Technische Universität München

Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Computergestützte Modellierung und Simulation

Vergleich mehrerer Gestaltgrammatikimplementierungen bezüglich ihrer geometrischen und semantischen Ausdrucksweise

Bachelorthesis

für den Bachelor of Science Studiengang Bauingenieurwesen

Autor:	Christian Exner
Matrikelnummer:	03716479
1. Betreuer:	Prof. Dr.-Ing. André Borrmann
2. Betreuer:	Lothar Kolbeck

Ausgabedatum:	15. November 2021
Abgabedatum:	14. April 2022

Abstract

Gestaltgrammatiken (engl: shape grammar) sind ein generativer Formalismus, der in frühen Phasen des Designprozesses angewandt wird. Die Idee Gestaltgrammatiken rechnergestützt umzusetzen, wird nun schon seit einigen Jahrzehnten verfolgt. Trotzdem gibt es bis heute keine einheitliche Implementierung von Gestaltgrammatiken Interpretern. In den Anfangsphasen wurden Interpreter oft entwickelt, um eine bestimmte Art von Design darzustellen. Die drei ausgewählten Gestaltgrammatiken Interpreter vereinen sich in dem Ansatz allgemein anwendbar zu sein, jedoch unterscheiden sie sich untereinander in der Ausführung. Um eine bessere Erkenntnis zu erlangen, für welche Art von Problem diese Interpreter geeignet sind, werden sie anhand von einem vordefinierten Grundrissproblems getestet und anschließend bewertet.

Inhaltsverzeichnis

Abbildungsverzeichnis V

Tabellenverzeichnis VII

Abkürzungsverzeichnis VIII

1 Einführung und Motivation 1

1.1 Motivation 1

1.2 Problemstellung 2

1.3 Ziel der Arbeit 2

1.4 Aufbau der Arbeit 3

2 State of the Art 4

2.1 Shape grammar .... **Fehler! Textmarke nicht definiert.**

2.2 Graph grammar 6

2.2.1 Punkt zu Knoten und Linien zu Kanten 6

2.2.2 Linien zu Knoten und Schnittpunkte zu Kanten 7

2.2.3 Punkte und Linien zu Knoten 7

2.3 Split grammar 7

2.4 Sortal grammar 9

2.5 Forschungslücke 9

3 Methodik und Vorgehen 10

3.1 Design des Grundrissproblems 10

3.2 Entscheidungshilfe für Anwender 10

4 Shape Grammar Implementationen 12

4.1 Spapper 12

4.1.1 Entstehung 12

4.1.2 Interpreter 13

4.1.3 Regelerstellung und Anwendung 14

4.1.4 Regeln 19

4.2 GRAPE 23

4.2.1 Entstehung 23
4.2.2 Interpreter 23
4.2.3 Regelerstellung und Anwendung 26
4.2.4 Regeln 26
4.3 sortal 29
4.3.1 Entstehung 29
4.3.2 Interpreter 29
4.3.3 Regelerstellung und Anwendung 30

5 Analyse 32
5.1 Spapper 32
5.1.1 Nutzerfreundlichkeit 32
5.1.2 Regelerstellung 32
5.1.3 Regelanwendung 32
5.1.4 Anwendungsfall 33
5.2 GRAPE 33
5.2.1 Nutzerfreundlichkeit 33
5.2.2 Regelerstellung 33
5.2.3 Regelanwendung 34
5.2.4 Anwendungsfall 34
5.3 Sortal 34
5.3.1 Nutzerfreundlichkeit 34
5.3.2 Regelerstellung 35
5.3.3 Regelanwendung 35
5.3.4 Anwendungsfall 35
5.4 Resume 36

6 Zusammenfassung und Fazit 38

Literaturverzeichnis 39

Anhang A 41

Abbildungsverzeichnis

Abbildung 1: Grundform 4
Abbildung 2: Regel 4
Abbildung 3: Grundform 5
Abbildung 4: erste Regelanwendung 5
Abbildung 5: zweite Regelanwendung 5
Abbildung 6: Beispiel split grammar Unterteilung (Quelle: Wonka 2003) 8
Abbildung 7: Grundrissproblem 10
Abbildung 8: spapper Interface 13
Abbildung 9: spapper Toolbar 13
Abbildung 10: spapper FPC 14
Abbildung 11: spapper Free Parameter Configurator (FPC) 14
Abbildung 12: spapper Geometrische Grundkörper 15
Abbildung 13: spapper LHS 15
Abbildung 14: spapper RHS 15
Abbildung 15: spapper FPC von Box in RHS bei dem Parameter von entsperrter Box aus LHS angezeigt werden 16
Abbildung 16: spapper RHS H/L/W entsperrt und LHS Parameter in Equation-Feld eingefügt 16
Abbildung 17: spapper RHS FPC mit angepassten Platzierungsparameter 18
Abbildung 18: spapper Regelliste 18
Abbildung 19: spapper apply rule Fenster, um Objekt auszuwählen 18
Abbildung 20: spapper Bodenregel x-Richtung LHS 19
Abbildung 21: spapper Bodenregel x-Richtung RHS Box1 20
Abbildung 22: spapper Bodenregel x-Richtung RHS Box001 20
Abbildung 23: spapper Wandregel in x-Ebene über y-Koordinate verschoben, RHS 21
Abbildung 24: spapper Boden verdoppeln RHS 21
Abbildung 25: spapper Fenster Ausschnitt x-Ebene RHS 22

Abbildung 26: spapper Dach RHS 22
Abbildung 27: GRAPE Interface 23
Abbildung 28: GRAPE gestellte Grammatiken 24
Abbildung 29: GRAPE Erstellungsmenü für neue Regeln 24
Abbildung 30: GRAPE Schema 24
Abbildung 31: GRAPE properties area 24
Abbildung 32: GRAPE Mapping 25
Abbildung 33: GRAPE Layer 25
Abbildung 34: GRAPE LHS und RHS 25
Abbildung 35: GRAPE Overlay für Regelanwendung 26
Abbildung 36: GRAPE Grundform mit Label 27
Abbildung 37: GRAPE Raumunterteilung 27
Abbildung 38: GRAPE Fenster 28
Abbildung 39: GRAPE Tür 28
Abbildung 40: sortal CAD-Zeichenfläche und Grasshopper für visuelle Programmierung 29
Abbildung 41: sortal zwei Cruves jeweils für LHS und RHS 30
Abbildung 42: sortal Geometrie Elemente in Grasshhopper 30
Abbildung 43: sortal SGI Toolbar 30
Abbildung 44: sortal Curve zu einer Shape zuweisen 30
Abbildung 45: sortal Beispielregel 31
Abbildung 46: sortal Regel anwenden 31

Tabellenverzeichnis

Tabelle 1: Analyse visualisiert 37

Abkürzungsverzeichnis

LHS	Left-hand-side
RHS	Right-hand-side
FPC	Free Parameter Configurator

1 Einführung und Motivation

1.1 Motivation

Gestaltgrammatiken (engl. Shape grammars) sind eine von mehreren Möglichkeiten der inversen Analyse und anschließenden prozeduralen Generation von Entwurfs Geometrien. Der Mensch als Entwickler ist dazu geneigt sich auf seine Erfahrungen und Vertrautheiten zu stützen. Diese wendet er dann in dem was er tut an. Als prozeduraler Mechanismus ermöglichen Gestaltgrammatiken die Erkundung eines weiteren Lösungsraums, indem die angewandten Regeln oder ihre Parametrik variiert werden können. Anfangs wurden Gestaltgrammatiken noch mit Farbe und Bleistift auf Papier gezeichnet. Inzwischen gibt es viele Gestaltgrammatik Implementation, die es ermöglichen computergestützt zu arbeiten. Was die gezeichneten und computergestützten Gestaltgrammatiken gemeinsam haben, ist der Gedanke, dass Systeme geometrische Regelmäßigkeiten und Muster haben, welche abstrahiert werden können. Wenn man das akzeptiert, kann man die innere Logik der Geometrie regelbasiert formulieren. Gestaltgrammatiken finden ihre Anwendung im Architektur-Design, Ingenieurwesen und Produkt-Design. Zwei bekannte Beispiele bei deren Design Gestaltgrammatiken genutzt wurden, sind die Coca-Cola Flasche und das Motorrad Harley Davidson (Chau et al. 2004). Eine weitere Anwendung von Gestaltgrammatiken spezifisch für Architektur und Ingenieurwesen ist die Möglichkeit personalisierte Anpassung von Massenwohnungen zu vollziehen. Die Erkundung eines durch Regeln definierten Lösungsraums kann an Kostenfunktionen gekoppelt sein, die den Ressourcenverbrauch oder anderen quantifizierbaren Aspekten des Entwurfs berücksichtigen. Nachdem computergestützte Implementierungen von Gestaltgrammatiken ein recht junges und unkonsolidiertes Forschungsfeld sind, fehlt es einer einheitlichen Betrachtung verschiedener Implementierungslösungen. In den folgenden Abschnitten wird genauer auf das Thema eingegangen, mit welchem sich diese Bachelorarbeit auseinandersetzt.

1.2 Problemstellung

An Gestaltgrammatiken wird schon seit über 40 Jahren gearbeitet, jedoch werden sie bis heute noch kaum in Entwicklungs- und Designprozessen angewandt. Das liegt unter anderem daran, dass es kaum kommerzielle oder frei zugängliche Softwareimplementierungen gibt. Zusätzlich unterscheiden sich viele Interpreter in ihren Datenstrukturen und Algorithmen für Gestaltgrammatiken, sowie der Anwendung. So besteht bei Gestaltgrammatik Implementierungen für den Nutzer das Problem, dass er bereits vor der Auswahl des Interpreters wissen muss, wie dieser funktioniert. In dieser Arbeit sollen anhand eines vordefinierten Grundrissproblems drei Interpreter vorgestellt werden. Die Implementierungen verfolgen zwar alle die gleiche Grundidee, jedoch unterscheiden sie sich in der Umsetzung.

1.3 Ziel der Arbeit

In der Vergangenheit wurden Gestaltgrammatik Implementationen immer für eine bestimmte Problemstellung erstellt. Der Vorteil dabei war, dass die Grammatik genau auf die Problemstellung angepasst werden konnte. Die drei Interpreter, die in dieser Bachelorarbeit vorgestellt werden, verfolgen den gemeinsamen Anspruch allgemein anwendbar zu sein. Die Herausforderung, die bei diesen Interpretern aufkommt, ist dem Anspruch von Gestaltgrammatiken gerecht zu werden. So sollte eine Gestaltgrammatik Implementation idealerweise grafische Nutzerinteraktion unterstützen, Entstehung (engl. Emergence) erlauben, nicht von vordefinierten Teilen abhängig und parametrisch sein (Gips 1999) (Übersetzung des Autors). Der Punkt Entstehung ist im Design vor allem in der Anfangsphase sehr relevant, allerdings schwer mit der logischen Berechnung von CAD-Programmen zu vereinen (Iestyn Jowers 2019). Zusätzlich sorgt die Unterstützung von Entstehung dazu, dass die Implementierung sich mit dem Problem der subshape detection auseinandersetzen muss. Bei der subshape detection sollen Formen, die durch Regelanwendung entstanden sind, erkannt werden.

Die drei ausgewählten Interpreter haben sich mit diesen Problemen aus der Forschung auseinandergesetzt und jeweils unterschiedliche Ansätze und Lösungen gefunden, um damit umzugehen. Der größte Unterschied, der die Interpreter unterscheidet, ist die Wahl des Raums, in dem sie arbeiten. Zwei Interpreter wählen den Ansatz die Hebung von Punkten und Kurven zu Flächen bzw. die Projektion von Flächen in Punkten und Kurven U_{13} darzustellen, während der dritte Interpreter Volumenkörper im 3D-Raum unterstützt, U_{33}.

Diese Bachelorarbeit hat das Ziel Ingenieuren bei der Anforderungsanalyse an eine passende Implementierung zu unterstützen und eine Hilfestellung bei der Auswahl von Gestaltgrammatik Interpretern zu bieten. Dafür wird jeder Interpreter zuerst anhand eines Beispiels vorgestellt. Um die Interpreter untereinander zu vergleichen und zu unterscheiden, werden diese an ausgewählten Kriterien getestet. Dabei steht im Fokus, ob die einzelnen Interpreter die Anforderungen, welche Gestaltgrammatik Implementationen aufgesetzt werden, umgesetzt haben.

1.4 Aufbau der Arbeit

Kapitel 2 beginnt mit einer Definition von Gestaltgrammatiken und liefert weitere Information zum heutigen Stand der Technik. In den folgenden Unterpunkten werden drei Typen an formalen Grammatiken vorgestellt, welche für die in dieser Arbeit vorgestellten Interpreter relevant sind. Zusätzlich wird beschrieben, wie sich die einzelnen Grammatiken in ihrer Vorgehensweise untereinander unterscheiden.

Darauffolgend wird in Kapitel 3 auf Forschungsfragen eingegangen. Zusätzlich wird das Experimenten Design vorgestellt, sowie die Kriterien anhand welchen die Entscheidungshilfe für die Ingenieure festgelegt werden.

In Kapiteln 4 wird auf die Interpreter spapper, GRAPE und sortal eingegangen. Um einen möglichst guten Vergleich zwischen den drei Interpretern liefern zu können ist auch der Aufbau der drei Unterkapitel identisch gehalten. Dabei wird als erstes darauf eingegangen in welchem zeitlichen und anwendungsspezifischen Kontext der jeweilige Interpreter entstanden ist. Anschließend wird erklärt, wie der Interpreter installiert wird, welche zusätzlichen Programme benötigt werden und wie das Interface aufgebaut ist. Als nächstes wird die Regelerstellung und Anwendung anhand von Beispielen beschrieben. Dabei wird bereits darauf eingegangen, was der jeweilige Interpreter für Bearbeitungsmöglichkeiten liefert. Das bezieht sich vor allem auf die in Kapitel 3 beschriebenen Kriterien wie z.B. die Möglichkeit Regeln parametrisch zu erstellen. Danach werden die Regeln vorgestellt, mit denen das Grundrissproblem gelöst wurde.

In Kapitel 5 folgt dann die Analyse und der Vergleich der drei Interpreter untereinander.

Abschließend wird in Kapitel 6 eine Zusammenfassung und ein Fazit zu Gestaltgrammatiken und den Interpretern gegeben.

2 State of the Art

2.1 Gestaltgrammatiken

Eine Gestaltgrammatik ist ein prozeduraler Mechanismus, der anhand von Regelsätzen die Entwicklung von Geometrien beschreibt. Gestaltgrammatiken können Formen die durch Formberechnung (computation) entstehen erkennen und manipulieren. Dabei wird eine Grundform (left hand side (LHS)) durch eine Endform (right hand side (RHS)) ausgetauscht. Stiny hat 1980 beschrieben, dass Gestaltgrammatik aus vier Komponenten bestehen:

1. eine endliche Menge von Formen S
2. eine endliche Menge von Symbolen L
3. eine endliche Menge von Regeln R der Form r: a→b
4. eine Form I die eine beschriftete Form ist, und auch als Initialform bezeichnet wird. (Stiny 1980 S.347) (Übersetzung des Verfassers)

Um Veränderungen an Formen durchzuführen, werden Regeln erstellt. Diese Regeln erkennen dann, ob eine für die Regel passende Grundform gegeben ist. Der Standardablauf ist nun, dass die LHS mit der RHS ausgetauscht wird (a→b). Ein einfaches Beispiel ist, das eine Grundform a ein Quadrat (Abbildung 1) um ein weiteres Quadrat, welches quer verschoben ist, erweitert wird (Abbildung 2).

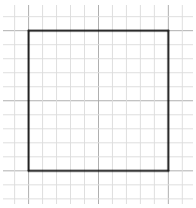

Abbildung 1: Grundform

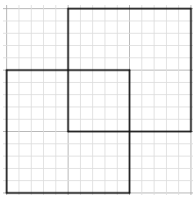

Abbildung 2: Regel

Eine Anwendung der Regel würde wie folgt aussehen (Abbildung 3-5)

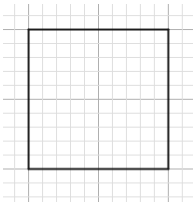

Abbildung 3: Grundform

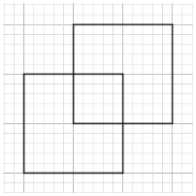

Abbildung 4: erste Regelanwendung

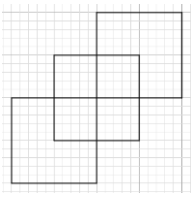

Abbildung 5: zweite Regelanwendung

Erweiterungen sind Regeln, die parametrisch oder kontext-sensitiv definiert werden. Kontext-sensitive Regeln werden mit $xAy \rightarrow xby$ beschrieben, wobei $A \in N$, $x,y,b \in T^*$ und $b \notin \varepsilon$, mit:

- nichtterminal Vokabular N
- T* ist das kleinste Set der terminalen Grammatiken die unter string Verkettungen geschlossen sind
- ε string der Länge 0 (Krishnamurti, Stouffs 1993) (Übersetzung des Verfassers)

Die Manipulation bringt Veränderung und Veränderung unterstützt Entstehung, weshalb Gestaltgrammatiken als Hilfestellung im kreativen rechnergestützten Entwurf Design verwendet werden können.

Anfangs wurden Gestaltgrammatiken hauptsächlich für Punkte und Linien verwendet. In den letzten 10 Jahren der Forschung hat man angefangen auch 3d-basierte Volumenkörper (engl. solids) zu prozessieren. Um allgemein zwischen 2d und 3d basierten Gestaltgrammatiken unterscheiden zu können gibt es die U_{ij}-Notationen. Dabei werden Elemente einer Dimension i im Raum einer Dimension j beschrieben (McKay 2012, S.4). Somit beschreibt U_{12} eine Hebung von Linien- auf Flächendarstellungen, das bedeutet, dass in der LHS einer Regel wireframeartige Muster referenziert werden, welche vom Interpreter in Flächengeometrien in der RHS gehoben werden. U_{33} steht demnach für Volumenkörper im 3D-Raum.

In wissenschaftlichen Schreiben werden Gestaltgrammatiken oft als spatial grammar bezeichnet. Das liegt vor allem an der Definition von Gestaltgrammatiken, die beschreibt, dass Gestaltgrammatiken eine Darstellung von maximalen Elementen sind, welche Entstehung (engl. Emergence) unterstützen. Das bedeutet, dass Formen, die bei der Formberechnung aufkommen erkannt werden können und diese dann auch manipulierbar sind, ohne dass die Formen explizit dargestellt werden. Die für den industriellen Produkt Design typischen Formen der Volumenkörper und Flächen können

theoretisch maximal dargestellt werden. Allerdings hat man bis heute Probleme bei der Implementierung die Voraussetzung zu erfüllen, dass gleiche Grenzen dieser Formelemente erkannt werden (McKay, Chase et al. 2012).

In anderen wissenschaftlichen Schreiben wird zwischen Gestalt- und spatial grammar unterschieden, da Gestaltgrammatiken anders als spatial grammar direkt an räumlichen Formen agieren (Krishnamurti & Stouffs 1993). Allgemein kann man sagen, dass sich spatial grammar mit einem breiteren Spektrum befassen als Gestaltgrammatiken. Spatial grammar beschränken sich nicht nur auf Formen, sondern befassen sich zusätzlich noch mit der Semantik. Zur Vereinfachung wird in dieser BA hauptsächlich die Bezeichnung Gestaltgrammatik benutzt.

2.2 Graph grammar

Eine Möglichkeit Gestaltgrammatiken zu implementieren ist die Nutzung von Graphen als Datenstruktur und von Graphersetzung zur Entwurfsmanipulation. Der Vorteil bei der Umwandlung in Graphen ist, dass Graphen sich besser für Computer Implementationen eignen und bereits existierende Graphenbibilotheken mit Implementierung eines Ersetzungsregelmechanismus dazu genutzt werden können genau angepasste Software zu erstellen. Vor allem bei Designs, die vor allem durch Konnektivität zwischen den einzelnen Elementen dargestellt werden, eignen sie graph grammar besonders gut (Krishnamurti & Stouffs 1993). Ein weiterer Vorteil der Graphen ist, dass topologische Muster dargestellt und gespeichert werden können. Geometrie lastige Darstellungen müssen auf Ähnlichkeitstransformationen zurückgreifen (Krishnamurti Ramesh 1992). Außerdem bieten graph grammar eine einfache Lösung für das subshape detection Problem. Graph grammar unterscheiden sich untereinander in ihrer Vorgehensweise. In den folgenden drei Kapiteln werden drei Vorgehensweisen beschrieben, die von Thomas Grasl 2011 vorgestellt wurden und sich auf U_{12} Implementierungen konzentrieren.

2.2.1 Punkt zu Knoten und Linien zu Kanten

Der einfachste Weg ist Punkte zu Knoten und Linien zu Kanten zuzuordnen. Das Hauptproblem bei dieser Vorgehensweise ist, dass keine maximalen Linien dargestellt werden können. Das sorgt dafür, das entstehende Formen schwer zu erkennen sind. Ein Lösungsansatz für dieses Problem haben Kelles et al. (2010) vorgestellt, indem sie alle Knoten einer Linie verkoppelt haben. Das Problem an dieser Vorgehensweise

ist, dass sie sehr Ressourcenintensiv ist und daher die Berechnungszeit steigt. Eine bessere Alternative ist die Benutzung von Hyperkanten. In einer Hyperkante können mehrere Knoten gespeichert werden, was dafür sorgt, dass die Hyperkante als maximal dargestellt werden kann. Allerdings gibt es nur wenige Graphersetzungsalgorithmen die auf Hypergraphen verallgemeinert wurden.

2.2.2 Linien zu Knoten und Schnittpunkte zu Kanten

Bei dieser Vorgehensweise wird jedes Liniensegment als Knoten dargestellt und Kanten werden genutzt, um Kreuzpunkte darzustellen. Das führt allerdings wieder zu dem Problem, dass keine maximalen Linien dargestellt werden können. Deshalb koppelt man maximale Linien zu Knoten, was dafür sorgt, dass es weniger Kanten gibt und entstandene Formen erkannt werden können. Wenn sich allerdings mehr als zwei Linien in einem Punkt überschneiden, führt das bei dieser Vorgehensweise zu Problemen, welche jedoch wieder durch die Einführung von Hyperkanten gelöst werden können.

2.2.3 Punkte und Linien zu Knoten

Bei diesem Ansatz können Punkte und Liniensegmente als Knoten dargestellt werden. Da Liniensegmente anstatt Linien dargestellt werden sind maximale Linien nicht darstellbar. Es besteht hier allerdings die Möglichkeit, ohne die Anwendung von Hyperkanten dieses Problem zu lösen, indem man anstatt der Liniensegmente maximale Linien benutzt. Man geht nun davon aus, dass Punkte und maximale Linien durch Knoten dargestellt werden und Kanten für verschiedene Beziehungen zuständig sind. Dadurch, dass mehrere Klassen zu Knoten verknüpft werden, werden mehr Knoten gebraucht, um eine Form darzustellen. Allerdings können dadurch auch andere Unterscheidungsmerkmale den Klassen zugeordnet werden, wie z.B. Beschriftungen, Flächen, Volumenkörper, Farben.

2.3 Split grammar

Split grammar sind eine Art von Gestaltgrammatiken, die im 3-dimensionalen arbeitet. Dabei wird das Design z.B. eines Gebäudes als einheitliches Modell betrachtet. So kann nun z.B. eine Wand in vier Abschnitte unterteilt werden, von denen dann zwei

Abschnitte zu Fenstern, ein Abschnitt zur Tür und der letzte Abschnitt einer Fassade zugewiesen werden.

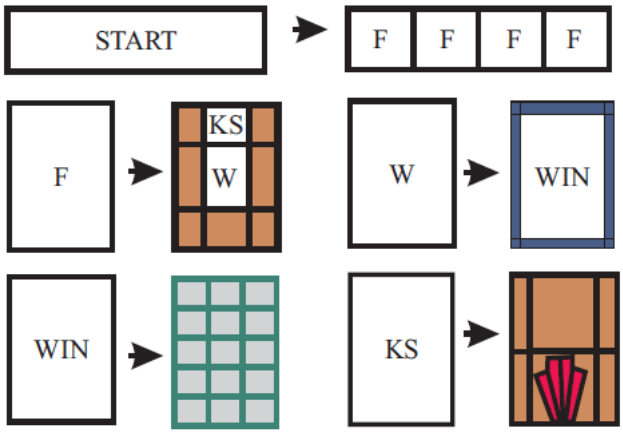

Abbildung 6: Beispiel split grammar Unterteilung (Quelle: Wonka 2003)

Diese Unterteilungen können beliebig detailliert erstellt werden, wodurch auch komplexe Gebilde mit einem einfachen Algorithmus dargestellt werden können. Split grammar sind gut für automatisierte Modellierung geeignet. Um eine gute Balance zwischen Ausdruckfähigkeit der Grammatik und Automatisierung zu erreichen, werden Beschränkungen aufgestellt. Dazu gehört z.B. die Anzahl an unterschiedlichen Designs, die erlaubt sind. Split grammar verarbeiten parametrisierte und markierte basic shapes (unter basic shapes versteht man z.B. Quader, oder Zylinder) (Wonka 2003). Bei der Anwendung von Regeln wird zwischen einer split rule und einer conversion rule unterschieden. Bei der split rule a→b ist ein basic shape a in der Untermenge von B wobei b bis auf ein Element c die gleichen Elemente wie a besitzt. Das Element c wird dann aufgeteilt (gesplitted). Die conversion rule a→b transformiert basic shapes. Dabei ist a Untermenge von B, in der eine basic shape ist. In b sind die gleichen Elemente wie a allerdings ist die basic shape eine andere. Die basic shape von b muss allerdings im Volumen der basic shape von a enthalten sein (Wonka 2003).

2.4 Sortal grammar

Sortal grammar basieren auf der Datenstruktur von sorts. Sorts können wie eine hierarchische Struktur von Eigenschaften angesehen werden, in der jede Eigenschaft einen bestimmten Datentyp besitzt (Rudi Stouffs 2008). Zusätzlich eignen sich sorts dazu geometrische und semantische Informationen flexibel zu konfigurieren. Dadurch können Labels, Farben und Geometrieparameter abgespeichert und gematched werden (Rudi Stouffs, 2008). Bei den Farben wird auch auf die Durchsichtigkeit, Deckfähigkeit und Rangliste geachtet. Sortal grammar beinhaltet:

- Punkt- und Liniensegmente bei denen die Strichfärbung sowie Liniendicke erkannt wird
- Flächensegmente mit zugehöriger Füllfärbung
- Markierte Punkte, bei denen die Markierung eine dazugehörige Färbung erhält
- Markierte Linien- und Flächensegmente

Die Füllfärbungen können als numerisches Gewicht oder als aufzählender Wert in einer Rangliste spezifiziert werden.

2.5 Forschungslücke

Wenn man anfängt mit Gestaltgrammatiken zu arbeiten liegt das erste Problem bereits darin, dass man nicht nur lernen muss, wie man mit Gestaltgrammatiken umgeht, sondern auch wie das Programm, welches man bedient funktioniert. Dieses Problem wird zusätzlich erschwert, da verschiedene Gestaltgrammatik Interpreter auf unterschiedlichen Programmen laufen. Bis heute gibt es noch keine einheitliche Implementierung. Ein weiteres Problem ist, dass sich der Nutzer vor der Wahl des Gestaltgrammatik Interpreter damit auseinandersetzen muss, was er für sein spezifisches Problem braucht. Es gibt inzwischen einige Gestaltgrammatik Interpreter, jedoch unterscheiden sich diese alle in ihrer Funktionsweise.

3 Methodik und Vorgehen

3.1 Design des Grundrissproblems

Damit eine Umfangreiche und aussagekräftige Evaluierung der Gestaltgrammatik Interpreter erfolgen kann, werden die Interpreter an einem einheitlichen Problem getestet. Die Aufgabe ist hierbei mit Hilfe von erstellten Regeln den Grundriss eines Gebäudes zu bilden (Abbildung 7). Je nach Interpreter wird das Grundrissproblem in 2D oder 3D gelöst. Zusätzlich zu dem Grundriss wird getestet, wie sich Raumunterteilungen einbringen lassen, sowie das Einfügen von Fenstern und Türen. Im 2D Interpreter werden Fenster und Türen mit Markierungen sichtbar gemacht.

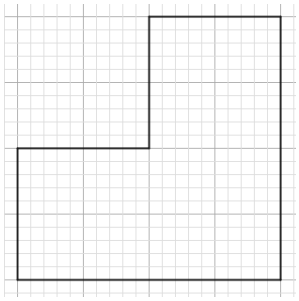

Abbildung 7: Grundrissproblem

3.2 Entscheidungshilfe für Anwender

Anhand des vorhergegangenen Experimenten Designs werden die einzelnen Interpreter bewertet. Der erste Bewertungspunkt ist die Nutzerfreundlichkeit. Dabei umfasst die Nutzerfreundlichkeit viele einzelne Punkte, die alle in die Erfahrung des Anwenders mit einspielen. Dazu gehören unter anderem die Komplexität der Installierung des Interpreters, Übersichtlichkeit des Interface, Laufstabilität des Programms, Einarbeitung mit Hilfe gestellter Tutorials, Regelerstellung und Regelanwendung. Weitere Bewertungsinhalte sind die Erstellung und Anwendung von Regeln. Dabei wird getestet, ob es möglich ist Regeln parametrisch zu erstellen. Des Weiteren wird die Freiheit bewertet, die einem der Interpreter bei der Regelerstellung gewährleistet. Zusätzlich wird darauf geachtet, ob das Programm in der Lage ist durch vorhergegangen Regelanwendungen neu entstandene Flächen zu erkennen (subshape detection). Des Weiteren wird überprüft, ob der Interpreter Label unterstützt.

Zum Schluss wird getestet, welche Möglichkeiten der Interpreter bei der Regelanwendung bietet. Merkmale, auf die dabei geachtet werden sind z.B., ob der Interpreter definierte Regeln automatisch durchführen kann, aber auch ob der Interpreter mehrere Lösungsvorschläge anbietet. Dazu gehört auch die Frage, ob der Interpreter Regelmäßigkeiten in den Formen und Geometrien erkennt und somit die Anzahl der benötigten Regeln verringert wird.

Die Bewertungspunkte sind so gewählt, dass Ingenieure ihr Problem mit der Analyse der drei Interpreter abgleichen können und dadurch eine Entscheidungshilfe auf die Frage finden, welcher Gestaltgrammatik Interpreter für ihr persönliches Problem am besten geeignet ist. Zusätzlich zur Analyse werden die Interpreter für verschiedene Anwendungsfälle vorgeschlagen. Dabei wird auch die unterschiedliche Komplexität der Interpreter und die damit einhergehende Lernkurve mit in Betracht gezogen.

4 Gestaltgrammatik Implementationen

4.1 Spapper

4.1.1 Entstehung

Vor der Entwicklung von spapper gab es schon eine Handvoll an Gestaltgrammatik Interpretern, allerdings waren nur wenige davon 3D. Zusätzlich wurden die 3D Gestaltgrammatiken, die es bis zu diesem Zeitpunkt gab hauptsächlich für eine bestimmte Problemstellung entwickelt. Das Ziel von spapper war es, einen Interpreter zu erstellen bei dem die Nutzer ihre eigenen Regeln erstellen können. Die Anforderungen an den Idealen 3D Gestaltgrammatiken können wie folgt zusammengefasst werden (Frank Hoisl, Kristina Shea, 2010) (Übersetzung des Verfassers):

- Allgemein, das heißt keine Beschränkungen auf ein bestimmtes Problem
- Uneingeschränkte Vokabeln die eine große Vielfalt an 3D Objekten zulässt
- Definition von parametrischen Regeln
- Uneingeschränkte Menge an Regeln
- Uneingeschränkte Menge an Objekten, die in einer Regel genutzt werden können
- Regeln sollen nicht nur erstellt, sondern auch editiert werden können
- Grafische Darstellung und direkte Manipulation von Objekten in Regeln
- Umfassende Bedienung von Transformation Tätigkeiten in 3D
- Automatische Anpassung der LHS der Regel nach Transformation und unter Berücksichtigung von parametrischen Abhängigkeiten
- Interaktive Anwendung von Regeln (automatisch, semi-automatisch, manuell)
- Ein intuitives Nutzer Interface, was wenig, bis keine Programmierung braucht

Spapper ist als Prototyp Software aus diesem Ansatz heraus entstanden und wurde 2010-03-07 als Download auf sourceforge freigegeben.

4.1.2 Interpreter

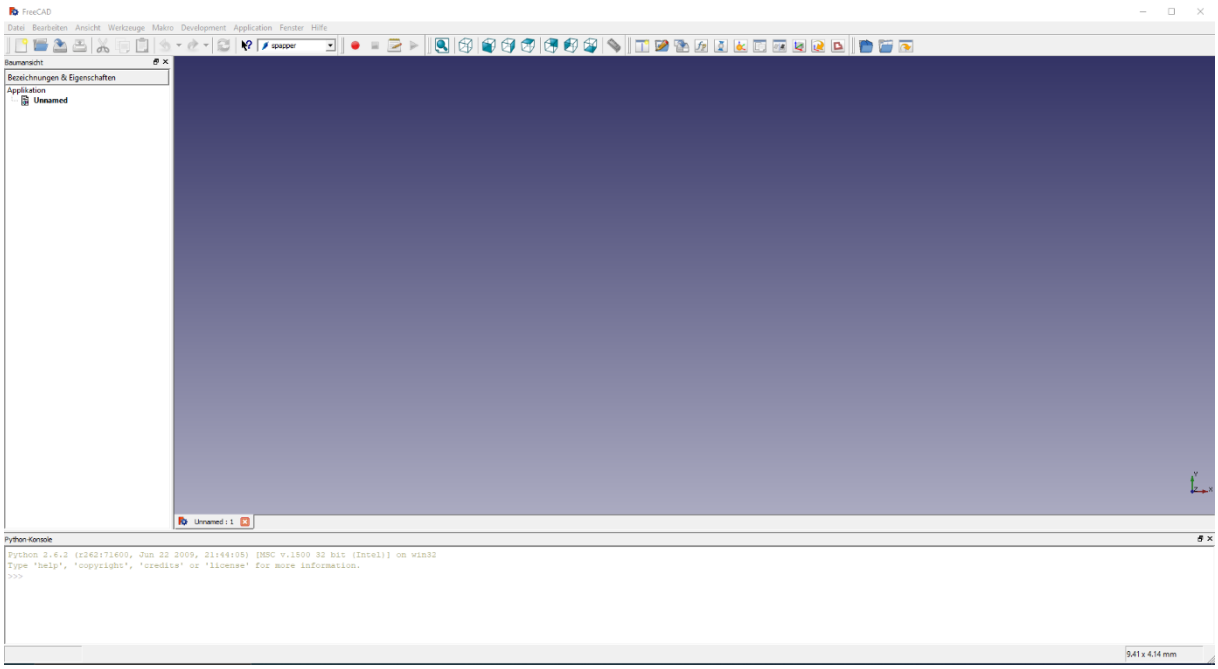

Abbildung 8: spapper Interface

Spapper ist ein interaktiver visueller 3D spatial grammar Interpreter basierend auf dem CAD Programm FreeCAD. Die Version auf der spapper läuft ist FreeCAD 0.11. Spapper steht zum Download auf sourceforge.net bereit, FreeCAD 0.11 gibt es dort nicht zu downloaden, da dieses CAD Programm bis heute regelmäßig geupdated wird, weshalb man FreeCAD 0.11 anderweitig downloaden muss. Nachdem man die beiden Programme heruntergeladen hat, findet man im Downloadordner von spapper eine Anleitung in der beschrieben wird, wie man spapper zu FreeCAD 0.11 hinzufügt. Nachdem man die Anleitung durchgeführt hat, kann man FreeCAD öffnen und unter Ansicht-Arbeitsbereich spapper auswählen. Dadurch öffnet sich die Toolbar mit den 14 Anwendungen die spapper bereitstellt (Abbildung 9). Diese Tools unterscheiden sich in elf Entwicklungstools und drei Anwendungstools.

Abbildung 9: spapper Toolbar

Die Entwicklungstools werden genutzt, um Regeln zu erstellen, speichern und editieren. Der „Free Parameter Configurator“ (FPC) (Abbildung 10) ist in diesem Zusammenhang das meistgenutzte Tool, da hier die geometrischen und räumlichen Parameter der Objekte angepasst werden können.

Abbildung 10: spapper FPC

Andere Entwicklungstools beinhalten das Einfügen von Label und lokalen Koordinatensystemen, sowie das Öffnen von der LHS und RHS zur Regelerstellung, editieren der Regeln und Speicherung.

Die Anwendungstools, werden genutzt, um gespeicherte Regeln zu öffnen und anzuwenden.

In 4.1.3 wird auf einzelne Tools genauer eingegangen.

4.1.3 Regelerstellung und Anwendung

Der erste Schritt, bevor man anfängt Regeln bei spapper zu erstellen ist, dass man sich überlegt, nach welchem Ansatz die Regel erstellt werden sollen. Eine Möglichkeit ist, mit einem Würfel als Grundkörper zu starten und Regel für Regel Bereiche aus dem Würfel zu entnehmen, bis die Form des Grundkörpers erreicht wird (split grammar Ansatz).

Anders kann man allerdings auch Regeln erstellen, die dafür sorgen, dass einzelne Körper wie Bauteile aneinandergesetzt werden und somit die Endform erstellt wird. Im Rahmen dieser Bachelorarbeit wurde sich für die zweite Herangehensweise entschieden. Das Grundrissproblem wird zur Vereinfachung hauptsächlich aus Platten und Scheiben mit den Maßen 10/10/1 aufgebaut.

Im Folgenden wird auf einzelne Tools eingegangen, die auf die Regelerstellung einen großen Einfluss nehmen. Eine volle Übersicht über die erstellten Regeln findet sich im nächsten Kapitel.

Bei der Regelerstellung in spapper ist anzumerken, dass sich ein Großteil davon im FPC (Abbildung 11) abspielt. Dementsprechend wird in diesem Kapitel vertieft auf den Umgang mit dem FPC eingegangen.

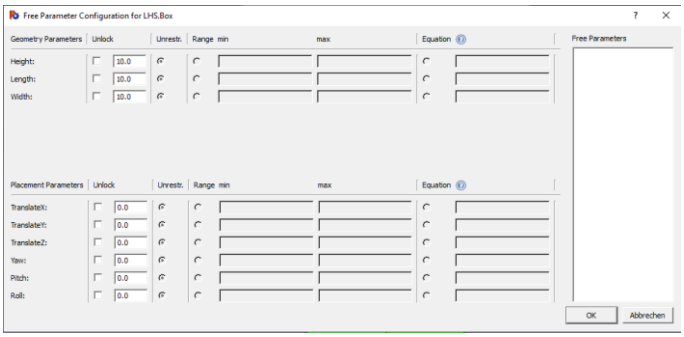

Abbildung 11: spapper Free Parameter Configurator (FPC)

Fängt man nun an seine Regeln zu erstellen öffnet man wie in 4.1.2 beschrieben als erstes eine LHS/RHS. Nun erstellen wird der erste Grundkörper erstellt. Dafür wird die FreeCAD Anwendung Parametrisch → Grundkörper erstellen verwendet. Dort hat man die Wahl zwischen einer Vielzahl an Grundkörpern.

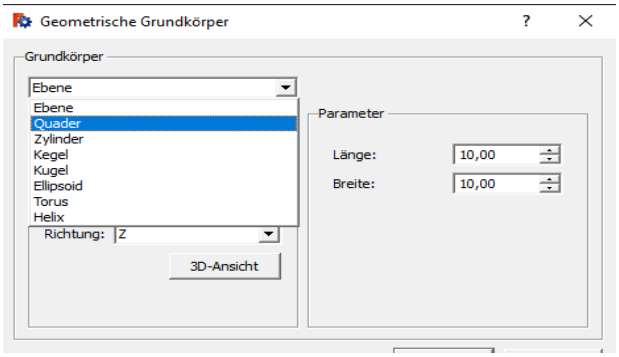

Abbildung 12: spapper Geometrische Grundkörper

In die LHS wird ein Quader mit den Maßen 10/10/1 eingefügt (Abbildung 13).

Dieser Volumenkörper wird nun mit dem entsprechenden Tool auf die RHS kopiert. Zusätzlich wird ein zweiter Quader mit den gleichen Maßen um 10 Einheiten in x-Richtung versetzt hinzugefügt (Abbildung 14).

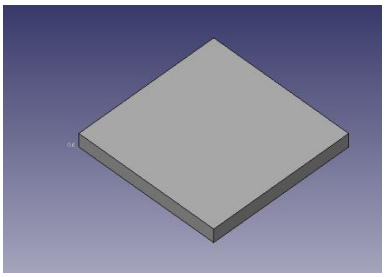

Abbildung 13: spapper LHS

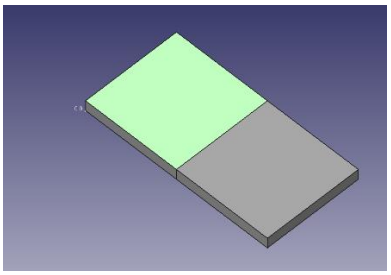

Abbildung 14: spapper RHS

Auf den ersten Blick kann man nun denken, dass unsere erste Regel fertiggestellt ist. Allerdings ist die Anfertigung der Grundkörper bei der Regelerstellung der kleinere Teil. Damit unsere Regel wie gewollt funktioniert, muss mit Hilfe des FPC jedes Objekt in der LHS und RHS bearbeitet werden.

In der LHS wird mit dem FPC genau festgelegt, welche Formen erkannt werden. Standardmäßig sind Height/Length/Width (H/L/W) gesperrt. Das bedeutet, dass die Regel mit der LHS nur Objekte erkennt, welche genau den angegeben Maßen entsprechen. Wenn man die Eingangswerte entsperrt, gibt es drei weitere Einstellungsmöglichkeiten. Unrestrained, Range min-max und Equation (siehe Abbildung 15).

Für unsere Regel wurden in der LHS H/L/W als Unrestrained eingestellt. Das hat zur Folge, dass die LHS alle Quader erkennt und dementsprechend als Initialregel verwendet werden kann. Eine weiter Sache, die das Entsperren von H/L/W mit sich bringt

ist, dass die entsperrten Parameter nun als freie Parameter im FPC von den anderen Objekten in der LHS sowie RHS angezeigt werden (Abbildung 15).

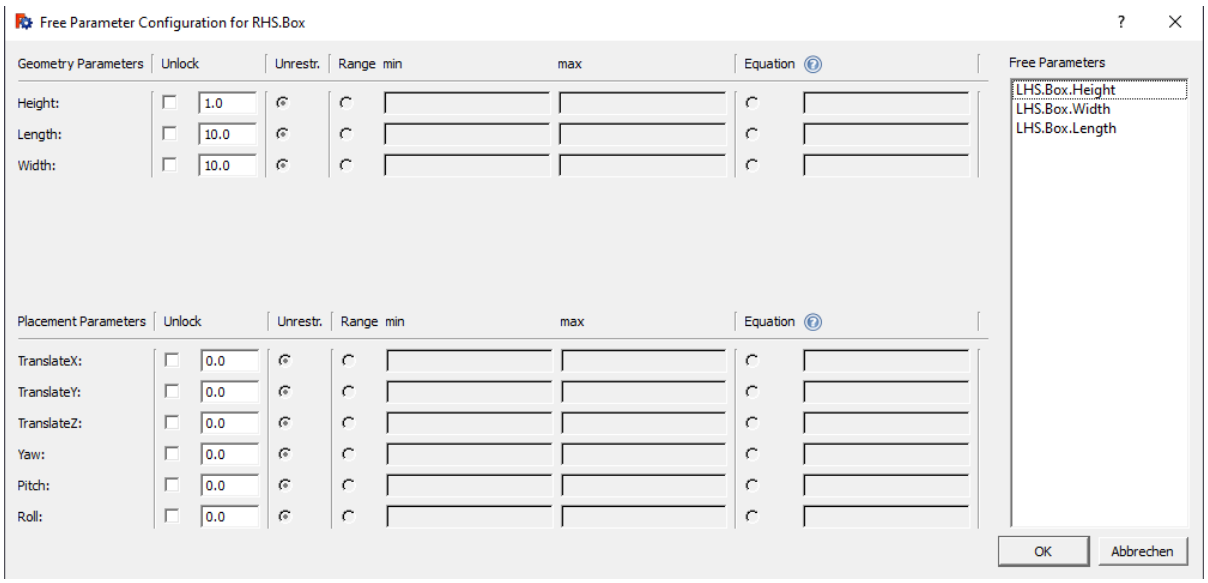

Abbildung 15: spapper FPC von Box in RHS bei dem Parameter von entsperrter Box aus LHS angezeigt werden

Damit ermöglicht uns spapper parametrische Abhängigkeiten zwischen LHS und RHS zu erstellen.

In der RHS wird nun das erste Objekt, welches Ursprünglich aus der LHS in die RHS kopiert wurde mit dem FPC bearbeitet. Dabei wird auch hier H/L/W entsperrt. Das gleiche wird auch mit der in x-Richtung anliegenden Platte gemacht.

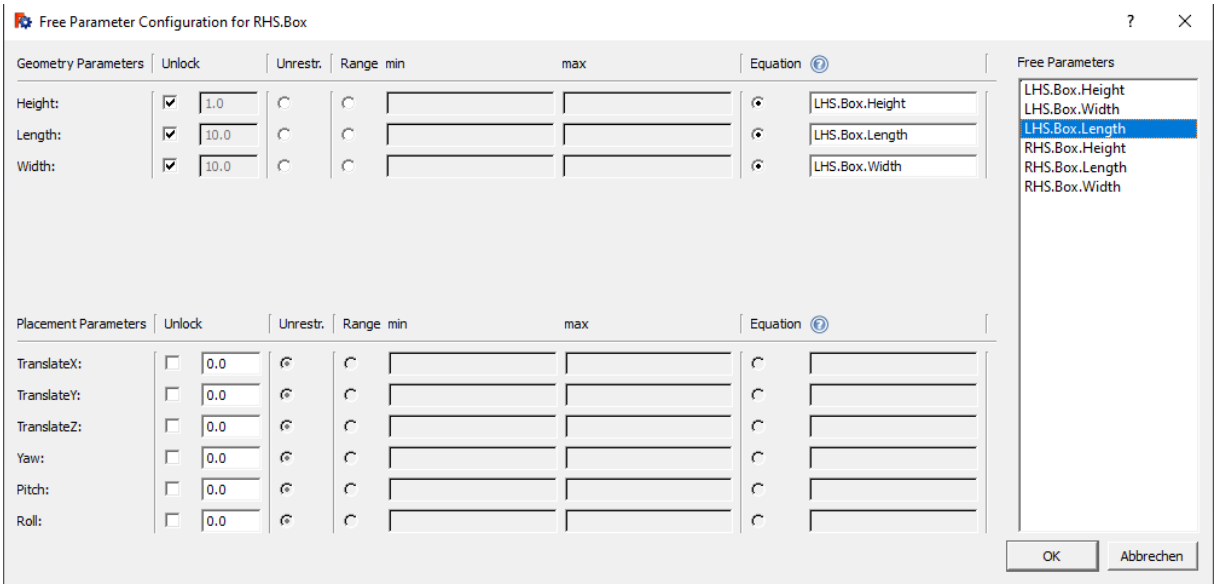

Abbildung 16: spapper RHS H/L/W entsperrt und LHS Parameter in Equation-Feld eingefügt

Anschließend werden bei beiden Bodenplatten der RHS die freien Parameter der LHS per drag and drop in das Equation Feld gezogen (Abbildung 16).

Um diesen Schritt nachzuvollziehen, muss man zuerst verstehen, dass spapper mit der Regelausführung nicht nur die Form der Objekte verändert, sondern auch direkt die Maße, die im FPC angegeben sind, einstellt. In unserem Beispiel hat die Platte in der RHS die Maße 10/10/1, mit welchen sie ursprünglich erstellt wurde. Wenn man diese Parameter nun im FPC nicht freigibt, werden alle Bodenplatten nach der Regelanwendung automatisch auf die Maße 10/10/1 festgelegt, unabhängig davon, wie die Maße der Ursprungsform festgelegt waren. Spapper passt die Maße der RHS nicht an die Maße der LHS an, sondern führt nur die Angaben aus dem FPC aus. Durch das Einfügen der LHS Parameter in die FPC der RHS-Körper wird nun bei der Regelanwendung die Bemaßung der Ursprungsform übernommen, anstatt diese zu überschreiben.

In das Equation Feld im FPC können nicht nur die freigestellten Parameter eingebracht werden, sondern auch mathematische Formeln mit Addition, Subtraktion, Division und Multiplikation. Zusätzlich kennt spapper auch Π, e und weiter Operatoren aus der Mathematikbibliothek. Alle diese Operatoren können mit freien Parametern verbunden werden.

Da spapper wie oben beschrieben nicht selbst mitdenkt, sondern nur die Einstellungen im FPC ausführt müssen auch die Platzierungsparameter angepasst werden.

Unsere Regel kann bis jetzt Platten in den Maßen 10/10/1 aneinanderreihen. Möchte man jedoch eine Platte mit einer anderen Länge hinzufügen wird die Regel wie sie bis jetzt besteht auf Fehler stoßen.

Das liegt daran, dass nicht nur geometrische Parameter mit dem FPC variabel angepasst werden müssen, sondern auch Platzierungsparameter zu bearbeiten sind. Möchte man nun in unserem Beispiel das Platten mit unterschiedlichen Längen aneinander angereiht werden können muss die x-Koordinate in der RHS von der hinzugefügten Bodenplatte variabel auf die Länge der kopierten RHS-Platte eingestellt werden (Abbildung 17).

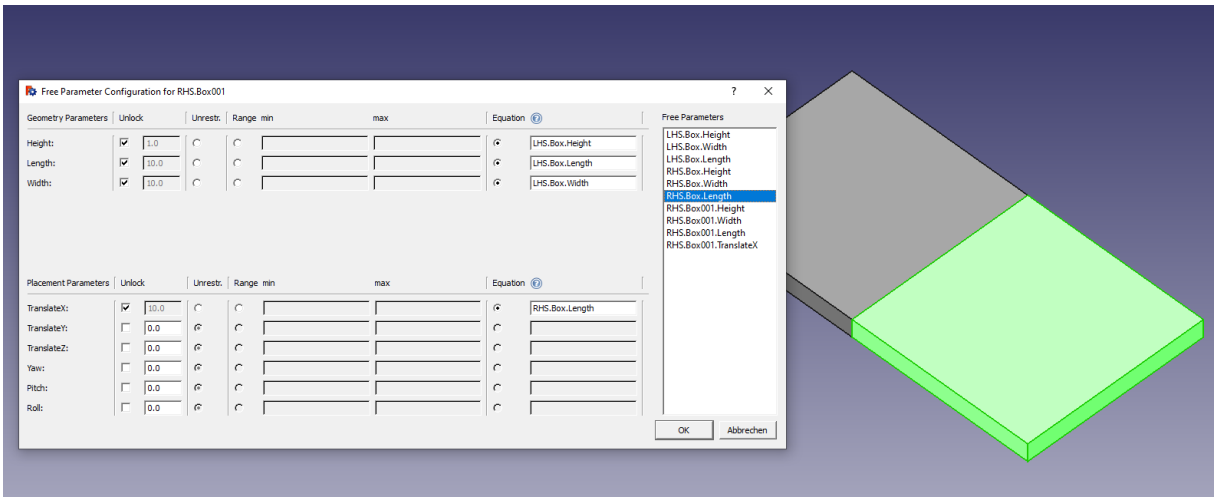

Abbildung 17: spapper RHS FPC mit angepassten Platzierungsparameter

Was nach der Regelerstellung noch fehlt ist die Anwendung der Regeln. Öffnet man die gespeicherten Regeln, werden diese in einer Liste dargestellt (Abbildung 18). In der Liste sind zusätzlich noch Einstellungen enthalten, welche die Regelanwendung individuell anpassbar machen.

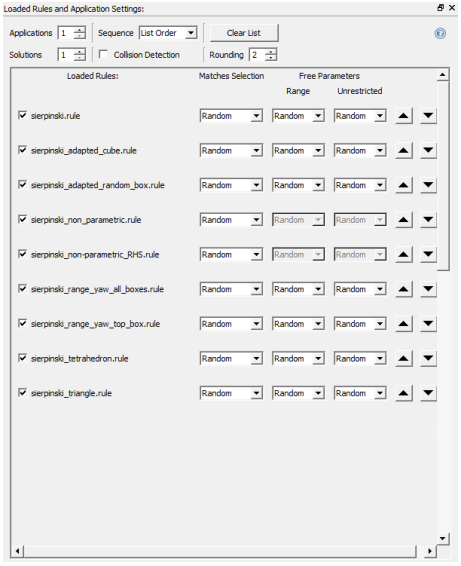

Abbildung 18: spapper Regelliste

Da für unser Grundrissproblem eine vorbestimmte Geometrie gefragt war, wurden die Regeln ausschließlich manuell angewandt. Dafür wählt man in der Liste die gewünschte Regel aus und stellt bei Match Selection auf manuell.

Wenn man nun auf apply rule drückt, öffnet sich ein Fenster, in dem alle Objekte aufgelistet werden, welche für die Regel erkannt wurden (Abbildung 19).

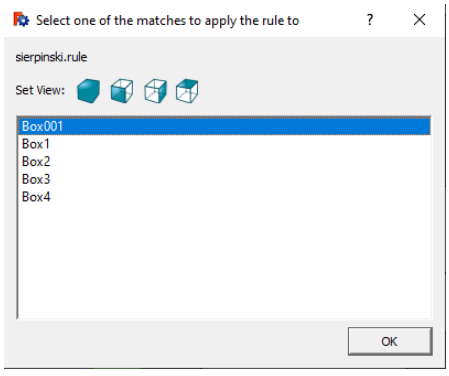

Abbildung 19: spapper apply rule Fenster, um Objekt auszuwählen

Spapper bietet allerdings auch die Möglichkeit Regeln automatisiert anzuwenden. Dabei kann unter anderem festgelegt werden, ob die Regeln zufällig ausgewählt werden, oder ob sie der Liste nach ablaufen. Zusätzlich kann auch die Anzahl der dargestellten Lösungen festgelegt werden. Außerdem gibt es Collision Detection. Standardmäßig ist die Collision Detection ausgeschaltet. Aktiviert man diese sorgt sie dafür, dass Regel, bei denen Formen kollidieren nicht durchgeführt

werden. Grundsätzlich werden Regeln in der Form a→b angewandt, was bedeutet, dass die LHS komplett durch die RHS ersetzt wird.

4.1.4 Regeln

In diesem Kapitel werden alle Regeln aufgezeigt, die erstellt wurden, um das Grundrissproblem zu lösen. Insgesamt wurden 13 Regeln erstellt. Allerdings wurde keine Regel erstellt, mit der eine Tür eingefügt werden kann, da diese in ihren Grundsätzen identisch, wie die Regel für ein Fenster ist.

Die Überlegung hinter diesen Regeln war es, den Grundriss wie Bausteine aufzubauen. Dementsprechend sind auch die Regeln danach ausgelegt.

Angefangen wird damit den Grundriss mit Hilfe von Bodenplatten zu erstellen. Dafür wurden zwei Regeln erstellt, die jeweils in positive x- und y-Richtung eine weitere Bodenplatte anbringen. Für volle Bewegungsfreiheit in alle Richtung muss man das gleiche auch für die negative x- und y-Richtung machen. Das wurde in diesem Beispiel allerdings ausgelassen, da sich der Grundaufbau der Regel nicht verändert hat.

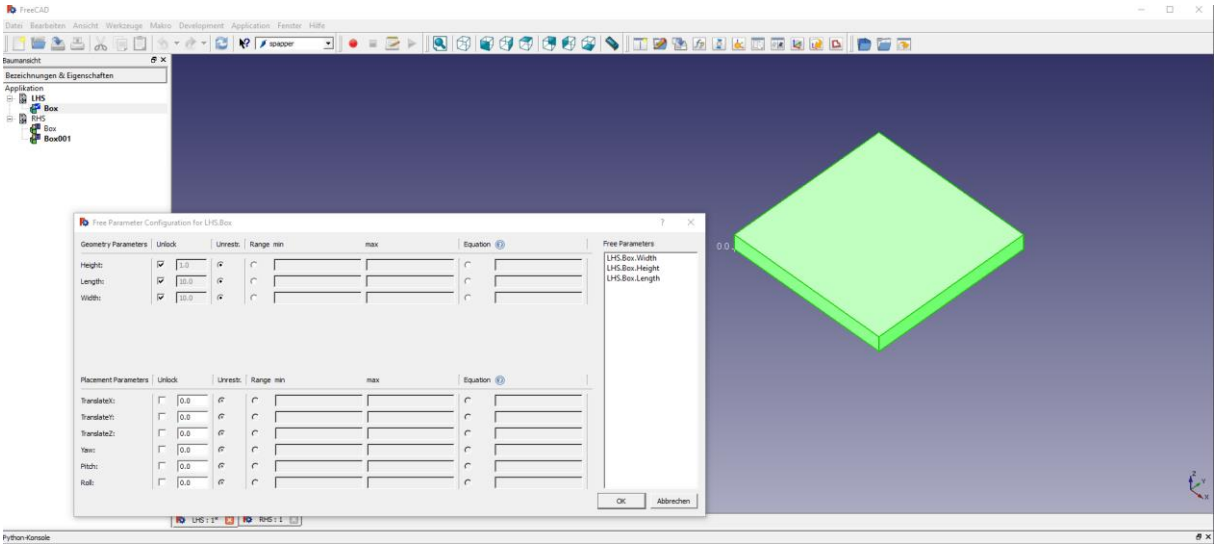

Abbildung 20: spapper Bodenregel x-Richtung LHS

Abbildung 20 zeigt die LHS der Bodenregel. Die LHS ist für die x- und y-Richtung identisch. Es wurden bewusst alle Parameter entsperrt, da die Bodenregeln als Anfangsregel gedacht sind und deshalb jeden Grundkörper erkennen sollen. In Abbildung 21 und 22 wird die RHS der Regel dargestellt. Die Höhe ist auf 1 festgelegt, da wie vorher beschrieben hauptsächlich mit 10/10/1 Körpern als Ausgang gearbeitet wurde. Box001 aus Abbildung 22 hat bei Platzierungsparameter die bereits im vorherigen Kapitel angesprochene x-Koordinate der LHS.Box.Length eingetragen.

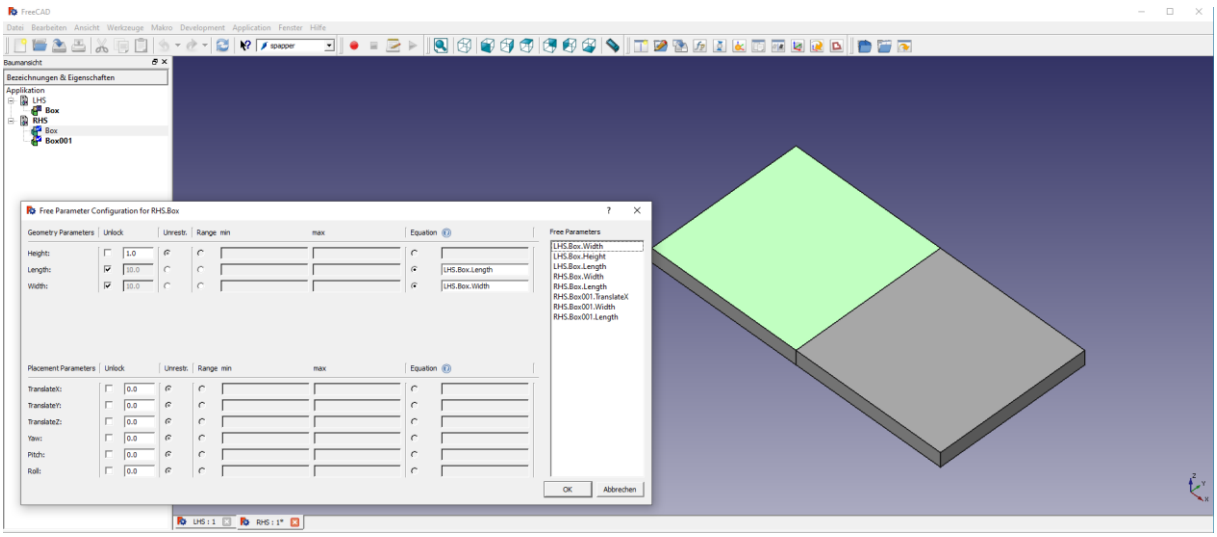

Abbildung 21: spapper Bodenregel x-Richtung RHS Box1

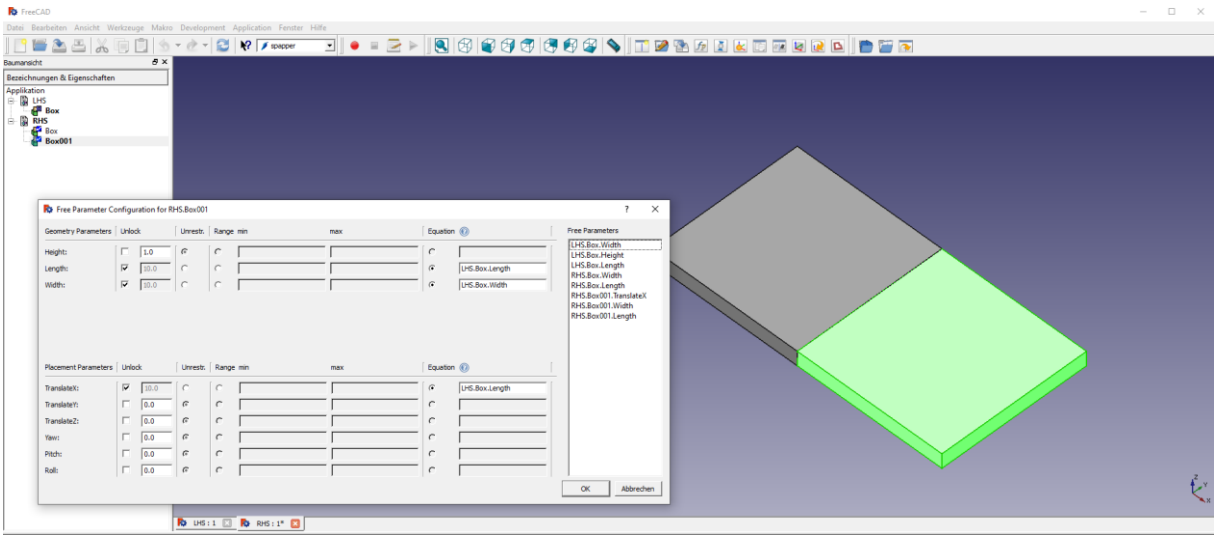

Abbildung 22: spapper Bodenregel x-Richtung RHS Box001

Hat man seinen Grundriss erstellt ist der nächste Schritt das Einfügen der Wände. Dafür wurde für jede Wandfläche des Quadrats eine Wandregel erstellt. Da sich die vier Regeln im Grundprinzip überschneiden, wird nur eine der vier Regeln grafisch dargestellt. Abbildung 23 zeigt die RHS der Wandregel in x-Ebene und über die y-Koordinate verschoben.

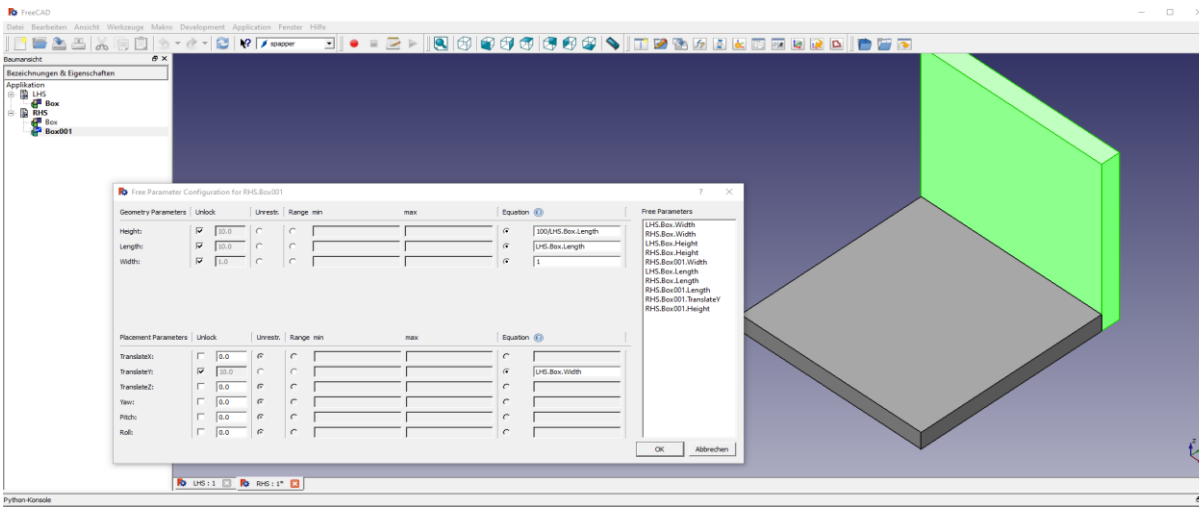

Abbildung 23: spapper Wandregel in x-Ebene über y-Koordinate verschoben, RHS

Die Höhe in der Wandregel wurde auf 100/LHS.Box.Lenght eingestellt. Das liegt daran, dass zusätzlich Regeln erstellt wurden, mit denen die Bodenplatten in ihrer breite und tiefe verdoppelt (Abbildung 24) und halbiert werden können. Die Wandregel wurde dementsprechend von den Parametern der Bodenplatte abhängig gemacht, wodurch eine größere Bodenplatte eine niedrigere Wand und eine kleine Bodenplatte eine höhere Wand hervorbringt. Der Grundgedanke dazu war, dass in Realität schmale Gebäude höher sowie breite/tiefe Gebäude flacher gebaut werden. Die 100 entsteht hier als Berechnungsparameter, da die Grundfläche einer Platte standardmäßig 10x10=100 ergibt.

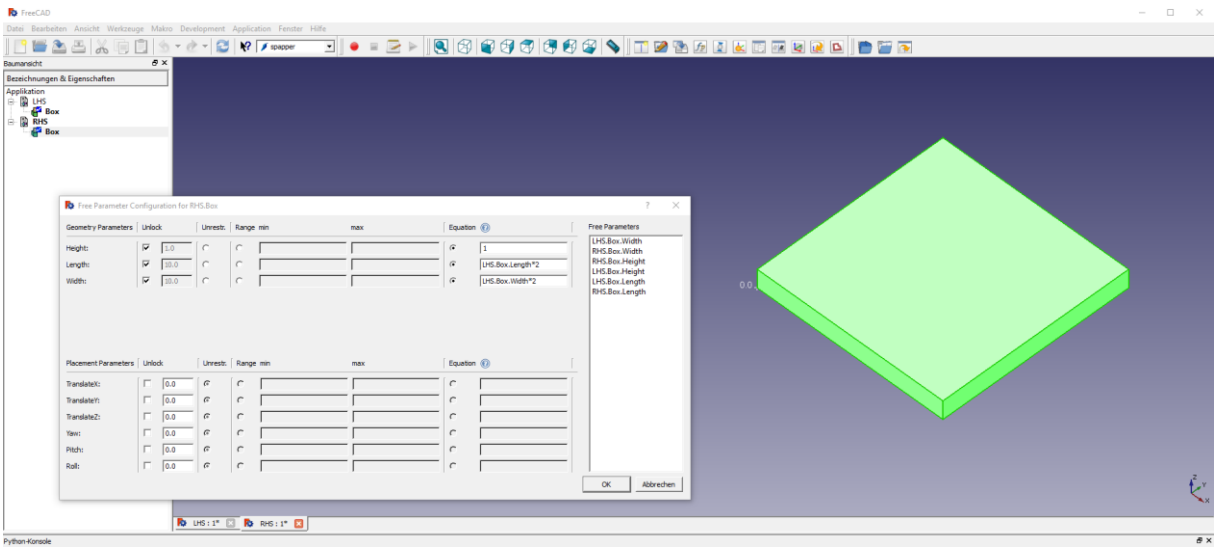

Abbildung 24: spapper Boden verdoppeln RHS

Für die Wand wurden zwei Fensterregeln erstellt. Einmal für die x-Ebene und für die y-Ebene. Die Fenster passen sich direkt den Ausgangsparametern der Wand an,

wodurch die Abhängigkeit zur Bodenplatte nicht benötigt wird. Das Fenster in Abbildung 25 ist dabei lediglich ein ausgeschnittener Teil der Wandscheibe, dessen Größe und Platzierung mit den Parametern der Wandscheibe skalieren.

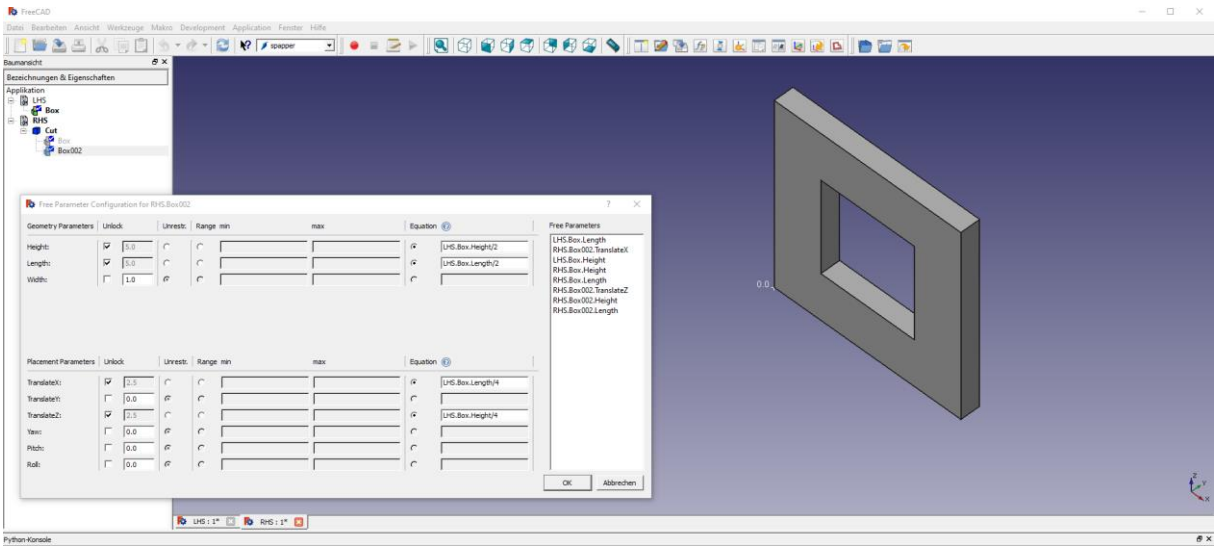

Abbildung 25: spapper Fenster Ausschnitt x-Ebene RHS

Mit den bis hierhin vorgestellten Regeln kann man bereits das Grundrissproblem lösen. Um die Regeln noch etwas abzurunden, wurden allerdings noch zusätzlich Regeln erstellt, mit denen ein Satteldach dargestellt werden kann.

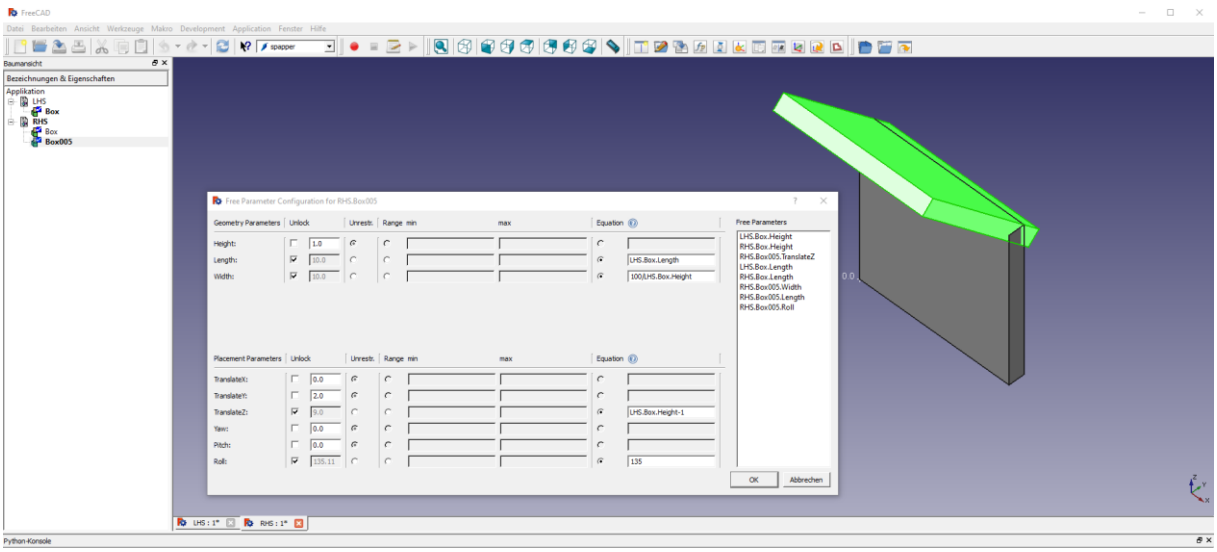

Abbildung 26: spapper Dach RHS

Abbildung 26 zeigt eine der beiden Regeln. Auch dem Dach wurde die Abhängigkeit zur Größe der Bodenplatte mitgegeben.

Abschließend ist anzumerken, dass alle hier erstellten Regeln in ihrer Komplexität weiter vertieft werden können. Die Regeln wurden ohne vorherige Einarbeit in Gestaltgrammatiken erstellt und dienen nur dem Zweck den Interpreter kennenzulernen.

4.2 GRAPE

4.2.1 Entstehung

Der Ansatz hinter GRAPE ist es Gestaltgrammatiken in graph grammar umzuordnen. Der Grund dafür ist, dass graph grammars zugänglicher für Computer Implementationen sind. Graphen sind eine beliebte Wahl, um die Struktur von Formen darzustellen (Heissermann, 1994; Kelles et al., 2009) (Übersetzung des Verfassers). Wie in Kapitel 2 beschrieben, verfolgen graph grammars unterschiedliche Ansätze. GRAPE bezieht sich auf die Darstellung Punkte und Linien zu Knoten (Thomas Grasl, 2011). GRAPE wird als Codename für GRAphs und shaPEs genutzt, was auf die Parallelen hinweisen soll.

Abbildung 27: GRAPE Interface

4.2.2 Interpreter

Auf GRAPE kann man im Browser unter der Seite grape.swap-zt.com zugreifen. Bis jetzt gibt es nur die App im Browser. GRAPE befindet sich in einer experimentellen Phase. Um alle Funktionen der App anwenden zu können braucht man einen Account. Der Account ermöglicht es eigene Grammatiken zu erstellen, diese zu speichern und anzuwenden. Ohne Account kann man nur die vorgegebenen Grammatiken anwenden (Abbildung 28).

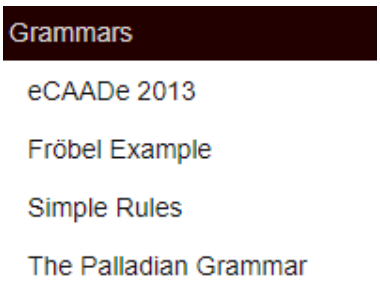

Abbildung 28: GRAPE gestellte Grammatiken

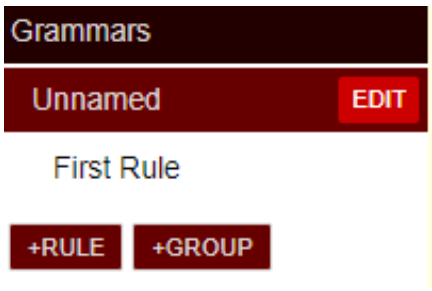

Abbildung 29: GRAPE Erstellungsmenü für neue Regeln

Das Interface von GRAPE ist eine große Zeichenfläche (Abbildung 27). An der linken Seite kann man zwischen den Grammatiken wechseln. Es werden vier fertige Grammatiken gestellt, mit denen man experimentieren kann (Abbildung 28).

Falls man mit einem Account eingeloggt ist, findet man unter den vorgegebenen Grammatiken ein rotes +grammar. Dort kann man seine eigenen Grammatiken erstellen.

Unter +rule wird eine neue Regel erstellt (Abbildung 29).

Dafür öffnet sich auf der rechten Seite die sogenannte properties area (Abbildung 30). Hier kann man der Regel einen Namen und eine Beschreibung geben, sowie Eigenschaften der Regel festlegen.

Das Feld Type bietet einem die Möglichkeit zwischen Editor, Macro und Manuel auszuwählen. Das sind drei unterschiedliche Arten Regeln zu erstellen. Bei Editor wird das Zeichenfeld genutzt während z.B. bei Manuel die Regel als GrGen.NET syntax beschrieben wird. In dieser Arbeit wurde nur mit dem Typ Editor gearbeitet.

Abbildung 31: GRAPE properties area

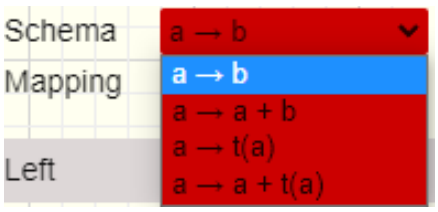

Abbildung 30: GRAPE Schema

Schema beschreibt die Art der Regelanwendung (Abbildung 31). So wird bei a→b die LHS gelöscht und durch RHS ersetzt. a→a+b zeichnet die RHS über die LHS. a→t(a) ist eine Transformationsregel, bei der die LHS gelöscht wird und die RHS als Ableitung von der LHS mit bestimmten Transformationseigenschaften wie z.B. einer Drehung

um 90° dargestellt wird. a→a+t(a) ist das gleiche wie a→t(a) nur das die LHS nicht gelöscht wird.

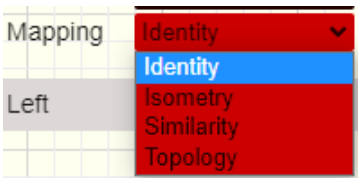

Abbildung 32: GRAPE Mapping

Mapping gibt einem die Möglichkeit zwischen vier verschiedenen Einstellungen auszuwählen (Abbildung 32). Diese Einstellung bestimmen, wie die Regeln Matches erkennen. Identity Regeln erkennt nur LHS die nicht transformiert wurden. Isometry erkennt LHS die transformiert wurden. Similarity erkennt auch LHS die transformiert wurden und zusätzlich noch skalierte LHS. Bei Topology sind nur die Nachbarlinien dafür entscheidend, ob die LHS erkannt wird oder nicht.

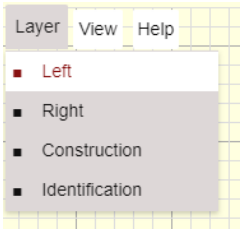

Abbildung 33: GRAPE Layer

Wenn man nun seine Objekte zeichnet, kann man bei Layer zwischen LHS, RHS, Construction und Identifiaction auswählen (Abbildung 33). Die gezeichneten Linien werden rechts im Interface angezeigt (Abbildung 34). Dort kann man per drag and drop Linien von LHS zu RHS und zurück verschieben. Mit Linksklick auf eine Linie können geometrische Angaben manuell eingegeben werden.

Mit Save kann man nun die erstellte Regel speichern.

Gespeicherte Regeln werden nun an der gleichen Stelle wie die vier vorgegeben Regeln dargestellt und können auch genauso genutzt werden, solange man eingeloggt ist.

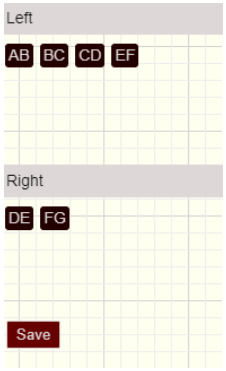

Abbildung 34: GRAPE LHS und RHS

4.2.3 Regelerstellung und Anwendung

GRAPE bietet eine Vielfalt an Einstellungsmöglichkeiten zur Regelerstellung. Im Rahmen dieser Bachelorarbeit wurde wie bereits oben beschrieben jede Regel mit dem Editor erstellt.

Da bereits mit der Einführung des Interface die Regelbearbeitungstools vorgestellt wurden, wird in diesem Kapitel nur noch auf die Regelanwendung eingegangen.

Regeln werden in der normalen CAD-Zeichenfläche angewandt. Führt man nun eine Regel aus, schaltet GRAPE automatisch in den Grammar Mode. Im Grammar Mode kann nicht mehr gezeichnet werden. GRAPE öffnet bei der Regelanwendung ein kleines Fenster (Abbildung 35). Dort werden alle möglichen Lösungen aufgezeigt. Die einzelnen Lösungen werden in der Zeichenfläche direkt angezeigt.

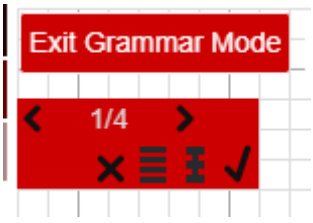

Abbildung 35: GRAPE Overlay für Regelanwendung

- Mit dem X kann der Vorgang abgebrochen werden.
- Der Haken wendet die ausgewählte Regel an.
- Das zweite Zeichen wendet alle möglichen Lösungen hintereinander an.
- Zeichen Nummer drei wendet alle Lösungen nacheinander an.

4.2.4 Regeln

Die erste Regel, mit welcher in GRAPE unser Beispiel gestartet wird, ist eine Regel, die ein Quadrat in die Form des Grundrissproblems bringt. Dabei gibt es unterschiedliche Möglichkeiten, wie man das darstellt. In unserem Fall wurde sich dafür entschieden, dass sich die Regel an unterschiedliche geometrische Maßangaben anpassen soll. Daraus sind zwei Regeln entstanden, einmal mit und einmal ohne Label, die beide ein Quadrat in die Grundform bringen. Die LHS wird in GRAPE immer schwarz dargestellt und die RHS rot.

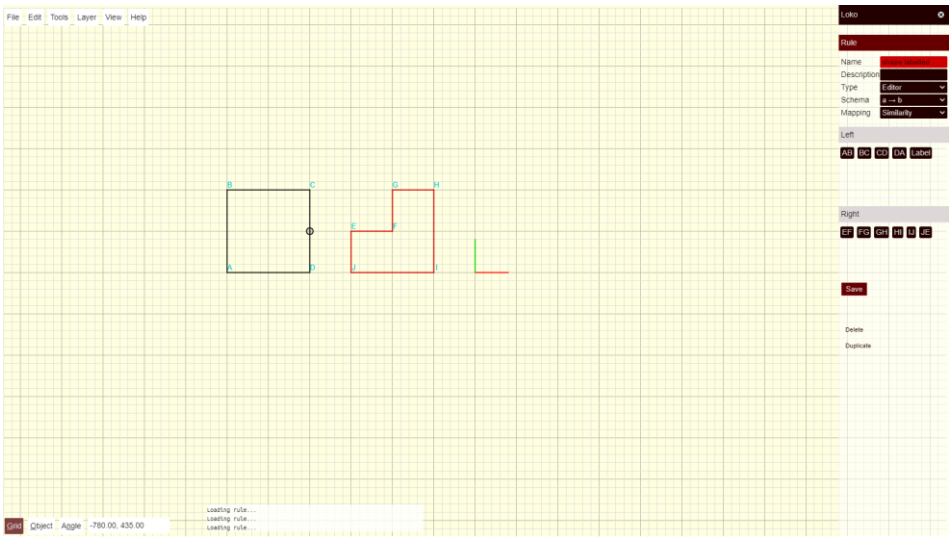

Abbildung 36: GRAPE Grundform mit Label

Die Regel mit Label bietet in der Anwendung 2 Lösungen, während die gleiche Regel ohne Label 8 Lösungen, also sechs weitere für die drei zusätzlichen Seiten, liefert.

Mit der nächsten Regel wird eine Raumeinteilung dargestellt. Da unser Grundriss sehr einfach gehalten ist, liefert diese Regel bei der Anwendung nur zwei Lösungen.

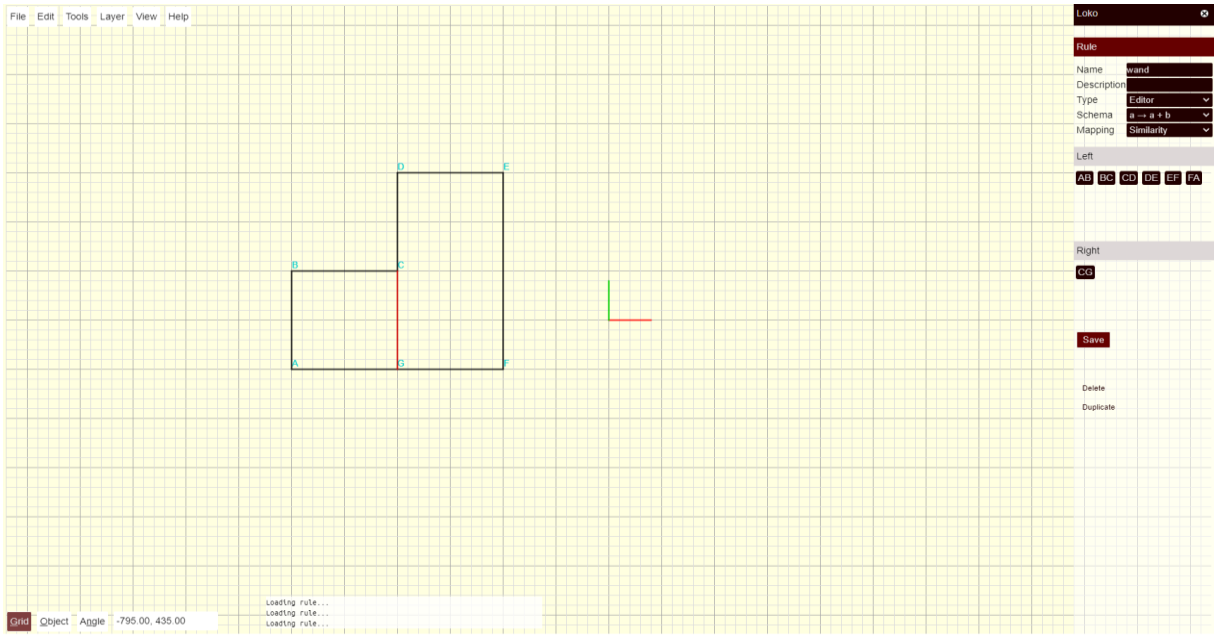

Abbildung 37: GRAPE Raumunterteilung

Zur Vollendung der Grundrissproblems fehlen noch zwei Regeln. Eine Regel für Fenster (Abbildung 38) und eine für die Tür (Abbildung 39). Der Grundaufbau der Regeln ist hier sehr identisch, weshalb sie ein gutes Beispiel sind, um den Unterschied in der Mapping Einstellung zwischen Topology und Similarity darzustellen.

Topology sorgt in der Fensterregel dafür, dass das Fenster alle Seiten des Grundrisses erkennt. Das bedeutet, dass das Fenster zwölf verschiedene Lösungen hat.

Similarity bei der Regel für die Tür stellt sicher, dass nur Kurven mit den gleichen Maßen erkannt werden, weshalb die Tür bei der Regelanwendung nur zwei verschieden Lösungen findet. Jeweils einmal an der langen Seite.

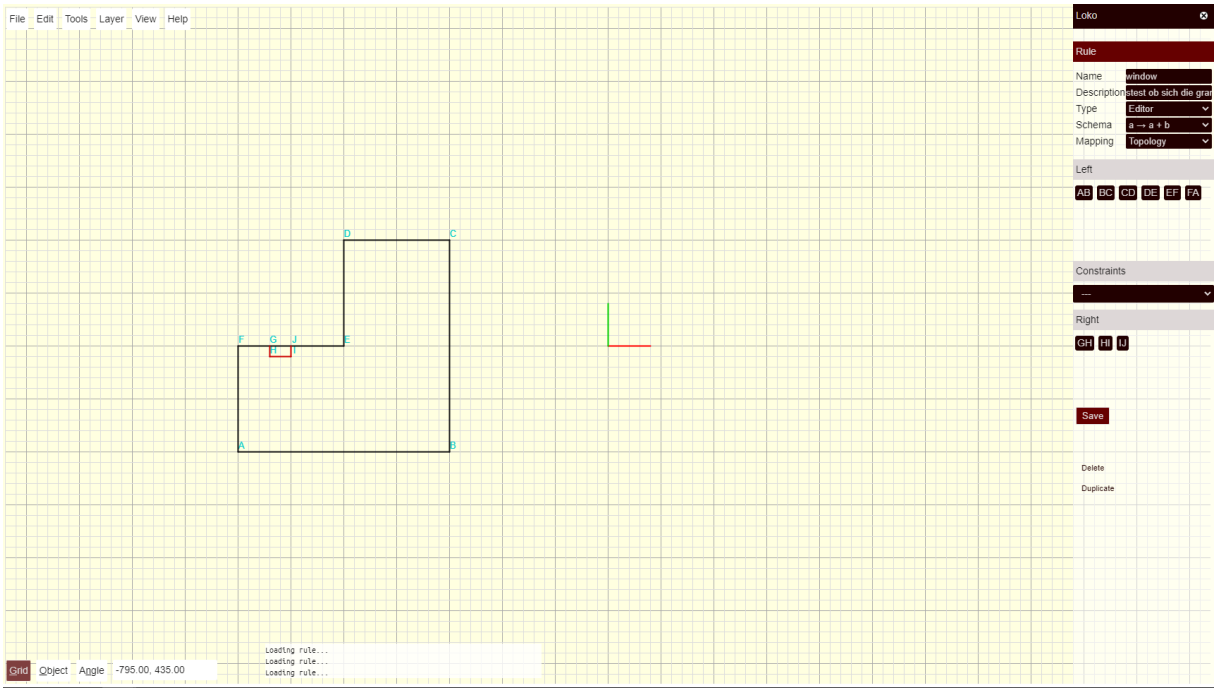

Abbildung 38: GRAPE Fenster

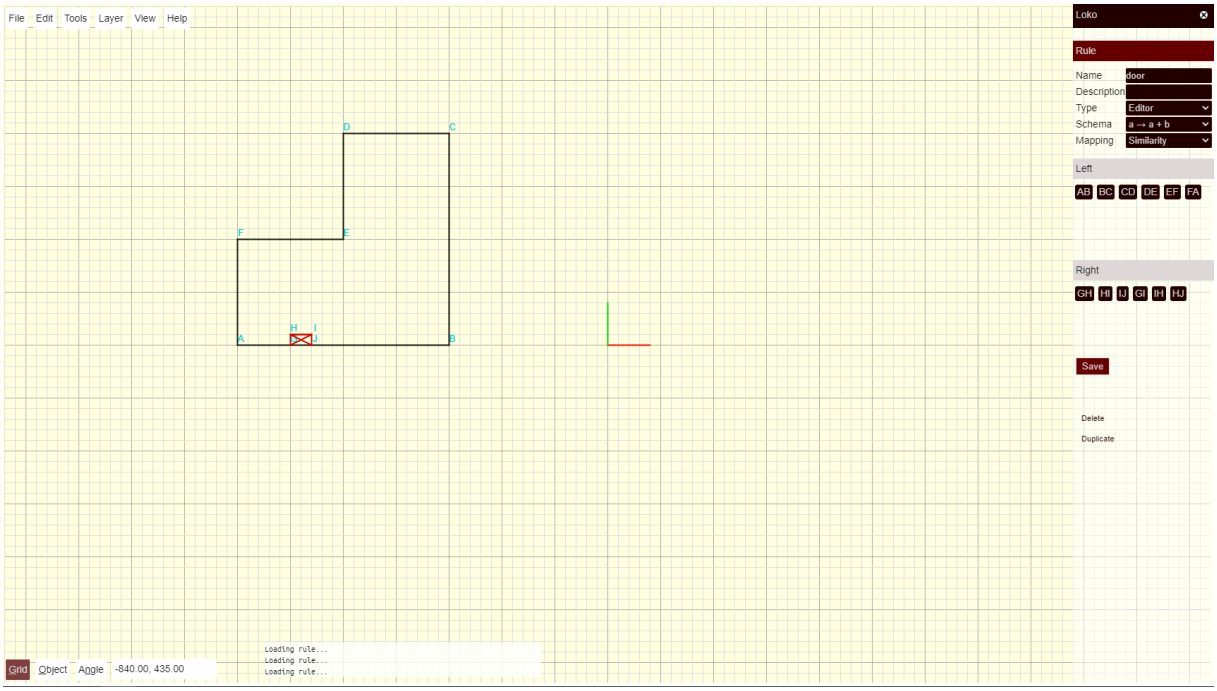

Abbildung 39: GRAPE Tür

4.3 sortal

4.3.1 Entstehung

Das Ziel hinter sortal war es, einen Interpreter zu erstellen, der in den frühen Phasen des Designs unterstützen kann. Dafür muss die Spezifikation und Nutzung von multiplen sich entwickelnden Darstellungen unterstützt werden (Rudi Stouffs 2007) (Übersetzung des Verfassers). Zusätzlich müssen noch Informationen zwischen diesen Darstellungen ausgetauscht werden. Dazu werden sorts genutzt. Die sortal Grammatik erweitert shape grammar und color grammar. Dadurch ist ein Interpreter entstanden, der shape grammar und color grammar mit dem Prinzip der sorts vereint.

4.3.2 Interpreter

Für sortal benötigt man das CAD Programm Rhino7, sowie die Erweiterung Grasshopper. Beides kann man auf der Rhino Seite Downloaden. Mit dem Download von Grasshopper folgt wie bereits bei spapper eine Anleitung zum Hinzufügen der Erweiterung, allerdings stellt sich diese für Anfänger als kompliziert dar. Hat man Rhino und Grasshopper erfolgreich installiert kann man Grasshopper in der Toolbar öffnen. Zeichnungen werden im Rhino Umfeld erstellt und die Regeln in der Grasshopper Anwendung mit grafischer Programmierung aufgebaut. Anders als bei der Vorstellung der vorherigen Interpreter werden hier nicht alle Funktionen besprochen, da sortal einen erheblich größeren Umfang hat und das somit im Rahmen dieser Bachelorarbeit nicht schaffbar ist.

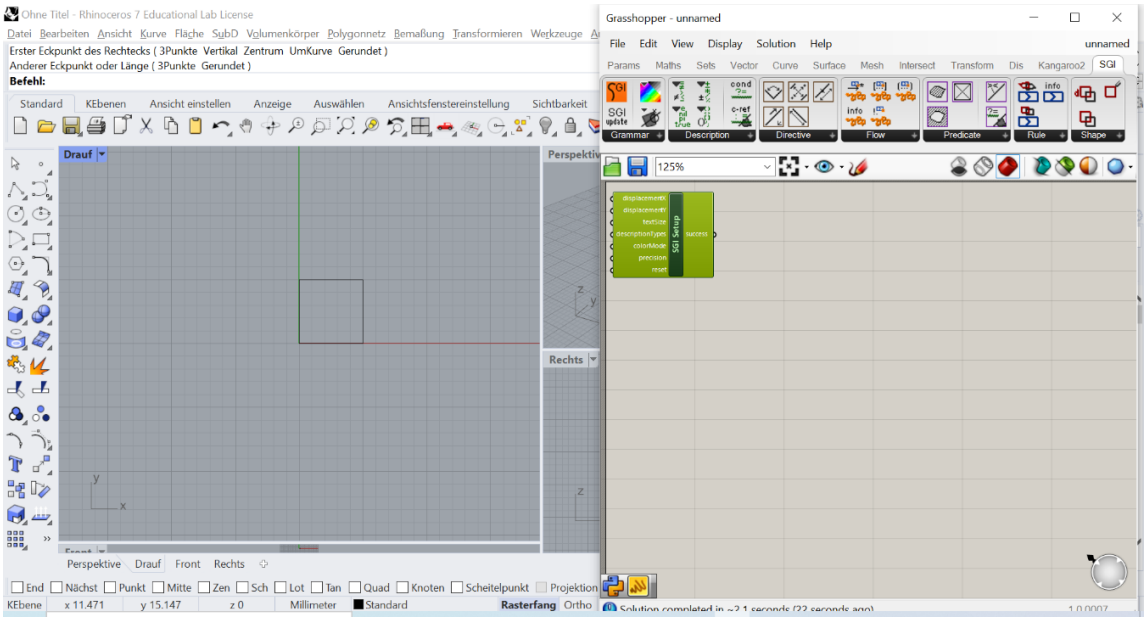

Abbildung 40: sortal CAD-Zeichenfläche und Grasshopper für visuelle Programmierung

4.3.3 Regelerstellung und Anwendung

Um eine Regel in sortal zu erstellen kann im CAD-Zeichenfeld (Abbildung 40) die gewünschte Form gezeichnet werden. Rhino bietet dafür eine Reihe an Standard CAD-Funktionen, um diesen Vorgang zu vereinfachen. Um gezeichnete Objekte in die Regel einzubinden, müssen diese zuerst z.B. als Curve in Grasshopper gekennzeichnet werden (Abbildung 41). Curves können in Grasshopper im Params Folder unter Geometry gefunden werden (Abbildung 42).

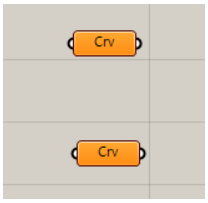

Abbildung 41: sortal zwei Cruves jeweils für LHS und RHS

Abbildung 42: sortal Geometrie Elemente in Grasshhopper

Ein Curve kann ein Objekt, oder auch mehrere Objekte enthalten.

Im nächsten Schritt müssen die Curves als Formobjekte gekennzeichnet werden. Dazu wählt man in Grasshopper im SGI Folder Shape aus (Abbildung 43). Hier bietet sortal unterschiedliche Möglichkeiten von Shapes an. Die Standardmäßige Auswahl ist SGI Shape.

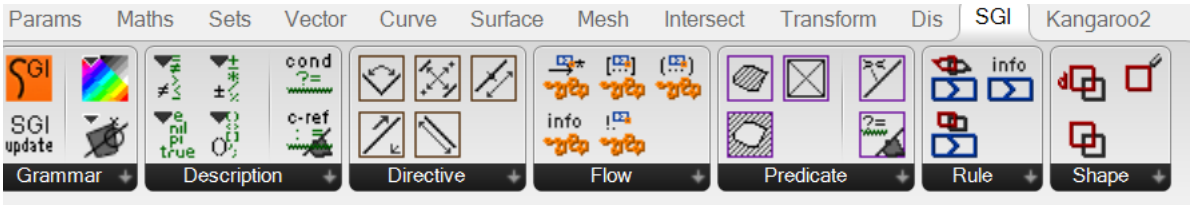

Abbildung 43: sortal SGI Toolbar

Um die Curves einer Shape zuzuweisen, muss man die gewählten Tools miteinander verbinden (Abbildung 44).

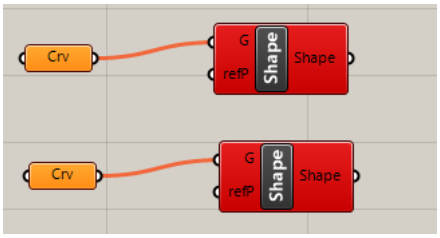

Abbildung 44: sortal Curve zu einer Shape zuweisen

Als nächstes werden die Shapes in eine Regel eingewiesen. Dafür wird bei den Tools SGI Rule ausgewählt. An dieser Stelle wird definiert welche Shape LHS und RHS ist. Jede Regel braucht zudem einen individuellen Namen.

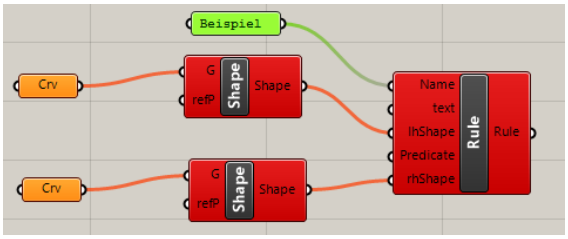

Abbildung 45: sortal Beispielregel

Abbildung 45 zeigt eine fertiggestellte einfache Regel. Sortal bietet eine Vielzahl an Möglichkeiten Regeln genauer zu definieren und individualisieren.

Um die erstellten Regeln anwenden zu können muss als erstes wie in Abbildung 44 ein Grundkörper definiert werden. Anschließend kann mit dem SGI Tool SGI Apply die Regel angewandt werden.

Abbildung 46 zeigt den kompletten Aufbau einer einfachen Regel mit anschließender Anwendung.

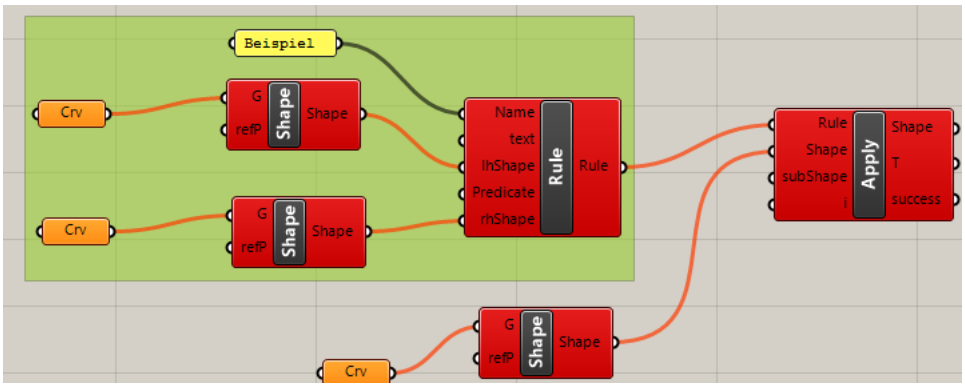

Abbildung 46: sortal Regel anwenden

Unter dem Punkt i kann die Anzahl festgelegt werden, wie oft die Regel angewandt wird. Standardmäßig ist i auf eine Lösung eingestellt. Das Ergebnis der Regelanwendung wird direkt in der CAD-Zeichenfläche angezeigt.

5 Analyse

5.1 Spapper

5.1.1 Nutzerfreundlichkeit

Spapper ist einfach zu installieren, allerdings gilt das nicht für FreeCAD, da die veraltete Version 0.11 für spapper benötigt wird. Hat man das allerdings geschafft bietet FreeCAD 0.11 und spapper ein selbsterklärendes Interface. Zusätzlich werden die Tools, welche spapper bereitstellt in einer beigeführten pdf-Datei erklärt. Ein Tutorial als Einführung gibt es allerdings nicht. Regeln erstellen, speichern und anwenden ist sehr einfach gehalten, allerdings müssen Regeln, nachdem sie editiert wurden wieder neu geöffnet und geladen werden, was dazu führen kann, dass das wiederholte Testen und Anpassen von vereinzelten Regeln sehr zeitaufwändig werden kann. Ein Problem das spapper mit sich bringt, ist das FreeCAD 0.11 beim Einfügen von bestimmten Objekten in die LHS oder RHS abstürzen kann.

5.1.2 Regelerstellung

Das einfache Interface sorgt dafür, dass Regeln in spapper einfach und schnell erstellt werden können. Dabei beschränkt sich spapper auf 3D Körper, was zur Folge hat, dass spapper keine Lösung für subshape detection bietet. Es ist allerdings möglich Regeln parametrisch zu erstellen. Mit dem FPC wird es dem Nutzer sehr einfach gemacht die einzelnen Objekte genau zu bearbeiten. Außerdem bietet spapper auch ein Tool für Label. Die Form und Menge an Objekten ist frei definierbar, das gilt für die LHS sowie dir RHS. Bei der Regelerstellung werden die Objekte grafisch dargestellt. Zusätzlich sind Regeln jederzeit editierbar.

5.1.3 Regelanwendung

Spapper bietet die Möglichkeit eine beliebige Anzahl an Regeln zu öffnen. Die Regeln werden in einer Liste aufgezeigt. In dieser Liste kann jede einzelne Regel individuell ausgewählt oder abgewählt werden. Regeln können automatisch semi-automatisch oder manuell angewandt werden. Die Anzahl der Regelanwendungen ist frei wählbar, sowie die Anzahl an Lösungen, welche dargestellt werden sollen. Bei einer manuellen

Anwendung wird jedoch der Lösungsvorschlag nicht grafisch dargestellt, bis die Regel durchgeführt wurde. Zusätzlich bietet spapper Collision detection.

Spapper erkennt keine Regelmäßigkeiten bei den erstellten Regeln, weshalb in unserem Beispiel viele fast identische Regeln erstellt werden mussten (z.B. vier Regeln für die Wandplatten).

5.1.4 Anwendungsfall

Um spapper anwenden zu können bedarf es kein großes Vorwissen im Bereich Gestaltgrammatiken. Dadurch, dass man nur die Anwendung mit dem FPC lernen muss, eignet sich spapper gut als Einstieg in 3D Interpreter. Außerdem können die erstellten 3D-Objekte im Anschluss z.B. mit einem 3D-Drucker in eine physische Form gebracht werden.

5.2 GRAPE

5.2.1 Nutzerfreundlichkeit

GRAPE benötigt keine Installation, da die Browser App eine CAD-Zeichenfläche bietet mit der gearbeitet werden kann. Die Auswahl der Regeln sowie die Anwendung der Tools ist sehr übersichtlich gestaltet. Zusätzlich zu dem intuitiven Interface gibt es eine Dokumentation von allen Tools und Einstellungen, die in GRAPE vorhanden sind. Regeln lassen sich einfach erstellen und sofort testen, indem man GRAPE mit einem weiteren Tab erneut öffnet und dort durch neues Laden der Seite direkt die gespeicherten Regeln anwenden kann.

5.2.2 Regelerstellung

Besitzt man einen Account bietet GRAPE eine gute Basis, um 2D Regeln zu erstellen. Dabei kann man Regeln grafisch oder auch mit Gr.Gen.NET syntax erstellen. Bei der Komplexität der Regel ist dem Nutzer mit einer Vielzahl an Einstellungsmöglichkeit sehr viel Freiraum gegeben. Teil davon sind unter anderem parametrische Regeln sowie Label und Transformationen. Um alle Möglichkeiten die GRAPE einem liefert nutzen zu können, sind jedoch Vorkenntnisse im Umgang mit Gestaltgrammatiken notwendig.

5.2.3 Regelanwendung

GRAPE schlägt bei Anwendung der Regel auf eine Form direkt alle Lösungen vor. Man kann die Lösungen manuell durchsuchen und auswählen, da die Formen direkt im CAD-Feld dargestellt werden. Es wird allerdings auch die Möglichkeit geboten alle möglichen Lösungen hintereinander durchzuführen oder aber auch alle Lösungen gleichzeitig anzuwenden. Zusätzlich besitzt GRAPE subshape detection, womit neu entstandene Flächen unter den richtigen Voraussetzungen direkt erkannt werden.

GRAPE erkennt bei der Regelanwendung gleiche Geometrien. So war es möglich in unserem Beispiel mit einer Regel für Fenster zwölf verschieden Lösungen zu erhalten.

5.2.4 Anwendungsfall

GRAPE eignet sich besonders gut als Einstieg in das Arbeiten mit Gestaltgrammatiken. Das liegt zu einem daran, dass GRAPE ohne Installation schnell und einfach eine Möglichkeit bietet verschiedene Gestaltgrammatiken auszutesten. Zusätzlich hat GRAPE ein leicht zu verstehendes Interface mit einer guten Dokumentation über alle Funktionen die GRAPE zu bieten hat. Durch das Angebot an unterschiedlichen Gestaltgrammatiken wie z.B. der Palladian grammar, reizt GRAPE den Nutzer sich vertieft mit Gestaltgrammatiken auseinanderzusetzen und sein Wissen zu erweitern.

5.3 Sortal

5.3.1 Nutzerfreundlichkeit

Rhino7 und auch sortal sind einfach als Download zu finden, allerdings ist das Einfügen von der GrasshopperSGI Erweiterung in Rhino7 im Vergleich zu spapper sehr kompliziert und für Laien, die sich in diesem Bereich nicht auskennen schwer durchzuführen. Rhino7 und die Grasshopper Erweiterung bieten eine Vielzahl an Tools an, was im ersten Moment das Interface sehr überfordernd wirken lässt. Allerdings sind alle Tools gut dokumentiert und dadurch schnell zu verstehen. Sortal bietet dazu auf ihrer Homepage eine Vielzahl an Tutorials an, in denen anhand von Beispielen gezeigt wird, wie sortal funktioniert und angewandt werden kann.

Die CAD-Zeichenfläche reagiert sofort auf Änderungen in Grasshopper. Dadurch kann ohne Aktualisierung oder andere Ladezeiten direkt eine Regel getestet und editiert werden.

5.3.2 Regelerstellung

Sortal bietet eine 2D CAD-Zeichenfläche in der Formen erstellt werden können. Gleichzeitig kann in der Grasshopper Erweiterung die Regel mittels visueller Programmierung bearbeitet werden. Es ist möglich Regeln parametrisch zu definieren, sowie Label einzufügen. Außerdem kann sortal auch zwischen Farben und Gewichtungen unterscheiden. Sortal bietet im Vergleich mit den beiden anderen Interpretern die meisten Möglichkeiten Regeln zu definieren. Zusätzlich hat die Art der Regelerstellung in Grasshopper den Vorteil, dass man alle Regeln immer im Überblick hat und diese flexibel miteinander kombinieren kann. Das gilt für die Regelerstellung, sowie die Anwendung.

5.3.3 Regelanwendung

Die Regelanwendung in sortal ist sehr einfach und trotzdem tiefgreifend. So ist die Standardeinstellung in SGI Rule, dass die Regel einmal durchgeführt wird und das Ergebnis direkt in der CAD-Zeichenfläche angezeigt wird. Die Anzahl der Regelanwendungen kann dabei frei gewählt werden. Es gibt auch die Möglichkeit alle möglichen Regeln auf einmal, sowie alle Regeln in Reihe anzuwenden. Möchte man allerdings nicht, dass die Regel angewandt wird, sondern nur die Möglichen Matches betrachten, liefert sortal auch dafür ein Tool. Wenn man in den Matches eine Lösung gefunden hat, die man als Regel durchführen möchte kann man in dem Regelfeld die Nummer des Matches eingeben. Sortal liefert bis jetzt bei den Matches noch keine Nummerierung, weshalb man diese händisch durchzählen muss. Allgemein sind Regeln und die Anwendung sehr einfach zu bearbeiten und umzustellen, was an der visuellen Programmierung liegt.

5.3.4 Anwendungsfall

Sortal ist besonders für Nutzer geeignet, die sich mit Gestaltgrammatiken auskennen und in der Lage sind komplexe Regeln zu erstellen. Die visuelle Programmierung ermöglicht es einem eine beliebige Anzahl an verschiedene Vorlagen zu erstellen und diese immer wieder erneut zu verwenden.

5.4 Resume

Bei der Wahl des geeigneten Gestaltgrammatik Interpretern sollte neben den Möglichkeiten, die diese Interpreter liefern auch bedacht werden, was das konkrete Ziel der Nutzung von dem Interpreter ist. So macht es keinen Sinn den Interpreter mit den meisten Anwendungsmöglichkeiten zu wählen, wenn man nur ein Gefühl für das Arbeiten mit Gestaltgrammatiken bekommen will. Das liegt vor allem daran, dass je mehr Möglichkeiten der Interpreter liefert, desto komplexer er auch aufgebaut ist. Eine weiter Überlegung, die man vor der Wahl des Interpreters auch noch treffen sollte, ist, ob man neue Formen entdecken möchte, oder ob man vorher definierte Formen darstellen möchte. In dieser Bachelorarbeit sollte eine im Voraus ausgedachte Form erstellt werden. Das hat bei einem Interpreter wie spapper hervorragend funktioniert, da man bei dieser Implementation die Regeln wie einzelne Bausteine aufeinander aufbauen konnte, bis man die gewünschte Form erreicht hat. So hat man sich nie in der Anwendung eingeschränkt gefühlt. Bei Interpretern wie GRAPE und sortal konnte die Aufgabenstellung zwar mit wenig Aufwand erfüllt werden, allerdings hatte man bei der Erstellung der Regeln nie das Gefühl das volle Ausmaß der Grammatik ausschöpfen zu können. Das lag vor allem daran das sortal und GRAPE zwei Interpreter sind, bei denen das Ziel neue Formen zu entdecken und nicht vordefiniertes nachzubilden ist. Bei diesen Aussagen ist jedoch anzumerken, dass vor der Anwendung der drei Interpreter keine Vorkenntnisse zur Arbeit mit Gestaltgrammatiken gegeben war und die Komplexität und tiefe aller Regeln mit einer entsprechenden Vorkenntnis geschärft werden können. Als Randbemerkung ist noch zu vermerken, dass GRAPE in dieser Arbeit als 2D-Interpreter genutzt wurde, jedoch bietet GRAPE eine vordefinierte Reihe an Regeln, die im 3-dimensionalen Raum Anwendung finden. Die CAD-Fläche kann in der Ansicht auch von 2D auf 3D umgestellt werden, jedoch war es beim Regelerstellen nicht möglich das zu replizieren.

In Tabelle 1 wird die Analyse der drei Interpreter visuell dargestellt. Die einzelnen Interpreter werden jeweils in einer Spalte dargestellt und jede Zeile beinhaltet einen Bewertungspunkt.

Tabelle 1: Analyse visualisiert

	spapper	**GRAPE**	**sortal**
2D/3D	3D	2D	2D
parametrisch	✓	✓	✓
Subshape detection	X	✓	✓
Label	✓	✓	✓
Farbenunterteilung möglich	X	X	✓
Erkennt Regelmäßigkeiten zwischen Regel und Form	X	✓	✓
Automatische Regelanwendung möglich	✓	✓	✓

6 Zusammenfassung und Fazit

Gestaltgrammatiken konnten sich bis jetzt noch nicht als Lösung für Designerstellung durchsetzen. Das liegt vor allem daran, dass trotz Jahrzehnte langer Forschung noch keine Lösung dazu gefunden wurde, wie man einen idealen Gestaltgrammatik Interpreter erstellen kann. Die Anforderungen an einen solchen Interpreter sind sehr komplex.

Es werden allerdings immer mehr Gestaltgrammatik Interpreter erstellt, welche vereinzelte Teilbereiche von Gestaltgrammatiken abdecken. Einige davon dienen nur als Prototyp um Forschungsergebnisse darzustellen. Andere werden noch bis heute Regelmäßig geupdated.

Die drei in dieser Arbeit vorgestellten Interpreter zeigen, wie stark sich die Herangehensweise in der Implementierung von Gestaltgrammatiken unterscheidet. Damit sich Gestaltgrammatiken tatsächlich etablieren können, muss vor allem eine einheitliche Struktur erschaffen werden, nach denen sich die Interpreter richten können. Ein Beispiel dafür sind CAD Programme, die viele unterschiedliche Auslegungen haben. Jedoch kommt man mit fast allen zurecht, nachdem man das Standardverfahren gelernt hat. Gestaltgrammatiken Interpreter sind in dieser Sicht noch zu divers.

Literaturverzeichnis

Agarwal, Manish; Cagan, Jonathan; Stiny, George (2000): A Micro Language: Generating MEMS Resonators by Using a Coupled Form — Function Shape Grammar. In: *Environ Plann B Plann Des* 27 (4), S. 615–626. DOI: 10.1068/b2619.

Chau, Hau Hing; Chen, Xiaojuan; McKay, Alison; Pennington, Alan de (2004): Evaluation of a 3D Shape Grammar Implementation. In: John S. Gero (Hg.): Design Computing and Cognition ’04. Dordrecht: Springer Netherlands, S. 357–376.

Gero, John S. (Hg.) (2004): Design Computing and Cognition ’04. Dordrecht: Springer Netherlands.

Hoisl, Frank; Shea, Kristina (2011): An interactive, visual approach to developing and applying parametric three-dimensional spatial grammars. In: *AIEDAM* 25 (4), S. 333–356. DOI: 10.1017/S0890060411000205.

James, Gips; Boston, College (1999): Computer implementation of shape grammars 1999. In: *Workshop on shape Computation MIT*.

Jowers, Iestyn; Earl, Chris; Stiny, George (2019): Shapes, structures and shape grammar implementation. In: *Computer-Aided Design* 111, S. 80–92. DOI: 10.1016/j.cad.2019.02.001.

Krishnamurti, Ramesh: Shape recognition in three dimensions. In: *Enviroment and Planning B:Planning and Design* 1992 (19), S. 585–603.

Krishnamurti, Ramesh; Stouffs, Rudi (2004): The boundary of a shape and its classification. In: *JDR* 4 (1), S. 0. DOI: 10.1504/JDR.2004.009843.

Lienhard, Stefan: Visualization, Adaptation, and Transformation of Procedural Grammars.

McKay, Alison; Chase, Scott; Shea, Kristina; Chau, Hau Hing (2012): Spatial grammar implementation: From theory to useable software. In: *AIEDAM* 26 (2), S. 143–159. DOI: 10.1017/S0890060412000042.

Ramesh, Krishnamurti; Rudi, Stouffs (1993): spatial grammars: Motivation, Comparison, and New Results. In: *CAAD Futures ‘93*, S. 57–75.

Rudi, Stouffs: on Shape Grammars, Color Grammars abd Sortal Grammars. In: *e-CAAADe 30*.

Stouffs, Rudi (2008): Constructing design representations using a sortal approach. In: *Advanced Engineering Informatics* 22 (1), S. 71–89. DOI: 10.1016/j.aei.2007.08.007.

Thomas, Grasl: GRAPE_A_parametric_shape_grammar_impleme.

Wonka, Peter; Wimmer, Michael; Sillion, Francois; Ribarsky, WIlliam: Instant Architectre 2003.

Wortmann, Thomas; Stouffs, Rudi (2018): Algorithmic complexity of shape grammar implementation. In: *AIEDAM* 32 (2), S. 138–146. DOI: 10.1017/S0890060417000440.

Zimmermann, Luca; Chen, Tian; Shea, Kristina (2017): Generative Shape Design Using 3D Spatial Grammars, Simulation and Optimization. In: John. S. Gero (Hg.): Design Computing and Cognition '16. Cham: Springer International Publishing, S. 279–297. (Hoisl und Shea 2011; James und Boston 1999; Gero 2004; Chau et al. 2004; Agarwal et al. 2000; Jowers et al. 2019; Krishnamurti und Stouffs 2004; Krishnamurti)

(Lienhard; McKay et al. 2012; Ramesh und Rudi 1993; Rudi; Stouffs 2008; Thomas; Wonka et al.; Wortmann und Stouffs 2018; Zimmermann et al. 2017)

Anhang A

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelor-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 2. Mai 2022

Vorname Nachname

Christian Exner

Siegmund-Schacky Str.18

D-80993 München

chrisexner5@gmail.com