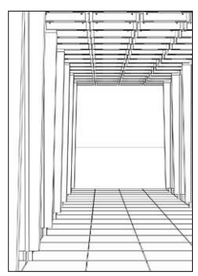
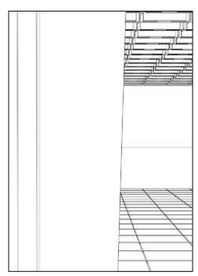
Zunächst werden unzählige Varianten generiert. Die räumlichen Darstellungen gestatten eine vergleichende Methodik. Aus diesen Varianten wird auf Basis der Intuition eine Auswahl von Räumen getroffen, die in ihren Merkmalen unterschieden und beschrieben werden.

Beim Generieren verschiedener Varianten entstehen auch Versionen, die redundant oder in ihrer Ausformulierung inadequat sind. Mit einer Optimierung des Algorithmus kann das verhindert werden, jedoch entfällt dadurch auch die Flexibilität. Somit werden die potentiellen Ausformulierungen eingegrenzt.





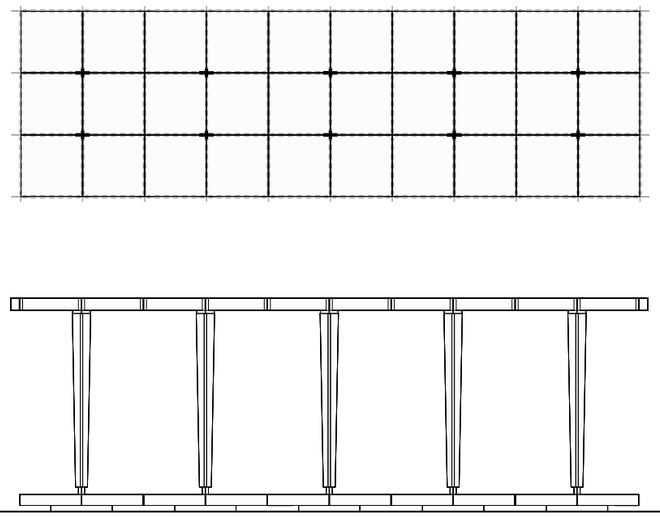
	I	J	K	L
1				
2				
3				
4				
5				

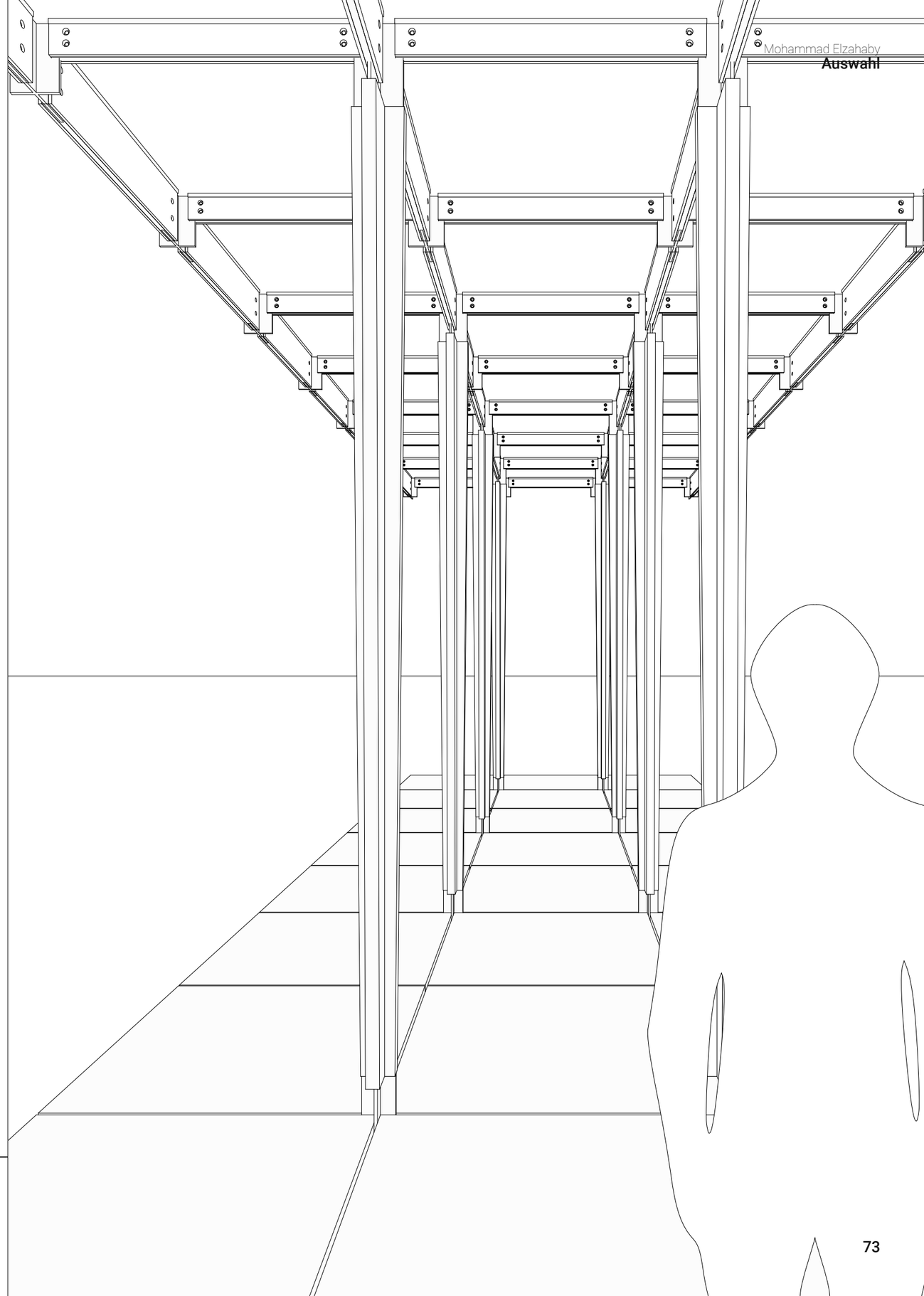
	M	N	O	P
1				
2				
3				
4				
5				

`room(200, 900, 300, 1, 8, 1, 2, 1)`

Raum A-2

Dieser Raum zeichnet sich durch seine Länge aus. Seine Gangartigkeit erlangt er durch das hohe Verhältnis von Länge zu Breite. Eine Dichte von 2:1 für die Anzahl der Stützen verstärkt die Wegeführung. Die Stützen führen direkt nach vorne. Durch die eindeutigere Steuerung der Raumproportion in eine Richtung entsteht ein Korridor, der die Bewegung in vorgibt.

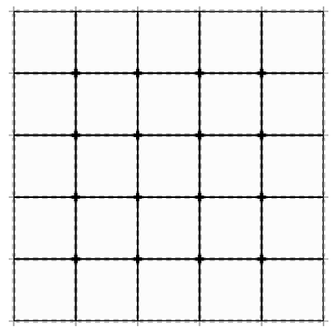


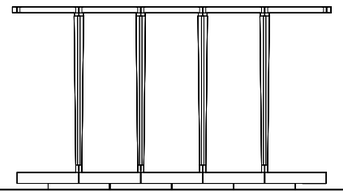


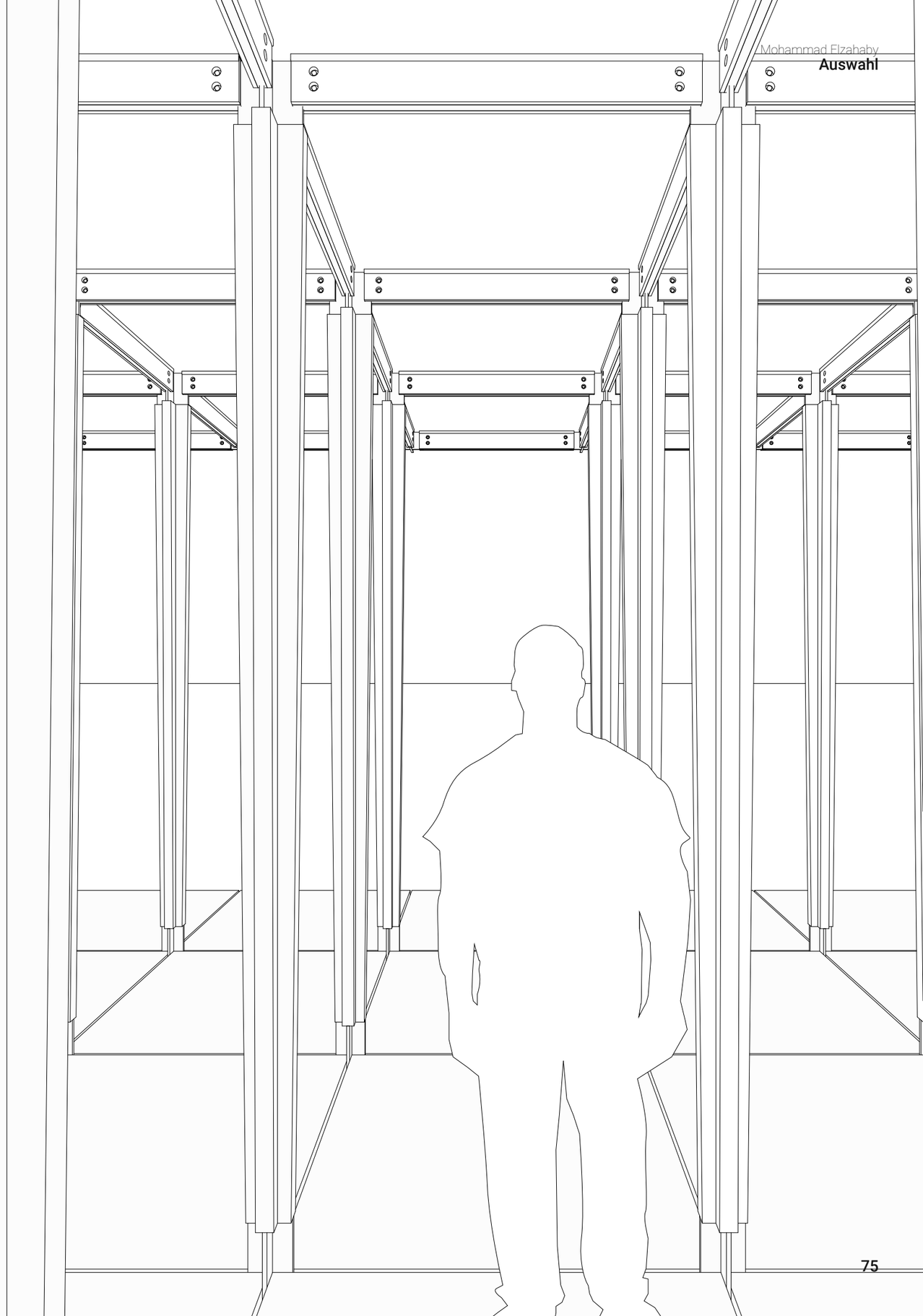
```
room(400, 400, 300, 3, 3, 1, 1, 1)
```

Raum F-5

Die Hohe dichte der Stützen zeichnet diesen Raum aus. Er verlangt nach filigranen Stützen. Eine Knappheit des Raumes entsteht.
Durch die regelmäßige Verteilung der Stützen entsteht ein nicht gerichteter Raum. Dieser gibt keine Bewegungsrichtung vor und führt somit zum Flanieren.



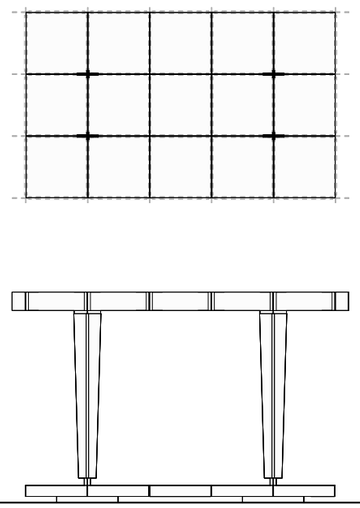


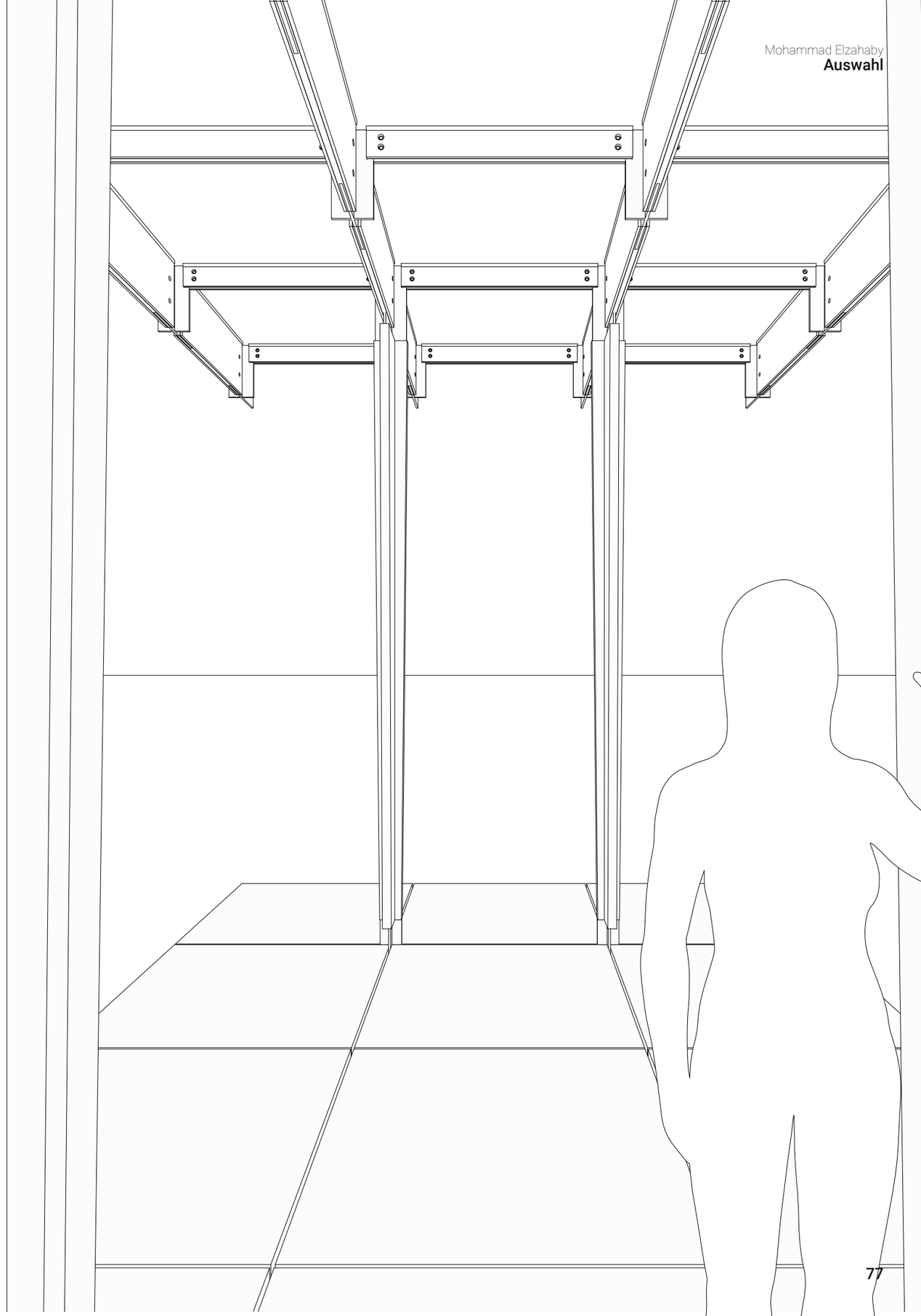


```
room(200, 400, 285, 1, 3, 1, 3, 1)
```

Raum D-1

Mit einem Verhältnis von 1:2 wird der Raum weitläufig und einladend. Die Richtung lässt sich durch die Stützen artikulieren. Diese ist mit einer Dichte von 1:3 so angeordnet, dass diese einen Räumlichkeit zwischen den tragenden Bauteilen bildet. Durch die unterschiedliche Ausbreitung entsteht eine subtile, aber doch wahrnehmbarer „Fluss“. Jedoch lässt sich durch die Weite der Drang zum Verweilen spüren.



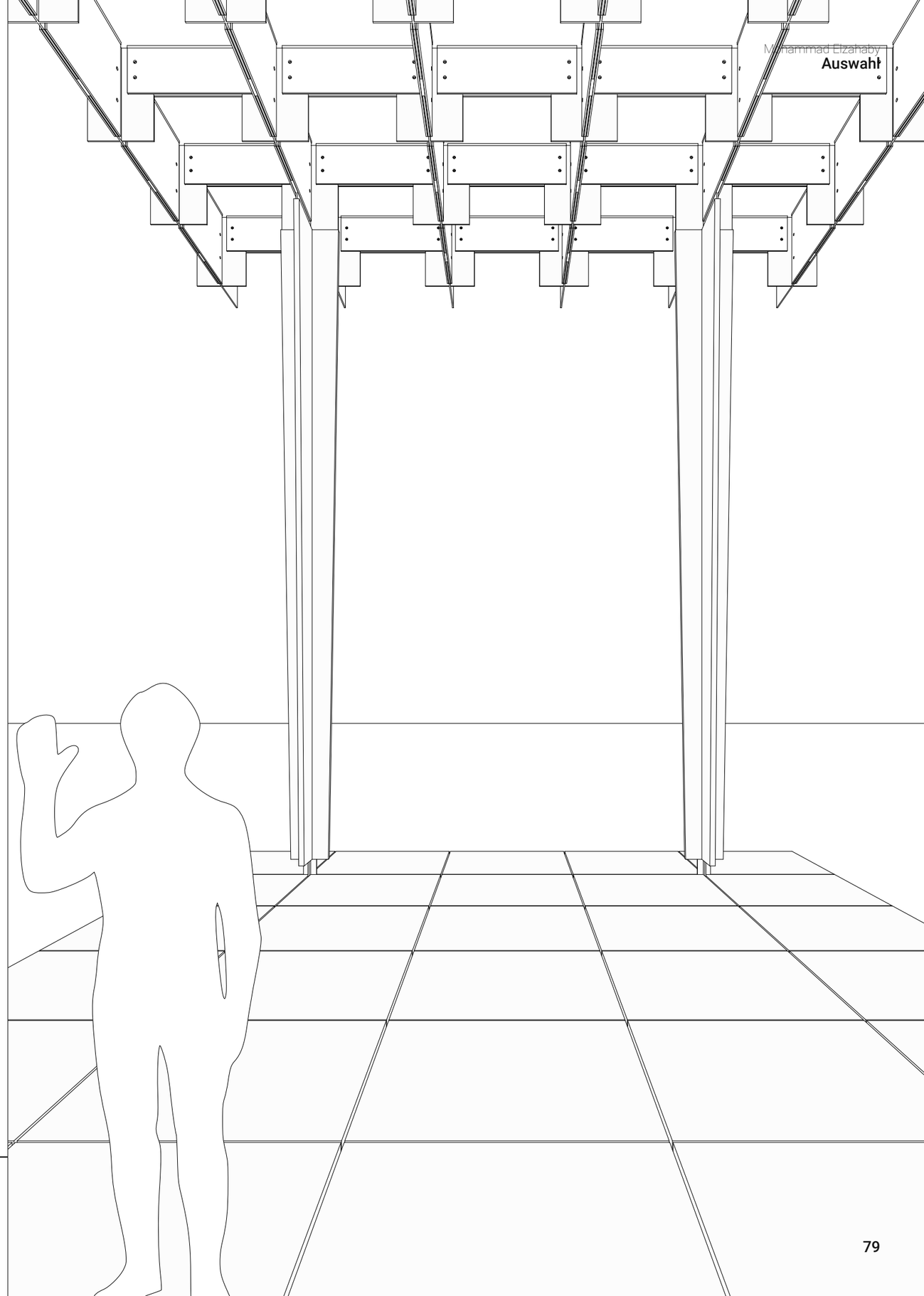


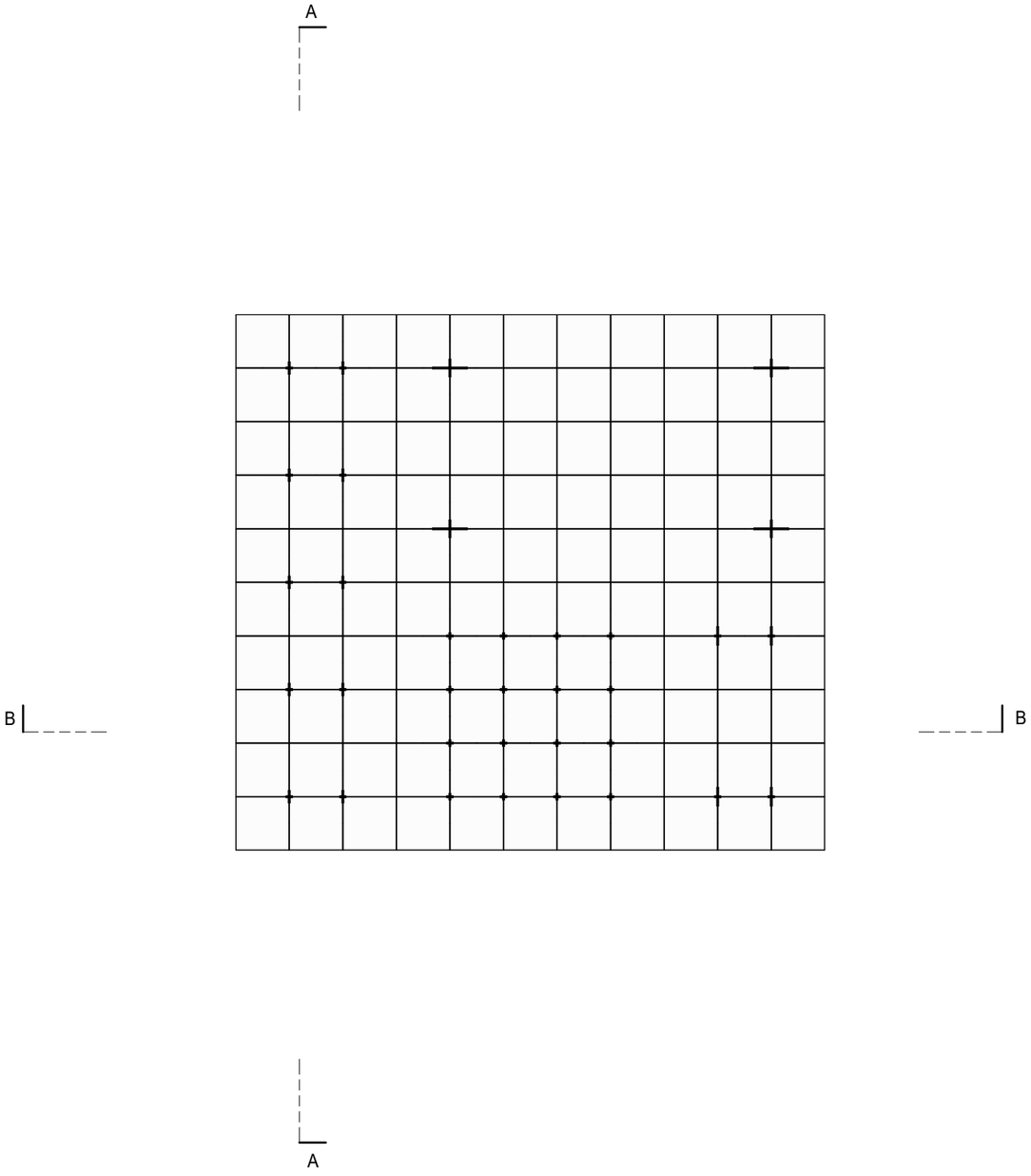
```
room(400, 700, 500,3, 6, 3, 1)
```

Raum I-3

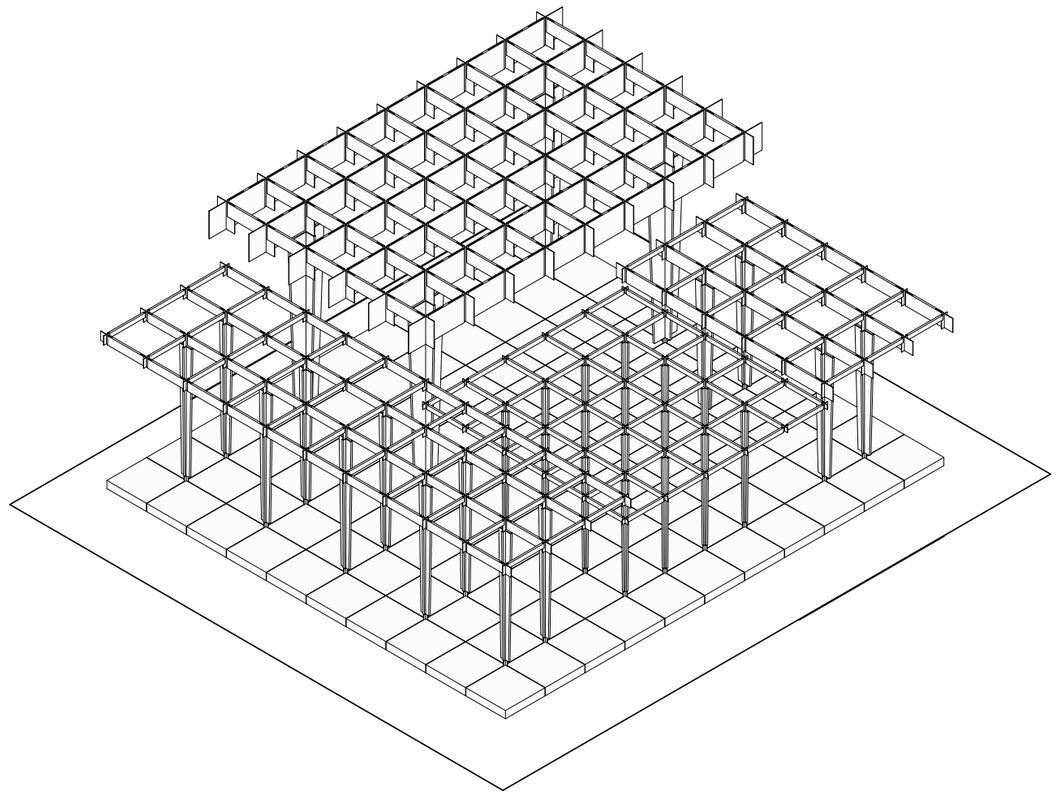
Durch die hohe Spannweite erweitern sich die Stützen zu längliche Bauteile, die wie das Relikt einer Wand wirken. Auch wenn der Raum mit seinem Verhältnis von 4:7 und großen Abständen zwischen den tragenden Bauteilen sehr weitläufig wirkt, geben die Länglichkeit der Stützen eine Richtung vor. Hier bekommt man den Eindruck in einer Halle zu sein.

1:100
0 1



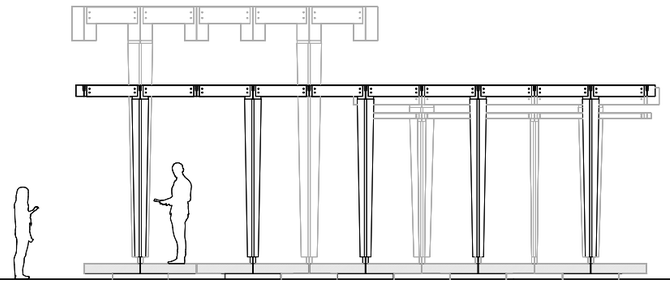


Grundriss

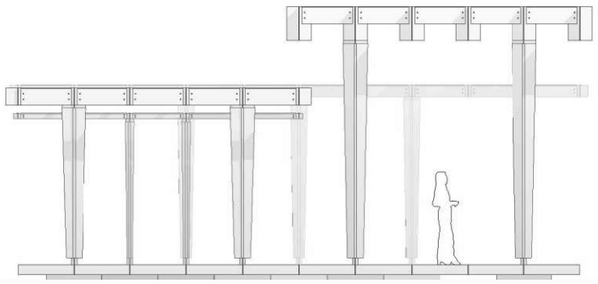


Axonometrie

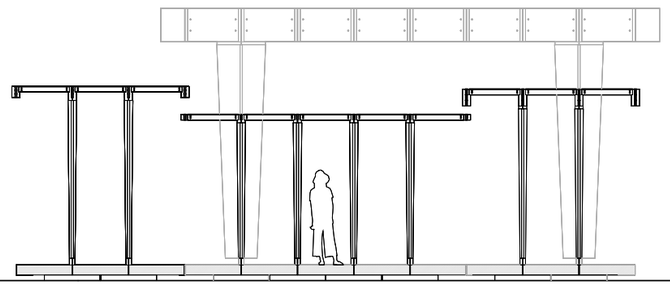
1:100
0 1



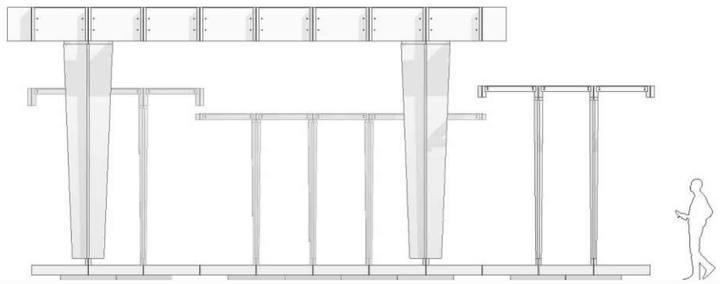
Schnitt A-A



Ansicht Nord



Schnitt B-B



Ansicht West

1:100
0 1

1:100
0 1

Conclusio

Es wird eine kurze Einführung in die Entwicklung des Computer Aided Design (CAD) im Laufe der Geschichte geschildert sowie die aktuellen Entwicklungen im Bereich des computergestützten Entwurfens und die Verwendung von Code als Mittel zur Automatisierung des Entwurfs dargestellt. Die Methode der Verwendung von Code in Kombination mit BIM anstelle eines manuellen Prozesses zur Erstellung eines Entwurfs wird erläutert. Dies wird an einem Design-Prototyp unter Verwendung der Software Umgebung Revit für den BIM-Aspekt sowie dem integrierten Tool Dynamo für den Code angewendet. Innerhalb von Dynamo wird sowohl die visuelle als auch die textuelle Programmiersprache DesignScript kombiniert verwendet. Der Algorithmus hinter dem Design wird im Detail erklärt. Mit dem fertigen Design-Algorithmus werden verschiedene Design-Studien auf ihre räumlichen Charakteristika untersucht, woraufhin distinktive Resultate des Algorithmus verwendet werden, um sie zu einer größeren räumlichen Konfiguration zu kombinieren. Die Ausführung des Skriptes bedarf einer Wartezeit von ca. 10 Sekunden, bei wiederholter Anwendung mit veränderten Variablen, bis hin zu 20 Sekunden. Das lässt sich auf mehrere Gründe zurückführen. Zum einem ist die Software-Umgebung nicht für solch eine Anwendung ausgelegt zum anderen kann Optimierung am Algorithmus zu einer Verbesserung der Laufzeit beitragen.
Obwohl der Algorithmus zur Automatisierung des Entwurfsprozesses verwendet wird, ist die gewonnene Effizienz, das schnelle wiederholte Überprüfen von Varianten bei einem Entwurf in kleinem Maßstab nicht merklich. Dennoch können einige Teile des Algorithmus in
verschiedenen Entwürfen mit unterschiedlichen BIM-Objekten oder Anwendungsfällen verwendet werden. Eine weitere Untersuchung der Methode für den Einsatz bei komplexeren Entwürfen sollte durchgeführt werden. Dazu könnte eine eigens für die Methode entwickelte und darauf ausgerichtete Programmiersprache, Framework oder Anwendungsumgebung hilfreich sein.
Weiters sollte nur das Prinzip des Auflösen eines Bauwerks in einzelne Objekte um damit ein Bauwerksmodell auf zu bauen jedoch sollen auch diese Objeekte programmierte Module sein die durch den Algorithmus aufgerufen werden können. Diese können dadurch ihr Verhalten selbst bestimmen und an den Algorithmus wieder geben. In der Arbeit wurden die Objekte auf Grund der Art und Weise wie die Anwendungsumgebung fungiert, nachträglich die Eigenschaften der Objekte definiert, was auch mit Nachteilen einhergeh So muss bei der Manipulierung des Objekts bereits der Endzustand des Objekts berechnet werden, bevor dieses mit Informationen befüllt wird und somit müssen alle Berechnungen bereits im Algorithmus passieren. Idealerweise berechnen die Objekte ihr Verhalten eigenständig und der Algorithmus definiert lediglich die tektonischen Zusammenhänge.
Das Coden in diesen Zusammenhang hat das potential das Entwerfen von Grund auf neu zu denken. Zum einen wird mit Punkten das Zusammenfügen von Bauteilen bewerkstelligt. Daraus resultiert gezwungenermaßen das Auflösen des Entwurfs in Punkten. Zum anderen wird durch die Möglichkeit unzählige Varianten zu erstellen diee Rolle des Entwurfs zu der eines Kurators der die Auswahl beschränkt und auch trifft. Man könnte meinen das der Architekt schon immer auch ein Kurator war der zwischen verschiedene Imaginationen eine Auswahl trifft die letzendlich Gebaut wird. Durch das algorithmische Entwerfen kann auch gleich Evaluiert werden und die Auswahl auf die für die Funktion Adequate Varianten beschränken.
Auch wenn sich viele Vorteile durch das Arbeiten mit Code ergeben, neigt dieser sehr zu beschränken. Sobald ein Algorithmus verfasst wurde ist es schwer aus dieser Box heraus zu kommen. Innerhalb gibt es noch unzählige möglichkeiten. Es bleibt jedoch die selbe Büchse.

Abbildungen

Abb. 1: Bild des Titelblattees des: Essai sur l'Architecture 2nd ed. 1755 by Charles Eisen (1720-1778).

Abb. 2: Digitale Analogie der Urhüte

Abb. 3: Diagram zur gegenüberstellung von linearen Prozessabfolge und Simultanes

Abb. 4: Darstellung eines L-System erstellten Bild https://onlinemathtools.com/l-system-generator

Abb. 5: Deutscher Pavillon, Barcelona, 1929, Querschnitt der Stütze (nach: Kenneth Frampton, Grundlagen der Architektur. Studien zur Kultur des Tektonischen, München und Stuttgart 1993, 191)

Abb. 6: „Leitfaden für die Komposition eines beliebigen Projekts" von Jean-Nicolas-Louis Durand aus
Précis des leçons d'architecture données à l'École polytechnique (Kompendium der Architekturvorlesungen an der École polytechnique), Paris 1802–05, Nachdruck von 1813 https://www.researchgate.net/publication/330409289

Abb. 7: Darstellung eines L-System erstellten Bild https://onlinemathtools.com/l-system-generator

Literatur

buildingSmart (2022) buildingSmart [Online]
https://www.buildingsmart.co.at
[Abgerufen am 03.04.2022]

Brnic, I. (2013)
Was the primitive hut actually a temple? The impact of recent archaeological excavations on the architectural theory of the primitive hut, in: San Rocco, Nr.8, S. 32–43, [online] https://publik.tuwien.ac.at/files/PubDat_247730.pdf [abgerufen am 05.05.2022]

Brnic, I. (2019)
Nahe Ferne, Sakrale Aspekte im Prisma der Profanbauten von Tadao Ando, Louis I. Kahn und Peter Zumthor
Park Books

Campbell, D. A. (2006) Modeling Rules, [Online] Available at: http://www.architectureweek.com/2006/1115/tools_2-1.html [abgerufen am 12.04.2022]

Eastman, C.; Teicholz, P.; Sacks, R.; Lee, G. (2018)
BIM Handbook: A Guide to Building Information Modeling for Owners, Designers,
Engineers, Contractors, and Facility Managers
New Jersey, USA: John Wiley & Sons, Inc.

Ferreira, B.; Leitão, A. (2015) Generative Design for Building Information Modeling, In:
eCAADe Conference [Online] http://cumincad.scix.net/data/works/att/ecaade2015_118.content.pdf [abgerufen am 05.05.2022]

Grasshopper (2022) Grasshopper [Online]
https://grasshopper.app/why-coding/
[Abgerufen am 03.04.2022]

Humppi, H.; Österlund, T. (2016) Algorithm-Aided BIM
BIM | Concepts - Volume 2 - eCAADe 34 | 603 [Online] http://papers.cumincad.org/data/works/att/ecaade2016_158.pdf

Jabi, W. (2013)
Parametric Design for Architecture
Laurence King Publishing Ltd

Koolhaas, R. (2018)
Elements of Architecture
taschen

Leach, N.; Yuan, P. (2018)
Computational Design
Tongji University Press

Picon, A. (2018) Jean-Nicolas-Louis Durand.
Standardisierung und Typus in der Architektur
ArchPlus [Online] https://www.researchgate.net/publication/330409289 [abgerufen am 05.05.2022]

Shaker, N.; Togelius, J.; Nelson, M. (2016)
Procedural Content Generation in Games
Cham, Schweiz: Springer

Wolff-Plottegg, M. (1996)
Architektur Algorithmen
http://plottegg.tuwien.ac.at/buch_toc.htm
[Abgerufen am 03.05.2022]
Passagen Verlag Ges.m.b.H