HAW HAMBURG

BACHELORTHESIS
Adrian Helberg

Template-basierte Synthese von Verzweigungsstrukturen mittels L-Systemen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Adrian Helberg

Template-basierte Synthese von Verzweigungsstrukturen mittels L-Systemen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: -

Eingereicht am: 18. Februar 2021

Adrian Helberg

Thema der Arbeit

Template-basierte Synthese von Verzweigungsstrukturen mittels L-Systemen

Stichworte

Computergrafik, prozedurale Modellierung, 3D Generierung, L-Systeme, formale Grammatik, prozedurale generierung

Kurzzusammenfassung

Hier kommt das Abstract hin ...

Adrian Helberg

Title of Thesis

Template-bases synthesis of branching structures using L-Systems

Keywords

Computer graphics, Procedural modeling, 3D generation, L-Systems, formal grammars, procedural generation

Abstract

The abstract belongs to here ...

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

HAW Hochschule für Angewandte Wissenschaften.

Symbolverzeichnis

Ω unit of electrical resistance.

Listings

1 Einleitung

Hochschule für Angewandte Wissenschaften (HAW) Ω HAW Hamburg(Damit das glossar keine fehler schmeißt)

Effizientes Objektdesign und -modellierung sind entscheidende Kernkompetenzen in verschiedenen Bereichen der digitalen Welt. Da die Erstellung geometrischer Objekte für Laien unintuitiv ist und ein großes Maß an Erfahrung und Expertise erfordert, ist dieses stetig wachsende Feld für Neueinsteiger nur sehr schwer zu erschließen. Die Forschung liefert hierzu einige Arbeiten zur prozeduralen Modellierung, um digitale Inhalte schneller und automatisiert zu erstellen. Gerade wenn es um die Darstellung natürlicher Umgebung geht, ist die Erstellung von ähnlichen Objekten, wie zum Beispiel verschiedene Bäume derselben Gattung eines Waldes, ein schwieriges Problem. Kleine Änderungen in prozeduralen Systemen können zu großen Veränderungen der Ergebnisse führen. Darum beschäftigt sich die inverse prozedurale Modellierung unter anderem mit dem Inferieren von Regeln und Mustern aus gegeben Objekten, um diese nach bestimmten Regeln zu modellieren. Ein wichtiges Werkzeug hierbei ist die Verwendung formaler Grammatiken als fundamentale Datenstruktur der Informatik, um Strukturen zu beschreiben. Eine spezielle Untergruppe sind die L-Systeme, die häufig bei der Beschreibung von Verzweigungsstrukturen und Selbstähnlichkeit zum Einsatz kommen.

Diese Arbeit soll sich mit der Erstellung eines prozeduralen Systems zur Synthese von Ähnlichkeitsabbildern beschäftigen. Hierzu soll über eine Benutzerschnittstelle eine Struktur erzeugt werden, aus der ein parametrisiertes L-Systems inferiert werden kann, das dann zur Generierung von ähnlichen Strukturen genutzt werden kann.

1.1 Problemstellung

Mit der Digitalisierung der Welt steigt auch der Bedarf an digitalen Inhalten. Zu den größten Feldern gehören Gaming- und Unterhaltungsindustrie, Datenvisualisierung und interaktive Anwendungen. Um eine erhöhte Quantität dieser Inhalte liefern zu können, werden Methodiken und Algorithmen gesucht, die eine Erstellung vereinfachen. Während Methodiken zur Kodifizierung bestimmter Strukturen in den Bereich der prozeduralen Modellierung fällt, findet das Ableiten von Regeln Anwendung in der inversen prozeduralen Modellierung. Weiter steigt mit dem digitalen und naturwissenschaftlichen Fortschritt die Anwendung immer komplexerer Strukturen, die ein Herausarbeiten der schwer zu kontrollierenden, prozeduralen Regeln immer schwieriger machen.

Ein Beispiel hierzu aus der Gaming-Industrie ist die frühere Verwendung unorganisierter Modelle. Unorganisierte Modelle sind nur bedingt Wiederverwendtbar, da nur die vorliegende Modellierung verwendet werden kann. Es besteht eine hohe Speicherkomplexität bei geringer Laufzeitkomplexität. Für kleinste Veränderungen am Modell ist eine erneute Modellierung nötig, die wiederum Speicher benötigt, um sie persistent speichern zu können. Heutzutage werden die Objekte nach bestimmten Kriterien organisiert, um eine austomatisierte Modellierung durch Algorithmen zu ermöglichen, um so aus einer Grundstruktur weitere Modelle zu erzeugen. Speicher- und Laufzeitkomplexität nähern sich an. Deshalb werden allgemeingültige, vielseitig anwendbare Algorithmen gesucht, die bestimmte natürliche Eigenschaften von Strukturen herausarbeiten („Reverse Engineering“), um diese für die (inverse) prozedurale Modellierung zur Verfügung zu stellen.

Diese Arbeit soll zeigen, wie sich durch die Erstellen eines Systems zur Generierung von ähnlichen Strukture aus einer Basistruktur aktuelle Ansätze aus der Forschung in einem Programm umsetzen lassen.

1.2 Ziele

Die Erstellung eines Systems zur Synthese von ähnlichen Struktures aus einer Basisstruktur soll zentrale Aufgabe dieser Arbeit sein. Aus der Fragestellung leiten sich folgende Teilziele ab:

- Die Erstellung eines Programms zur Umsetzung des erstellten Systems
- Anwenden von Algorithmen und Ansätzen der aktuellen Forschung
- Testen von Metriken und deren Auswirkung auf das Ergebnis
- Erstellung eigener Methodiken und Algorithmen zur Effizienten Lösung der Problemstellung
- Schaffen eines Teilsystems zum Extrahieren von Eigenschaften und Regeln einer Basisstruktur

Die Erstellung und Integration eines neuronalen Netzes, das laut aktueller Forschung gute Ergebnisse beim Lernen von Regeln aus einer Eingabestruktur liefert, kann in dieser Arbeiten aus Praktikabilitätsgründen nicht behandelt werden.

1.3 Methodik

Der Benutzer des Systems legt atomare Strukturen in Form von Zeichenketten an, die vom Programm eingelesen werden und als Templates zur Verfügung gestellt werden. Die Templates sind beliebig und können eine einfach Linie oder eine komplexe Verzweiung darstellen. Werden diese Strukturen auf der graphischen Oberfläche platziert und mit diversen Transformationen verändert, spricht man von Instanzen oder Template-Instanzen.

Strukturieren

Der Benutzer nutzt die grafische Benutzerschnittstelle, um aus einzelnen Templates eine zusammenhängende Basisstruktur zu erstellen. Neben der Position der Instanzen werden Transformationensparameter, wie Rotation oder Skalierung, angepasst.

Visualisieren

Der aktuelle Stand der Strukturierung ist jederzeit sichtbar. Liniensegmente und Bindungselemente werden in einem graphischen Element visualisiert und für eine Interaktion zur Verfügung gestellt.

Datenaufbereitung

Das Ergebnis der Strukturierung wird in einer Baumstruktur organisiert, in der jeder Knoten einer bestimmten Template-Instanz entspricht und die zugehörigen Kanten die geometrischen Transformationen relativ zum Elternknoten beschreibt.

Inferieren

Aus der Baumstruktur kann eine formale Grammatik in Form eines L-Systems abgeleitet werden. Diese Grammatik beschreibt lediglich die erstellte Basisstruktur und beinhaltet keine Transformationsparameter, da hier nur auf die Topologie des Baumes und nicht auf geometrische Unterschiede der Instanzen eingegangen wird.

Komprimieren

Um sich wiederholende Produktionregeln zu verhindern und so sowohl das Alphabet, also auch die Produktionsregelmenge zu kompromieren, wird die Baumstrukturen nach identischen, maximalen Unterbäumen durchsucht und durch zusammengefasste Instanzen vereinfacht. Hierbei gilt die Baumstruktur selbst nicht als Unterbaum.

Generalisieren

Ähnliche Produktionsregeln des L-Systems werden mithilfe einer Kostenfunktion zusammengefasst, um dises mit nicht-deterministischen Regeln und Rekursion zu generalisieren.

Randomisieren

Jedes Symbol der Grammatik nimmt eine Liste an Parametern entgegen, die nach bestimmten Kriterien pseudo-randomisiert werden, um Variationen von Template-Instanzen zu erstellen. Das Ausführen des L-Systems kann nun Ähnlichkeitsstrukturen für die Basisstruktur erzeugen.

1.4 Aufbau

Die Methodik zum Umsetzen des beschriebenen Systems wird wie folgt umgesetzt.

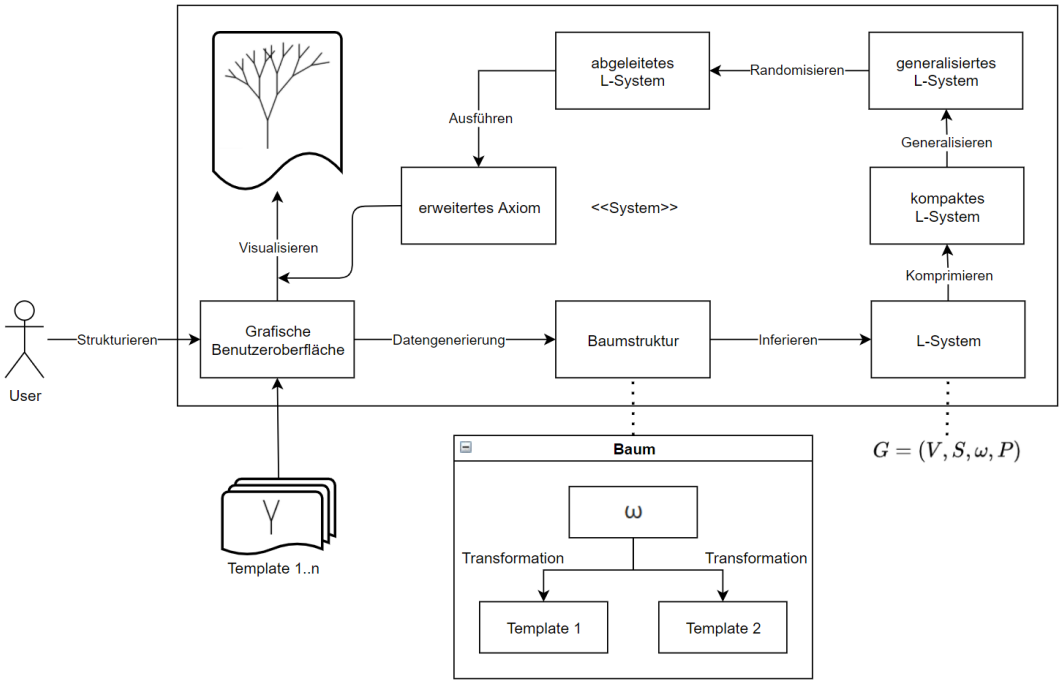

Abbildung 1.1: Architektur des Systems mit einigen Datenstrukturen

Die Darstellung zeigt eine grobe Übersicht der Anwendungsfälle innerhalb des Systems. Der Benutzer strukturiert vom System importierte Templates zu einer Basisstruktur mitteln grafischer Bedienelemente. Die Template-Instanzen werden hierbei in einer Baumtopologie organisiert. Anschließend wird ein L-System aus der Baumstruktur generiert und komprimiert. Bevor ranomisierte, geometrische Parameter auf das kompakte L-System

angewendet werden, wird dieses generalisiert. Zuletzt wird das Axiom des abgeleiteten L-Systems durch die Ausführung erweitert und sichtbar gemacht.

2 Grundlagen

Die Modellierung mithilfe von Grafiksoftware ist eine vergleichbar händische, langwierige Erstellung von Objekten. Hierbei hat der Designer (= Modellierer) die volle Kontrolle über die Strukturen des Objektes.
Bei der prozeduralen Modellierung werden spezifische Strukturen eines zu erstellenden physikalischen Objektes generalisiert und meist über eine Grammatik und globale Parameter abgebildet. Während bei der klassischen Modellierung die menschliche Intuition und bei der prozeduralen Modellierung eine parametrisierte Grammatik vorausgesetzt wird, arbeitet IPM mit bestehenden Modellen und extrahiert („lernt“) die Strukturen des Objektes, die automatisch in eine formale Grammatik überführt werden können. Die Generierung von prozeduralen Modellen ist ein wichtiges, offenes Problem.

Aktuelle Ansätze sind:

- Segmentierung von geometrischen Objekten in Ähnlichkeitsgruppen, um Muster (*engl. Patterns*) zu erkennen
- Kontrollierte Generierung durch Finden optimaler Prameter und Regeln

Weiter:

- (Inverse) prozeduralle Modellierung ist ein schwieriges Problem der Informatik → Quellen sichen und begründen warum das so schwierig ist ([2])
- Prozedurale Regeln sind schwer zu verstehen und kontrollieren → Zugang, Klarheit schaffen, wie man an solche Fragestellungen herangeht
- Schwierigkeit den Expansionsprozess zielgerichtet zu kontrollieren [20]

2.1 Grundbegriffe

- **Modellierung**: Um einen physikalischen Körper in ein digitales Objekt zu überführen, wird mithilfe von Abstraktion (oder Modellierung) ein mathematisches Modell erstellt, das diesen Körper formal beschreibt. 3D Grafiksoftware, wie Blender [28], wird genutzt um geometrische Körper zu modellieren, texturieren und zu animieren.
- **Prozedurale Modellierung**: *„It encompasses a wide variety of generative techniques that can (semi-)automatically produce a specific type of content based on a set of input parameters.“*
  „Prozedurale Modellierung beschreibt generative Techniken, die (semi-)automatisch spezifische, digitale Inhalte anhand von deskriptiven Parametern erzeugen“ (*Übersetzt durch den Autor*)
- **Inverse prozedurale Modellierung (IPM)**:*Aliaga et al.* spricht bei der inversen prozeduralen Modellierung von dem Finden einer prozeduralen Repräsentation von Strukturen bestehender Modelle.

2.2 Grundlegende Arbeiten

Grundlagen

- prozedurale Modellierung [22]
- inverse prozedurale Modellierung [2]
- L-Systeme [12]
- parametrisiert L-Systeme [21]
- Logo-Turtle [19]
- inverse Generierung von L-Systemem [10]

„Not much work addresses inverse procedural modeling of branching structures“

Modellieren von

- Bäumen und Landschaften [4]
- Fassaden [1]
- Gebäuden [15]
- Städten [18]

2.2.1 Softwaretechnik

Eine Fallstudie der Universität Karlsruhe [16] untersucht den Einsatz der Softwaretechnik Extreme Programming(XP) im Kontext der Erstellung von Abschlussarbeiten im Universitätsumfeld. Hierzu werden folgende Schlüsselpraktiken untersucht:

- XP als Softwaretechnik zur schrittweisen Annäherung an die Anforderungen eines Systems
- Änderung der Anforderungen an das Systems
- Funktionalitäten (**Features**) werden als Tätigkeiten des Benutzers (**User Stories**) definiert
- Zuerst werden Komponententests (Modultests) geschrieben und anschließend die Features (Test-driven Design)
- Keine seperaten Testing-Phasen
- Keine formalen Reviews oder Inspektionen
- Regelmäßige Integration von Änderungen
- Gemeinsame Implementierung (Pair Programming) in Zweiergruppen

Aus der Fallstudie geht hervor, dass Extreme Programming einige Vorteile bei der Bearbeitung eines Softwareprojektes einer Bachelorarbeit bietet. Zum einen können sich Anforderungen an das zu erstellende System durch parallele Literaturrecherche ändern, zum anderen können Arbeitspakete durch Releases abgebildet werden.

L-Systeme

Lindenmayer [12] führt eine mathematische Beschreibung zum Wachstum fadenförmiger Organismen ein. Sie zeigt, wie sich der Status von Zellen infolge ein oder mehrerer Einflüsse verhält. L-Systeme sind Ersetzungssysteme, die zur formalen Beschreibung von Zellteilung eingeführt werden, und atomare Teile mithilfe von Produktionsregeln ersetzen. Weiter werden Symbole zur formalen Beschreibung von Verzweigungen, die von Filamenten abgehen, genutzt. Die bekanntesten L-Systeme sind zeichenketten-basiert und werden von *Noam Chomsky* eingeführt. Sie ersetzen parallel Buchstaben eines Wortes, die von einer Grammatik über eine Sprache akzeptiert werden. L-Systeme können unter anderem parametrisiert oder nicht-parametrisiert und kontextfrei oder kontextsensitiv sein.

Formalismen

Ein L-System ist ein Tupel und hat folgende Form:

$$\mathcal{L} = \langle M, \omega, R \rangle, \text{ mit}$$

- M als Alphabet, das alle Symbole enthält, die in der Grammatik vorkommen können,
- ω als Axiom oder „Startwort“ und
- R als Menge aller Produktionsregeln, die für $\mathcal{L}$ gelten

Das Alphabet eines parametrischen Systems enthält Module (Symbole mit Parametern) anstatt Symbole:

$$M = \{A(P), B(P), \dots\} \text{ mit}$$

- $P = p_1, p_2 \dots$ als Modulparameter

Zeichen des Alphabets, die Ziel einer Produktionsregel sind, heißen Variablen. Alle anderen Zeichen aus M sind die Konstanten.

Das Axiom ω ist eine nicht-leere Sequenz an Modulen aus M^+ mit

- M^+ als Menge aller möglichen Zeichenketten aus Modulen aus M

Produktionsregeln sind geordnete Paare aus Wörtern über dem Alphabet, die bestimmte Ersetzungsregeln umsetzen. Hierbei werden Symbole aus einem Wort, die einer rechten Seite (*engl. right hand side (RHS)*) einer Produktionsregel entsprechen, durch die linke Seite des Paares (*engl. left hand side (LHS)*) ersetzt. Sie sind folgendermaßen aufgabaut:

$$A(P) \rightarrow x, x \in M^*$$

M^* ist die Menge aller möglichen Zeichenketten von M inklusive der leeren Zeichenkette ε. Ist die RHS jeder Produktionsregel ein einzelnes Symbol und gibt es zu jeder Variablen eine Regel, spricht man von einem kontextfreien, andernfalls von einem kontextsensitiven L-System.

Logo-Turtle-Algorithmus

Der Logo-Turtle-Algorithmus [19] beschreibt ein Vorgehen zur graphischen Beschreibung von L-Systemen, bei dem jeder Buchstabe in einem Wort einer bestimmten Zeichenoperation zugewiesen wird. So kann aus einem L-System ein grafisches Muster generiert werden, das mit einer Abfolge von Zeichenbefehlen an eine „Schildkröte“ gezeichnet wird. Das Triplett (x, y, θ) definiert den Status (**State**) der Schildkröte. Dieser setzt sich aus der aktuellen Position $\binom{x}{y}$ und dem aktuellen Rotationswinkel θ, der die Blickrichtung bestimmt, zusammen.

Der Algorithmus kann als Komprimierung eines geometrischen Musters gesehen werden. Folgende Symbole mit zugehörigen Steuerungsbefehlen und Statusveränderung sind definiert:

Symbol	Steuerung	Statusveränderung
$F(d)$	Gehe vom derzeitigen Punkt p_1 d Einheiten in die Blickrichtung zu dem Punkt p_2. Zeichne ein Liniensegment zwischen p_1 und p_2	ja, neue Position p_2
$+(\alpha)$	Setzt neuen Rotationswinkel $\theta = \theta + \alpha$	ja, neuer Rotationswinkel θ
$-(\alpha)$	Setzt neuen Rotationswinkel $\theta = \theta - \alpha$	ja, neuer Rotationswinkel θ
[	Lege den aktuellen State auf einen Stack	nein
]	Hole den State vom Stack und überschreibe den aktuellen mit diesem	nein

Alles zwischen den Symbolen [und] wird als Verzweigung interpretiert.

Bsp. $FF[FF]F$ mit Verzweigung $[FF]$

2.3 Verwandte Arbeiten

Bei *Inverse Procedural Modeling of Branching Structures by Inferring L-Systems*[9] geht es um ein Modell zum Lernen von L-Systemen von Verzweigungsstrukturen mithilfe maschinellen Lernens (*Deep Learning*) anhand beliebiger Grafiken. Hierzu werden atomare Strukturen mit einem neuronalen Netz erkannt, eine hierarchische Topologie (Baumstruktur) aufgebaut, aus der ein L-System inferiert und mit einem **Greedy** Algorithmus optimiert wird. Ausgabe des Systems ist ein generalisiertes L-System, aus dem ähnliche Strukturen, wie die der Inputgrafik, erstellt werden können.
Aus dieser Quelle werden folgende Konzepte genutzt:

- Nutzer einer Baumstruktur zur Organisation von genutzten atomaren Verzweigungsstrukturen (**Templates**) mit Knoten für Templates und Kanten für geometrische Transformationen
- Untersuchen der Baumstruktur auf Wiederholungen
- Parametrisierte L-Systeme (L-System mit **Modulen**) zur Abbildung von Transformationsparametern
- Kostenfunktion zur Bewertung eines L-Systems

Inverse prozedurale Modellierung anhand von Hausfassaden [13], 2D-Anordnungen [5, 24], biologischen Bäumen [23] und urbanen Strukturen [17]

Polynomiale Algorithmen zum Inferrieren von L-Systemen [14]

Framework zum Inferrieren von L-Systemen aus Vektordaten [24]

Generalisieren einen Regelsets aus einem Inputset mit Markov Chain Monte Carlo [26, 25]

Lernen von Regeln zum Layout für Gebäude [13]

Inverse prozedurale Modellierung von Fassaden [29]

3 Konzepte

...

3.1 Probleme & Lösungsansätze

...

3.2 Softwarearchitektur

Die Gliederung der Inhalte für die Softwarearchitektur erfolgt nach der arc42-Vorlage [3]

3.2.1 Einführung und Ziele

Ziel ist die Erstellung eines Programms zur Synthetisierung von Ähnlichkeitsabbildungen einer vom Benutzer erstellten Verzweigungsstruktur mittels Inferieren und Optimieren von L-Systemen. Die wesentlichen Features des Programms sind:

- Erstellung einer Verzweigungsstruktur über eine grafische Benutzeroberfläche (**GUI**)
- Einbindung atomarer Strukturen über externe Dateien
- Synthetisierung von ähnlichen Verzweigungsstrukturen anhand der erstellten Struktur
- Anzeigen von Verzweigungsstrukturen

Priorisierte (absteigend) Qualitätsziele, die bei der Erstellung des Systems umgesetzt werden sollten:

- **Funktionalität** durch Umsetzung aller Teilsysteme

- **Interoperabilität** durch Nutzen einer allgemeinen Repräsentation von L-Systemen, damit diese auch in anderen Programmen oder Algorithmen verwendet werden kann
- **Erweiterbarkeit** durch offene Entwurfsmuster (Design Pattern)
- **Modulare** Implementierung für effiziente Wartung und Erweiterung
- **Effizienz** durch effiziente Programmierung
- **Attraktivität** durch intuitive Benutzung (Benutzerfreundlichkeit)
- **Plattformunabhängigkeit** durch Verwenden des Java-Frameworks

3.2.2 Kontextabgrenzung

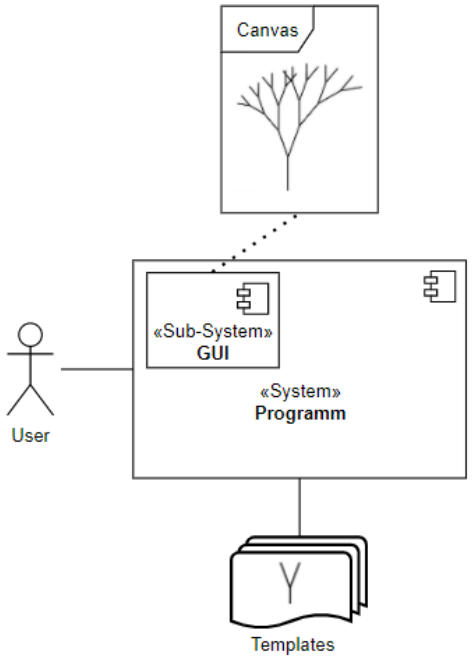

Abbildung 3.1: System und Systemumgebung

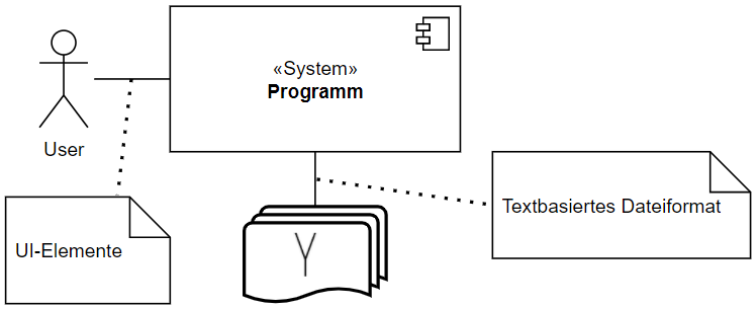

Abbildung 3.2: Technische Interaktion zwischen System und Systemumgebung

3.2.3 Lösungsstrategie

Gewählte Architekturansätze zur Erreichung der Qualitätsziele:

Qualitätsziel	**Architekturansatz**
Funktionalität	• Grafische Benutzerschnittstelle: *GUI* • Generieren der Baumstruktur: *TreeGenerator* • Ableiten von L-Systemen aus Baumstrukturen: *Inferer* • Generalisieren von L-Systemen: *Generalizer* • Randomisieren von L-System Parametern: *Randomizer*
Interoperabilität	Durch das Nutzen allgemeingültiger mathematischer Beschreibungen sollen erstellte L-Systeme in Fremdsystemen, wie Online Visualisierer, genutzt werden können
Erweiterbarkeit	Das Nutzen des **Pipeline Design Pattern**s soll das Erweitern des Systems durch Hinzufügen weiterer Teilschritte (**Pipes**) erleichtern. Trennung der grafischen Oberfläche und der Logik durch Aufbauen des Szenengraphen über ein XML-Dateiformat
Modularität	Sowohl eine sinnvolle Aufteilung von Funktionalitäten auf Dateien und Pakete (**Package**s), als auch effiziente Datenkapselung und geschlossene Informationskontexte sorgen für Modularität des Programms

3.2.4 Bausteinsicht

Ebene 1

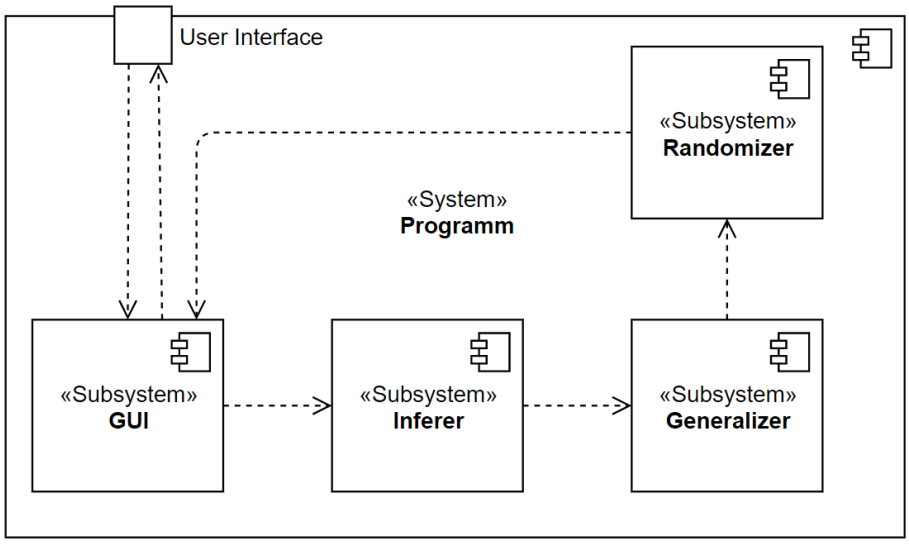

Abbildung 3.3: Subsysteme mit fachlichen Abhängigkeiten

Ebene 2

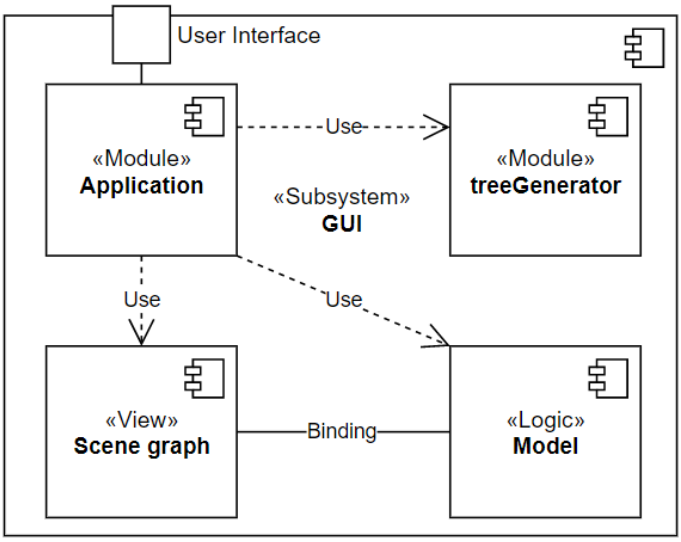

Abbildung 3.4: Subsystem GUI

3.2.5 Laufzeitsicht

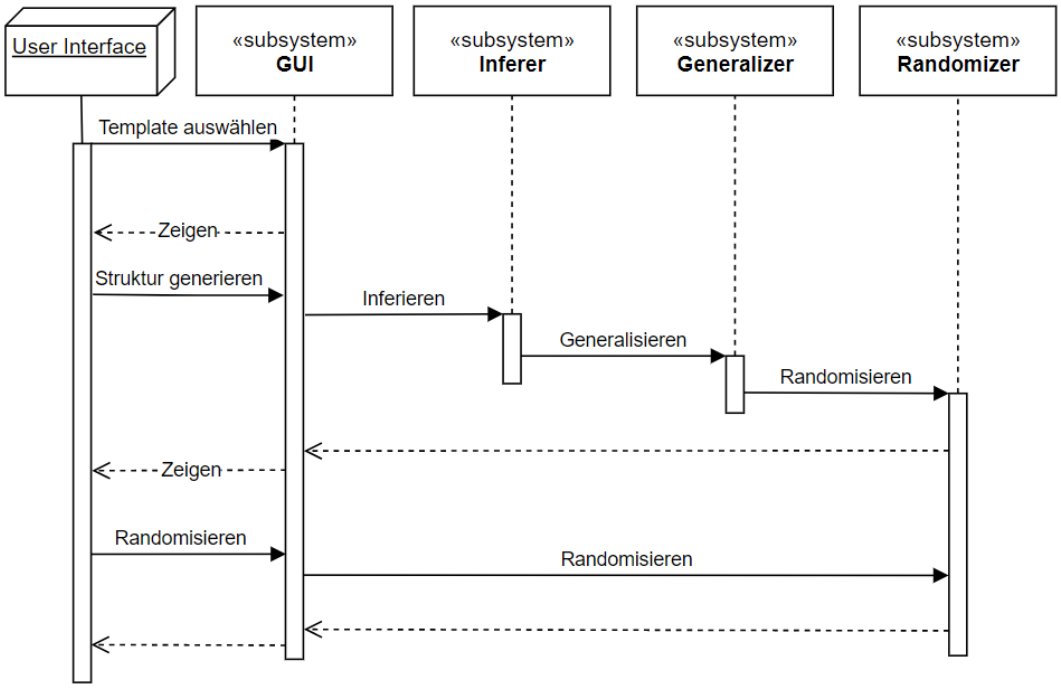

Abbildung 3.5: Laufzeitsicht

3.2.6 Verteilungsicht

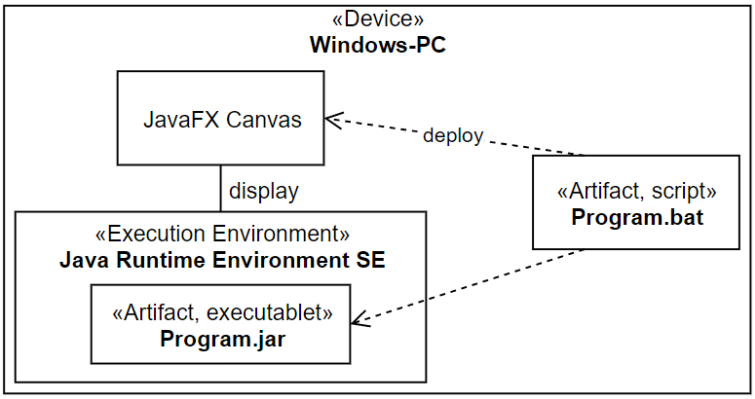

Abbildung 3.6: Infrastruktur Windows-PC

3.2.7 Konzepte

Testbarkeit

Um eine ausreichende Testabdeckung zu erreichen, werden Klassen als kleinstmögliche zu testende Einheit definiert und durch Komponententests geprüft. Der Name eines Tests setzt sich aus dem Präfix als Name der zu testenden Klasse oder Releases und dem Suffix „Test“ zusammen. Testsubjekte werden als **Blackbox** behandelt, also anhand der Spezifikation getestet.

Bsp.: Klasse *Inferer* mit Komponententest *InfererTest*

Da jedes Release der Implementierung ein funktionsfähiges System beinhaltet, kann auf Integrationstests verzichtet werden. Weiter wird ein Release anhand von funktionalen und nicht-funktionalen Anforderungen getestet. Anhand dieser Systemtests wird geprüft, ob Gesamtspezifikationen umgesetzt worden sind.

Bsp.: Release 2 mit Systemtest *Release2Test*

Zum Schluss der Implementierung wird ein Akzeptanztest durchgeführt.

Validierung

Der Benutzer des Systems nutzt ein grafische Schnittstelle. Somit kann sichergestellt werden, dass dieser keine ungültigen Eingaben tätigt. Werden Template-Dateien unter einem bestimmten Dateipfad nicht gefunden, wird eine *NotFound*-Exception protokolliert und dem Benutzer eine Nachricht über ein Pop-up mitgeteilt. Eine *IllegalArgumentException* wird verzeichnet und ausgegeben, wenn eingelesene Template-Dateien strukturelle Fehler aufweisen.

Fehlerbehandlung

Zur Fehlersuche und -behandlung biete sich eine Protokollierung über Vorgänge, Fehler und Ausnahmen an. Bestimmte Fehler werden dem Benutzer weitergegeben und grafisch angezeigt. Folgende Subsysteme werden in das Programm integriert:

- Ausnahmebehandlung (Exception Handling) und
- Protokollierung (Logging)

Erwartete Exceptions werden jeweils mit eigenen Klassen abgebildet, die von einer entsprechenden Klasse aus der Exception-Hierarchie abgeleitet sind. Das Logging ist statisch und überall im System zugreifbar, um eine einheitliche Protokollierung zu gewährleisten.

Datenstrukturen

TODO: Klassendiagramm aus IntelliJ generieren

Workflows & Algorithmen

Um als Benutzer des Systems eine Verzweigungsstruktur zu erstellen, wird folgender Arbeitsablauf umgesetzt:

Algorithmus 31: Erstellen einer Verzweigungsstruktur

```
1     Erster Anker ist vorselektiert
2     Wiederhole, bis Struktur fertiggestellt ist:
3     Selektiere ein Template aus der Liste
4     Setzt Parameter
5     Bestätige Auswahl und Parameter
6     Zeichne ausgewähltes Template mit Parameternd
7    Wähle nächsten Anker aus
```

Aus der Verzweigungsstruktur kann nun ein L-System erzeugt werden:

Für $\mathcal{L} = \langle M, \omega, R \rangle$:

Algorithmus 32: Inferieren eines L-Systems aus einer Baumstruktur

```
1   Initialisierung:
2   M = {F, S}
3   ω = S
4   R ← {α: S → A}
5   β = nächster Knoten*
6   M ← γ ∈ {A, B, ..., Z}, mit γ ∉ M
7
8   Schleife:
9   δ = Wort von β
10  ∀{X, Y, Z} ∈ δ:
11  Ersetze mit ζ ∈ {A, B, ..., Z}, mit ζ ∉ M
12  M ← ζ
13  R ← {γ → δ}
14  Wenn es ein Symbol η in M \ {F, S} gibt mit {η → bel.} ∉ R:
15  γ = η
16  Sonst:
17  Breche Schleife ab
18  β = nächster Knoten*
```

* nach Breitensuche, beginnend bei Wurzelknoten S

Um ein kompakteres, gewichtetes L-System zu erzeugen, werden sich wiederholende Unterrbäume gesucht und ersetzt. Die Gewichtung wird angewendet, um das erzeugte L-System in ein System mit kleiner Regelmenge ($w_l = 1$) oder mit großer Regelmenge ($w_l = 0$) umzuwandeln:

Algorithmus 33: Erstellen eines kompakten L-Systems mit Gewichtung w_l

```
1  Initialisierung:
2  L+ ← Ls*
3  L = ∅
4  Setze Gewichtungsparameter wl ∈ [0,1]
5  Finde maximalen Unterbaum T' aus T** mit Wiederholungen n
6
7  Reduzierung:
8  if n > 1
9  Ersetze alle Vorkommen von T' mit dem selben Symbol γ ∈ {A,B,...,Z}
10 R ← {γ → Ls} mit Ls aus T', R aus L
11 if Ci(L) ≥ Ci(L+)
12 break
13 T ← T'
14 L+ ← L
15 Finde maximalen Unterbaum T' aus T mit Wiederholungen n
```

* L_s als Zeichenkette des Ausgeführten L-Systems $\mathcal{L} = \langle M, \omega, R \rangle$ ** T als Baumstruktur des L-Systems

Die Kostenfunktion stellt die Anzahl Symbole aller *RHS* der Produktionsregeln mit der Menge an Anwendungen der *LHS* gegenüber:

Algorithmus 34: Kostenfunktion C_i mit Gewichtung w_l

$$C_i(\mathcal{L}) = \sum_{A(P) \to M^* \in \mathcal{L}} w_l * |M^*| + (1 - w_l) * N(A(P) \to M^*) ***$$

*** $N(\cdot)$ als Zählfunktion für die Anzahl Wiederholungen einer *LHS* einer Regel in einem ausgeführten L-System

Da das kompakte L-System eine Repräsentation der vom Benutzer erzeugten Verzweigungsstruktur darstellt, werden nun ähnliche Regeln miteinander verbunden und mit einer Wahrscheinlichkeit versehen, um nicht-deterministische Regeln hinzuzufügen.

Algorithmus 35: Längenfunktion L für Grammatiken

$$L(\mathcal{L}) = |M| + \sum_{A(P) \to M^* \in \mathcal{L}} |M^*|$$

Algorithmus 36: Kostenfunktion C_g mit Gewichtung w_0

1 $C_g(\mathcal{L}^*, \mathcal{L}^+) = w_0 * (L(\mathcal{L}^*) - L(\mathcal{L}^+)) + (1 - w_0) + D_g(\mathcal{L}^+, \mathcal{L}^*)$

Algorithmus 37: Generalisieren eines L-Systems mit Gewichtung w_0

```
1  Initialisierung:
2  Regelpaar p* = ∅
3  L* = L+
4  C_g^old = C_g(L* + {p*}, L*)
5
6  Schleife:
7  do
8  Finde Regelpaar p* mit minimalen Kosten C_g(L* + {p_i}, L*), ∀p_i ∈ P*
9  if C_g(L* + {p*}, L*) ≥ 0
10 break
11 c* = C_g(L* + {p*}, L*) − C_g^old
12 C_g^old = C_g(L* + {p*}, L*)
13 L* = L* + {p*}
14 while c* ≤ 0
```

* $\mathcal{P}$ als Menge aller möglichen Regelpaaren aus $\mathcal{L}^*$

4 Implementierung

4.1 Pakete?

Die Implementierung des Programms setzt sich aus folgenden Teilschritten zusammen:

- Erstellung der *GUI* (Paket gui, tree) mit
  - UI-Elementen
  - Render-Canvas
  - Dateianbindung der Templates
  - Erstellung der repräsentativen Baumstruktur
    (*treeGenerator*, Paket tree)
- Implementierung der Subsysteme als Pipes
  - *Inferer* (Paket grammar): Ableiten eines kompakten L-Systems aus einer Baumstruktur
  - *Generalizer* (Paket grammar): Generieren eines generalisierten L-Systems anhand eines „kleinen“ L-Systems
  - *Randomizer* (Paket grammar): Erzeugung von L-Systemen, die der erstellen Verzweigungsstruktur „ähnlich“ sind
- Komponenten- und Systemtests

Subsystem	Umsetzung
GUI	JavaFX als Framework zur Erstellung von grafischen und interaktiven Inhalten. Erstellung der Baumstruktur über dynamisches Erzeugen von Konten während der Strukturierung der Verzweigungsstruktur
Inferer	Algorithmus zum Iterieren maximaler Sub-Bäume und deren Reduzierung mittels Ersetzung durch Symbole und der zugehörigen Produktionsregel, bis eine Kostengrenze, die durch eine Kostenfunktion abgebildet werden kann, erreicht ist
Generalizer	Algorithmus zum Erweitern eines L-Systems um nicht-deterministischer Regeln und Erkennen rekursiver Strukturen
Randomizer	ddd sfevdhbnreaydtydbfsdxc

4.2 Entscheidungen

Mutable or Immutable Objects?

Risiken

Qualitätsmerkmale

Alternativen

Aufwand der Implementierung

4.3 Technologien

- Programmiersprache: Java Version 11 mit

  - JavaFX Version 15 (openjfx)

- Build-Management-Tool: Gradle[8] Version 6.7
- Versionskontrolle: Github Repository[7] via Git[6]
- IDE: JetBrains IntelliJ IDEA[11] 2020.2.2 (Ultimate Edition)
- Betriebssystem: Microsoft Windows 10 Pro 64 Bit
- User Story Map: Trello Board[27]

4.4 Hardware

- Prozessor: Intel Core i5-3570K CPU @ 3.40GHz

5 Evaluierung

...

5.1 Testumgebung

...

5.2 Beobachtungen & Ergebnisse

...

5.3 Diskussion und Bewertung

...

6 Ausblick

...

Literaturverzeichnis

[1] Alhalawani, Sawsan ; Yang, Yong-Liang ; Liu, Han ; Mitra, Niloy: Interactive Facades Analysis and Synthesis of Semi-Regular Facades. In: *Computer Graphics Forum* 32 (2013), 05

[2] Aliaga, Daniel G. ; Demir, undefinedlke ; Benes, Bedrich ; Wand, Michael: Inverse Procedural Modeling of 3D Models for Virtual Worlds. In: *ACM SIGGRAPH 2016 Courses*. New York, NY, USA : Association for Computing Machinery, 2016 (SIGGRAPH '16). – URL https://doi.org/10.1145/2897826.2927323. – ISBN 9781450342896

[3] Hruschka, Dr. P. ; Starke, Dr. G.: *arc42 Softwarearchitektur-Template*. https://arc42.de/template/. – Pragmatisches Muster für die Erstellung, Dokumentation und Kommunikation von Software- und Systemarchitekturen

[4] Deussen, Oliver ; Lintermann, Bernd: *Digital Design of Nature: Computer Generated Plants and Organics*. 1st. Springer Publishing Company, Incorporated, 2010. – ISBN 3642073638

[5] Ellis, Kevin ; Ritchie, Daniel ; Solar-Lezama, Armando ; Tenenbaum, Joshua B.: *Learning to Infer Graphics Programs from Hand-Drawn Images*. 2018

[6] *Git - the stupid content tracker*. https://git-scm.com/. – Git is a member of Software Freedom Conservancy

[7] *Github*. https://github.com/adrian-helberg/bachelor. – GitHub, Inc. (2020)

[8] *Gradle Build Tool*. https://gradle.org/. – Gradle Inc. 2020

[9] Guo, Jianwei ; Jiang, Haiyong ; Benes, Bedrich ; Deussen, Oliver ; Zhang, Xiaopeng ; Lischinski, Dani ; Huang, Hui: Inverse Procedural Modeling of Branching Structures by Inferring L-Systems. In: *ACM Trans. Graph.* 39 (2020), Juni, Nr. 5. – URL https://doi.org/10.1145/3394105. – ISSN 0730-0301

[10] Higuera, Colin de la: *Grammatical Inference: Learning Automata and Grammars*. USA : Cambridge University Press, 2010. – ISBN 0521763169

[11] *Jetbrains IntelliJ IDEA*. https://www.jetbrains.com/idea/. – Capable and Ergonomic IDE for JVM

[12] Lindenmayer, A: Mathematical models for cellular interactions in development. I. Filaments with one-sided inputs. In: *Journal of theoretical biology* 18 (1968), March, Nr. 3, S. 280—299. – URL https://doi.org/10.1016/0022-5193(68)90079-9. – ISSN 0022-5193

[13] Martinovic, Andelo ; Van Gool, Luc: Bayesian Grammar Learning for Inverse Procedural Modeling, 06 2013, S. 201–208

[14] McQuillan, Ian ; Bernard, Jason ; Prusinkiewicz, Przemyslaw: *Algorithms for Inferring Context-Sensitive L-Systems*. S. 117–130, 01 2018. – ISBN 978-3-319-92434-2

[15] Müller, Pascal ; Wonka, Peter ; Haegler, Simon ; Ulmer, Andreas ; Van Gool, Luc: Procedural Modeling of Buildings. In: *ACM Trans. Graph.* 25 (2006), 07, S. 614–623

[16] Muller, M. M. ; Tichy, W. F.: Case study: extreme programming in a university environment. In: *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, 2001, S. 537–544

[17] Nishida, Gen ; Garcia-Dorado, Ignacio ; Aliaga, Daniel G. ; Benes, Bedrich ; Bousseau, Adrien: Interactive Sketching of Urban Procedural Models. In: *ACM Trans. Graph.* 35 (2016), Juli, Nr. 4. – URL https://doi.org/10.1145/2897824.2925951. – ISSN 0730-0301

[18] Parish, Yoav ; Müller, Pascal: Procedural Modeling of Cities, 08 2001, S. 301–308

[19] Prusinkiewicz, P: Graphical Applications of L-Systems. In: *Proceedings on Graphics Interface '86/Vision Interface '86*. CAN : Canadian Information Processing Society, 1986, S. 247–253

[20] Prusinkiewicz, P. ; Lindenmayer, Aristid: *The Algorithmic Beauty of Plants*. Berlin, Heidelberg : Springer-Verlag, 1990. – ISBN 0387972978

[21] Prusinkiewicz, Przemyslaw ; Hammel, Mark ; Mjolsness, Eric: Animation of plant development, 09 1993, S. 351–360

[22] Smelik, Ruben M. ; Tutenel, Tim ; Bidarra, Rafael ; Benes, Bedrich: A Survey on Procedural Modelling for Virtual Worlds. In: *Comput. Graph. Forum* 33 (2014), September, Nr. 6, S. 31–50. – URL https://doi.org/10.1111/cgf.12276. – ISSN 0167-7055

[23] Stava, O. ; Pirk, S. ; Kratt, J. ; Chen, B. ; Mundefinedch, R. ; Deussen, O. ; Benes, B.: Inverse Procedural Modelling of Trees. In: *Comput. Graph. Forum* 33 (2014), September, Nr. 6, S. 118–131. – URL https://doi.org/10.1111/cgf.12282. – ISSN 0167-7055

[24] Stava, Ondrej ; Benes, Bedrich ; Mech, Radomir ; Aliaga, Daniel ; Kristof, Peter: Inverse Procedural Modeling by Automatic Generation of L-systems. In: *Computer Graphics Forum* 29 (2010), 05, S. 1467–8659

[25] Talton, Jerry ; Lou, Yu ; Duke, Jared ; Lesser, Steve ; Mech, Radomir ; Koltun, Vladlen: Metropolis Procedural Modeling. In: *ACM Transactions on Graphics* 30 (2011), 04

[26] Talton, Jerry ; Yang, Lingfeng ; Kumar, Ranjitha ; Lim, Maxine ; Goodman, Noah ; Mech, Radomir: Learning design patterns with Bayesian grammar induction. In: *UIST'12 - Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (2012), 10. ISBN 978-1-4503-1580-7

[27] *Trello Board*. https://trello.com/. – Trello Board, 2020 Atlassian

[28] Vazquez, Pablo ; Siddi, Francesco: *About Blender*. https://www.blender.org/about/. 2017. – Blender Foundation (2002)

[29] Xiao, Jianxiong ; Fang, Tian ; Zhao, Peng ; Ofek, Eyal ; Quan, Long: Image-based Façade Modeling. In: *ACM Transactions on Graphics* 27 (2008), 12, S. 161

A Anhang

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Glossar

HAW Hamburg Die HAW Hamburg ist die formalige Fachhochschule am Berliner Tor.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Template-basierte Synthese von Verzweigungsstrukturen mittels L-Systemen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original