Dokumentation

Kleines Projekt Simulation und technische Diagnose

Implementierung eines einfachen Lindenmayer-Systems

vorgelegt von

Sebastian Seidel (394185)

Aniket Rodrigues (386061)

Laurids von Emden (357005)

Leiter des Fachgebiets: Prof. Dr.-Ing. Clemens Gühmann

Betreuer: M.Sc. Daniel Thomanek

WiSe 2021/2022

Abgabedatum: 14.02.2022

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Institut für Energie- und Automatisierungstechnik
Fachgebiet Elektronische Mess- und Diagnosetechnik

Inhalt

Inhalt ii

1 Pflichtenheft 1

1.1 Projektziel 1

1.2 Anforderungen 1
1.2.1 Muss-Anforderungen 1
1.2.2 Qualitätsanforderungen 1
1.2.3 Optionale Anforderungen 2

2 Projektplanung 2

2.1 Arbeitspakete 2
2.1.1 Projektplanung 1: Erstellung Pflichtenheft 2
2.1.2 Projektplanung 2: Zeitplanung und Management 2
2.1.3 Simulation von Pflanzenwuchs: Informationssammlung, Zusammenstellung, Dokumentation der Ergebnisse 3
2.1.4 Arbeit mit der Software 3
2.1.5 Test der erstellten Software 3
2.1.6 Dokumentation der erstellten Software 4

2.2 Arbeitsbelastung 4

2.3 Meilensteine 5

2.4 Zeitplan 5

2.5 Rekapitulation Projektplanung 6

3 Analyse und Designdokumentation 6

3.1 Entscheidungen zur verwendeten Programmiersprache 6
3.1.1 Kriterium Lizenz 8
3.1.2 Kriterium Kompilierbar- und Ausführbarkeit 9
3.1.3 Kriterium Übersichtlichkeit Quellcode 11
3.1.4 Kriterium Performance 12
3.1.5 Kriterium struktureller Aufbau vom Programm 12
3.1.6 Entscheidung 13

4 Wartungshandbuch 14

4.1 Wie funktionieren Lindenmayer-Systeme? 14
4.1.1 Allgemein 14
4.1.2 Erzeugung von L-Systemen 14
4.1.3 Beispiel 15
4.1.4 Zeichenketten grafisch darstellen (Turtle-Grafik) 15
4.1.5 Verzweigte Lindenmayer-Systeme 17

4.2 Struktur der Software 18

5 Benutzerhandbuch 18

5.1 Voraussetzung 18

5.2 Layout und Funktion des Programms 19

6 Testdokumentation 22

6.1 Parametereingabe der GUI 22

6.2 Features der GUI 25

6.3 Anwendungsbeispiele 25

7 Fazit und Ausblick 27

8 Quellen 29

Anhang A a

Anhang B b

1 Pflichtenheft

1.1 Projektziel

Ziel dieses Projekts ist es, sich mit einer Methode auseinanderzusetzen, die die Möglichkeit der Prädiktion des Pflanzenwachstums entlang von Bahnstrecken dient. Ziel ist es, eine möglichst präzise Voraussage tätigen zu können, wann die Bepflanzung das Lichtraumprofil entlang von Bahnstrecken (geometrische Ausprägung des Regellichtraums nach EN 15273) erstmalig verletzen wird. Dazu soll inertial untersucht werden, ob sich sogenannte Lindenmayer-Systeme eignen, um das Pflanzenwachstum hinreichend zu simulieren. Darüber hinaus soll ein einfaches Lindenmayer-System implementiert, oder eine bereits vorhandene Implementierung recherchiert und angepasst werden.

Insbesondere geht es in dieser inertialen Phase des Projekts um die Recherche zu Lindenmayer-Systemen und bereits vorhandenen Implementierungen. Weiterhin soll die eigene oder die bereits vorhandene Implementierung an die besonderen Anforderungen des Bahnverkehrs angepasst werden. Der Pflanzenwuchs soll visuell dargestellt werden können. Ziel ist, dass das fertige System Bilder der simulierten Bepflanzung in verschiedenen Wachstumsstadien oder das gesamte Wachstum als kontinuierliche Simulation ausgibt.

1.2 Anforderungen

1.2.1 Muss-Anforderungen

- Dokumentation der Rechercheergebnisse zur Simulation von Pflanzenwuchs und zu Lindenmayer-Systemen
- Dokumentation der Rechercheergebnisse zu vorhandenen Implementierungen und Bewertung dieser hinsichtlich deren Eignung zum Einsatz für den vorliegenden Fall
- Arbeit mit der Software: Anpassen einer ausgewählten Implementierung oder Aufbau einer eigenen Software. Ziel ist die visuelle Darstellung von Pflanzenwuchs in verschiedenen Wachstumsstadien oder das gesamte Wachstum als kontinuierliche Simulation ausgeben zu können.
- Dokumentation des Evolutionsprozesses der Software, Wartungshandbuch und Benutzerhandbuch für die abgegebene Software erstellen

1.2.2 Qualitätsanforderungen

- Realistische Gestaltung des Projektplans
- Angemessene Anpassung des Projektplans während der Durchführung des Projekts
- Zeitgerechte und zuverlässige Realisierung der Kernziele des Projekts
- Gut strukturierte und modularisierte Software
- Gut erweiterbare Software
- Gute und verständliche Dokumentation der Projektarbeit

1.2.3 Optionale Anforderungen

- Untersuchen des Verhaltens der Software bei räumlicher Begrenzung des Pflanzenwuchses:
- Untersuchung des Verhaltens der Interaktion des Algorithmus mit anderen Systemen (bspw. Barriere oder Entfernen von Pflanzenstrukturen)

Ferner:

- Berücksichtigung der Norm EN 15273, die den Regellichtraum der Eisenbahn-Bau- und Betriebsordnung (EBO) regelt
- Voraussagemöglichkeit, wann die Bepflanzung die geometrischen Grenzen dieses Regellichtraums erstmalig verletzten wird
- Grafische Umsetzung von Regellichtraum und der geometrischen Grenzen dessen
- Zeitplanung von Pflanzenschnitt und Prognostizierung von Maßen für die zurückzuschneidende Biomasse

2 Projektplanung

2.1 Arbeitspakete

Die Projektplanung mittels Arbeitspaketen wird gewählt, da somit von Projektbeginn an Verantwortlichkeiten verteilt werden können und von jedem Gruppenmitglied langfristig Zeit eingeplant werden kann. Es verbessert die Übersichtlichkeit und sorgt für eine klare Aufgabenverteilung.

2.1.1 Projektplanung 1: Erstellung Pflichtenheft

Dieses Arbeitspaket umfasst die Dokumentation der Anforderungen an das Produkt aus Sicht der Entwickler.

Tabelle 1: Planung Arbeitspaket 1

Aufwand	10h
Personen	Hauptverantwortlich: Laurids von Emden (8) Mitarbeiter: Sebastian Seidel (1), Aniket Rodrigues (1)
Arbeitsmittel	Lastenheft, Vorlage, Literatur
Ergebnisse	Pflichtenheft

2.1.2 Projektplanung 2: Zeitplanung und Management

Dieses Arbeitspaket umfasst Definition der Arbeitspakete, die Aufgabenverteilung, die Erstellung des Zeitplans, die Definition von Meilensteinen und die regelmäßige Reflexion des Projektablaufs. Darüber hinaus beinhaltet das Arbeitspaket Gruppenmeetings zum Austausch über den Projektfortschritt und allgemein administrative Aufgaben (Mailverkehr, Cloud-Betreuung, etc.).

Tabelle 2: Planung Arbeitspaket 2

Aufwand	45h
Personen	Hauptverantwortlich: Laurids von Emden (25) Mitarbeiter: Sebastian Seidel (10), Aniket Rodrigues (10)
Arbeitsmittel	Pflichtenheft, Literatur
Ergebnisse	Projektplan

2.1.3 Simulation von Pflanzenwuchs: Informationssammlung, Zusammenstellung, Dokumentation der Ergebnisse

Dieses Arbeitspaket umfasst die Recherche zur Simulation von Pflanzenwuchs allgemein und speziell zu Lindenmayer-Systemen. Darüber hinaus soll zu bereits bestehender Software zur Simulation von Pflanzenwuchs recherchiert werden. Weiterhin umfasst es die Dokumentation der Rechercheergebnisse und eine Bewertung der gefundenen, vorhandenen Implementierungen hinsichtlich deren Eignung zum Einsatz für den vorliegenden Fall. Überlegungen und Entscheidungen sind zu dokumentieren.

Tabelle 3: Planung Arbeitspaket 3

Aufwand	90h
Personen	Hauptverantwortlich: Sebastian Seidel (30), Aniket Rodrigues (30), Laurids von Emden (30)
Arbeitsmittel	Literatur, Internet
Ergebnisse	Dokumentation der Rechercheergebnisse, Entscheidungen beruhend auf Rechercheergebnissen

2.1.4 Arbeit mit der Software

Dieses Arbeitspaket umfasst die Anpassung der eigenen oder der bereits vorhandenen Implementierung. Ziel ist die visuelle Darstellung von Pflanzenwuchs in verschiedenen Wachstumsstadien oder das gesamte Wachstum als kontinuierliche Simulation ausgeben zu können.

Tabelle 4: Planung Arbeitspaket 4

Aufwand	60h
Personen	Hauptverantwortlich: Sebastian Seidel (25), Aniket Rodrigues (25) Mitarbeiter: Laurids von Emden (10)
Arbeitsmittel	Dokumentation der Rechercheergebnisse, getroffene Entscheidungen
Ergebnisse	Software: Lindenmayer-System zur Prädiktion von Pflanzenwachstum

2.1.5 Test der erstellten Software

Dieses Arbeitspaket umfasst den Test der Software und umfasst eventuelle Anpassungen, die sich durch die Tests ergeben. Ebenfalls umfasst das Arbeitspaket die Dokumentation der durchgeführten Tests.

Tabelle 5: Planung Arbeitspaket 5

Aufwand	25h
Personen	Hauptverantwortlich: Sebastian Seidel (10), Aniket Rodrigues (10) Mitarbeiter: Laurids von Emden (5)
Arbeitsmittel	Lindenmayer-System zur Prädiktion von Pflanzenwachstum
Ergebnisse	Getestetes Lindenmayer-System zur Prädiktion von Pflanzenwachstum

2.1.6 Dokumentation der erstellten Software

Dieses Arbeitspaket umfasst die Dokumentation der erstellten Software. Dazu gehört ein Wartungshandbuch, welches Informationen enthält, die notwendig sind, um die Software zu erweitern oder auftretende Fehler zu finden. Darüber hinaus gehört zur Dokumentation ein Benutzerhandbuch, welches die zur Anwendung der Software notwendigen Schritte beinhaltet und mit dessen Hilfe es möglich sein muss, die Software in Betrieb zu nehmen und benutzen zu können. Zum Inhalt des Benutzerhandbuchs zählen notwendige Eingaben, Masken, Parameter, Schnittstellen und Anwendungsbeispiele.

Tabelle 6: Planung Arbeitspaket 6

Aufwand	28h
Personen	Hauptverantwortlich: Sebastian Seidel (10), Aniket Rodrigues (10) Mitarbeiter: Laurids von Emden (8)
Arbeitsmittel	Getestetes Lindenmayer-System zur Prädiktion von Pflanzenwachstum
Ergebnisse	Dokumentation zum gesteteten Lindenmayer-System zur Prädiktion von Pflanzenwachstum

2.2 Arbeitsbelastung

Tabelle 7: Planung Arbeitsbelastung

	Arbeitsstunden	Puffer
Sebastian Seidel	86	4
Aniket Rodrigues	86	4
Laurids von Emden	86	4

Berücksichtigen wir eine Projektlaufzeit von 13 Wochen (Inhaltlicher Projektstart 8.11.2021, Abschlusspräsentation 7.2.2022) beträgt der wöchentliche Aufwand pro Person:

$$\frac{90Std}{13\,Wochen} = 6h\,55\,min$$

2.3 Meilensteine

Fertigstellung Pflichtenheft und Zeitplanung

Zu diesem Zeitpunkt sind das Pflichtenheft und ein erster Entwurf der Projektplanung fertiggestellt (Arbeitspakete 1 und 2 [in Teilen]). Eine Durchsprache mit dem Betreuer wird angesetzt.

Beendigung Recherche und zugehöriger Dokumentation

Zu diesem Zeitpunkt ist Arbeitspaket 1.3 fertiggestellt. Eine Durchsprache mit dem Betreuer wird angesetzt.

Fertigstellung Software und Softwaretest

Zu diesem Zeitpunkt sind die Arbeitspakete 1.4 und 1.5 fertiggestellt. Eine Durchsprache mit dem Betreuer wird angesetzt.

Präsentation

Zu diesem Zeitpunkt sind alle Arbeiten zu dem Projekt abgeschlossen, die Dokumentation wird an den Betreuer übergeben und die Präsentation gehalten.

2.4 Zeitplan

Im Anhang findet sich ein Ganttchart, welches die Arbeitspakete und den zugehörigen Bearbeitungszeitraum enthält. Der Zeitplan wurde an aktuelle Entwicklungen in der Projektplanung angepasst, zu sehen vor allem an der Anzahl der Meilensteine. Der Zeitplan enthält einen Meilenstein weniger als in der anfänglichen Projektplanung vorgesehen. Die Gründe dafür sind im folgenden Kapitel 2.5 zu finden. Bei der Zeitplanung haben wir besonderes Augenmerk daraufgelegt, dass wir deutlich vor der Klausurenphase die Arbeit an dem Projekt abschließen. Das soll Zeitprobleme mit anderen Kursen dieses Semesters verhindern und uns die Möglichkeit geben, Klausuren am Fachgebiet MDT zu schreiben, für die das Bestehen eines Projekts Voraussetzung ist. Der 7.2. als Präsentationstermin stand für uns von vorneherein fest. Über die Weihnachtsfeiertage haben wir eine realistische Arbeitspause eingelegt, die auch so im Zeitplan vermerkt ist. Die Erstellung des Projektplans läuft parallel mit der Informationssammlung, um zu Beginn des Projekts Zeit zu gewinnen. Ebenso läuft die Dokumentation der Rechercheergebnisse parallel zum Erstellen und Anpassen der Software, da mit dem Wissen aus der Recherche bereits die Software bearbeitet werden kann nebenbei das recherchierte Wissen dokumentiert wird. Das Projektmanagement und damit einhergehend die Reflexion des Projektablaufs laufen über den ganzen Projektzeitraum parallel ab. Dazu gehören regelmäßige Meetings genauso wie das Lesen der Dokumentation anderer Gruppenmitglieder.

Im Anhang A ist der detaillierte Zeitplan in Form eines Gantt-Charts einzusehen.

2.5 Rekapitulation Projektplanung

Die Dokumente aus Kapitel 1 bis 2.4 sind bereits zum ersten Meeting abgegeben worden. Die Zeitplanung und auch die Meilensteine haben sich Verlauf des Projekts jedoch anders entwickelt als im Vorhinein angenommen:

Nach Weihnachten fand vorerst die Arbeit an der Software statt, und parallel bereits die Dokumentation der Ergebnisse wie bspw. das Erstellen von Benutzerhandbuch und Wartungshandbuch. Gravierender Unterschied zum Vorherigen Projektplan ist jedoch, dass wir einen Meilenstein weniger umgesetzt haben als ursprünglich geplant. Der Meilenstein „Fertigstellung Software und Softwaretest“ fiel als Durchsprache mit dem Betreuer weg, da der Zeitraum zwischen möglichem Meeting und Abschlusspräsentation zu kurz gewesen wäre. Wir lernen daraus, dass Meilensteine in Zukunft mit festen Daten eingeplant werden müssen und auch zu Beginn der Bearbeitungszeit des Projekts Termine für Durchsprachen mit allen beteiligten Personen vereinbart werden müssen.

3 Analyse und Designdokumentation

3.1 Entscheidungen zur verwendeten Programmiersprache

In diesem Abschnitt werden Programmiersprache diskutiert, welche im Laufe dieser Arbeit verwendet werden. Zur Auswahl stehen Python, Java, Javascript und Delphi. Python, Java und JavaScript wurden gewählt, da es die drei beliebtesten Programmiersprachen sind nach (Statistisches Bundesamt, 2021) sind. Zusätzlich wird die Programmiersprache Delphi aus persönlichem Interesse gewählt. Der einzige Nachteil von der IDE von Delphi ist, so wie sie im Rahmen dieser Arbeit verwendet wird, dass die IDE nur auf Windows funktioniert, wohin gegen Python, Java und JavaScript Plattformunabhängig sind. Zudem hat JavaScript noch einen weiteren Vorteil, nämlich es wird keine IDE benötigt, jedoch mündet der Vorteil auch in einem Nachteil, denn um JavaScript auszuführen, muss es zunächst über ein HTML Script aufgerufen werden. Dadurch benötigt die Ausführung eines JavaScriptes zwangsläufig einen Internet-Browser.

Tabelle 8: Untersuchte Programmiersprachen

Programmiersprache	**Entwicklungsumgebung**	**Kosten**
Python	Anaconda (Spyder)	Keine
Java	IntelliJ IDEA 2021.2.3 Community	Keine
JavaScript	Jeder Texteditor verwendbar	keine
Delphi	Delphi 10.4 Community	keine

Für die gewählten Programmiersprachen werden jeweils drei verschiedene GitHub Quellen untersucht. Der erste Punkt, ob eine Quelle verwendet wird, ist die Kompilier- und

Ausführbarkeit. Sollte eine Quelle nicht kompilierbar bzw. ausführbar sein, so wird diese Quelle als unbrauchbar eingestuft.

In Tabelle 2 werden alle GitHub Quellen dargestellt, welche im Rahmen dieser Arbeit untersucht werden.

Tabelle 9: GitHub Quellen für den Vergleich

Programmiersprache	Test-ID	Quelle	Lizenz
Python (P)	P01	https://github.com/ambron60/l-system-drawing	MIT License
	P02	https://github.com/namper/L-System	**Keine enthalten**
	P03	https://github.com/mundyreimer/rewriting_systems	GNU GENERAL PUBLIC LICENSE
Java (J)	J01	https://github.com/eobermuhlner/plant-generator	**Keine enthalten**
	J02	https://github.com/RobertoIA/Lindenmayer	**Keine enthalten**
	J03	https://github.com/apinkney97/LSystem	**Keine enthalten**
JavaScript (Js)	Js01	https://github.com/CodingTrain/website/tree/main/CodingChallenges/CC_016_LSystem/P5	MIT License
	Js02	https://github.com/NolanDC/fractals	**Keine enthalten**
	Js03	https://github.com/jobtalle/Lindenmayer3D	MIT License
Delphi (D)	D01	https://github.com/donatbrzoska/L-SystemDrawer	**Kein enthalten**
	D02	https://github.com/NamalD/l-system	**Keine enthalten**

Wie in Tabelle 2 zu sehen, werden für jede Programmiersprache drei Quellen verwendet, mit Ausnahme von Delphi, da es auf GitHub nur zwei Quellen gibt. Außerdem ist zu beachten, dass einige Quelle keine Lizenz angegeben haben und somit das generelle Copy-Right Gesetzt gilt, wodurch diese Software nicht verwendet werden darf (Claudia Fröhling, 2014). Damit die anderen Quellen jedoch verglichen werden können, wird sich vorab für ein System entschieden. Die dazugehörigen Eigenschaften können der Tabelle 3 entnommen werden. Das Ziel dieses Tests ist es mit den Parametern aus Tabelle 3 eine Darstellung zu erzeugen, die so aussieht wie in Abbildung 1.

Tabelle 10: Lindenmayer Regel

Parameter	Wert
Axiom	F
Regel	F -> FF-[-F+F+F]+ [+F-F-F]
Winkel [°]	22,5
Iteration	4

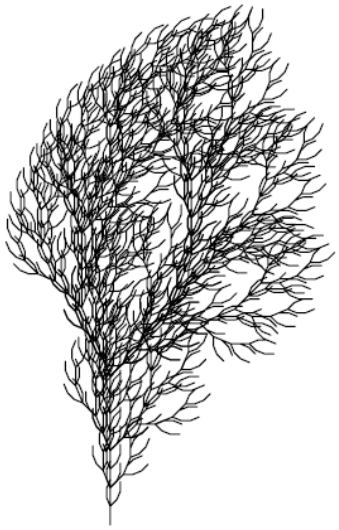


Abbildung 1: Ideal Bild des Lindenmayer-Systems von Tabelle 3, nach (Prusinkiewicz & Lindenmayer, 2004, S. 25)

Die Bewertung der Programmiersprachen, erfolgt zum Teil subjektiv und zum Teil objektiv, wobei es insgesamt vier Kriterien gibt. Das erste Kriterium befasst sich mit der Art der Lizenz, d.h. darf die Quellen verwendet und angepasst werden. Als Zweites wird die Übersichtlichkeit und damit die spätere Anpassbarkeit des Quellcodes bewertet. Im nächsten Bewertungsschritt wird die Performance betrachtet, also die benötigte Zeit, die das Programm für ein vollständiges gezeichnetes Lindenmayer-System benötigt. Zum Schluss wird der strukturelle Aufbau des Programmes betrachtet, d.h. wie ist die Ordnerstruktur und gibt es viele Dateien.

3.1.1 Kriterium Lizenz

Wie bereits oben erwähnt, besitzen einige Quellen keine Lizenz, wodurch diese Quellen, zur weiteren Verwendung nicht benutzt werden können. Dementsprechend beschränkt sich die Arbeit auf die Quellen in Tabelle 4, in denen eine gültige Lizenz vorhanden ist.

Tabelle 11: Quelle mit gültigen Lizenzen

Programmiersprache	Test-ID	Quelle	Lizenz
Python (P)	P01	https://github.com/ambron60/l-system-drawing	MIT License
	P03	https://github.com/mundyreimer/rewriting_systems	GNU GENERAL PUBLIC LICENSE
JavaScript (Js)	Js01	https://github.com/CodingTrain/website/tree/main/CodingChallenges/CC_016_LSystem/P5	MIT License
	Js03	https://github.com/jobtalle/Lindenmayer3D	MIT License

Die Lizenz der Quelle mit der Test-ID Js01 ist einige Ordner weiter oben in der Adresse (https://github.com/CodingTrain/website).

3.1.2 Kriterium Kompilierbar- und Ausführbarkeit

Um die entsprechenden Quellen zu bearbeiten und später anpassen zu können, müssen diese auch kompilier- und anpassbar sein. Sollte eine Quelle nicht kompilierbar sein, wird diese Quelle im weiteren Verlauf der Arbeit nicht weiter beachtet.

Tabelle 12: Kompilierbarkeit und Ausführbarlkeit der Quellen

Programmiersprache	**Test-ID**	Kompilierbar (Ja/Nein)	Ausführbar (Ja/Nein)	Verwendung (Ja/Nein)
Python (P)	P01	Ja	Ja	Ja
	P03	Ja	Ja	Ja
JavaScript (Js)	Js01	Ja	Ja	Ja
	Js03	Ja	Ja	Ja

Wie aus Tabelle 5 ersichtlich, können alle Quellen zur weiteren Bewertung verwendet werden, diese kompilierbar und ausführbar sind. Des Weiteren sind die Ergebnisse der obengenannten Quellen in den nachfolgenden Abbildungen gezeigt.

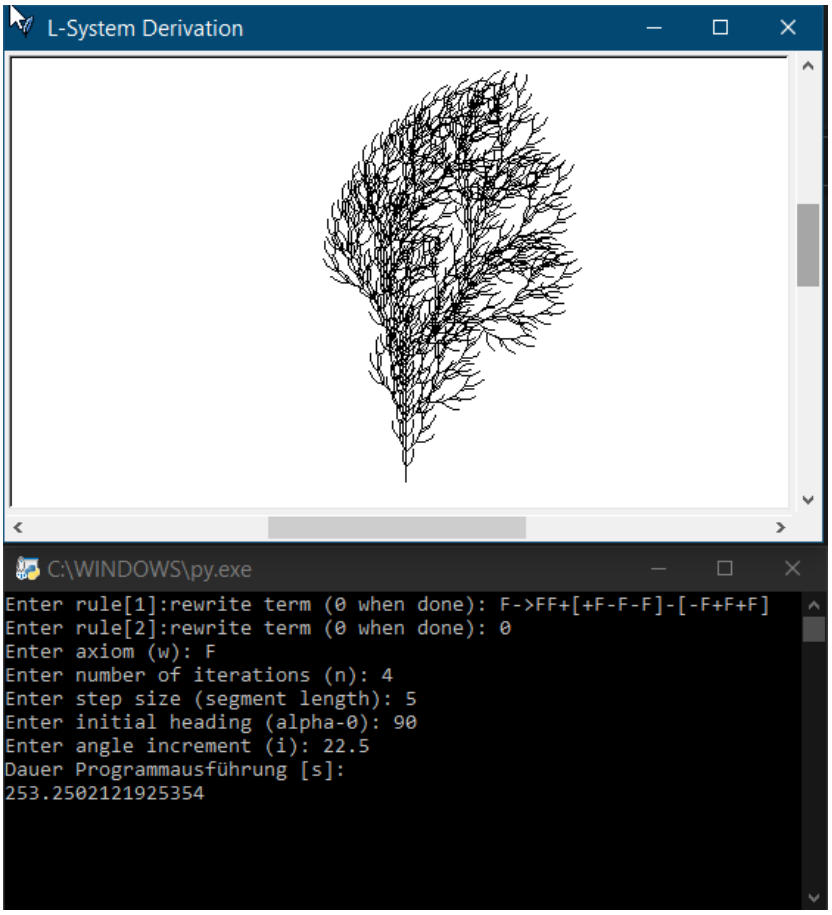


Abbildung 2: Ergebnis Programmende Test-ID P01

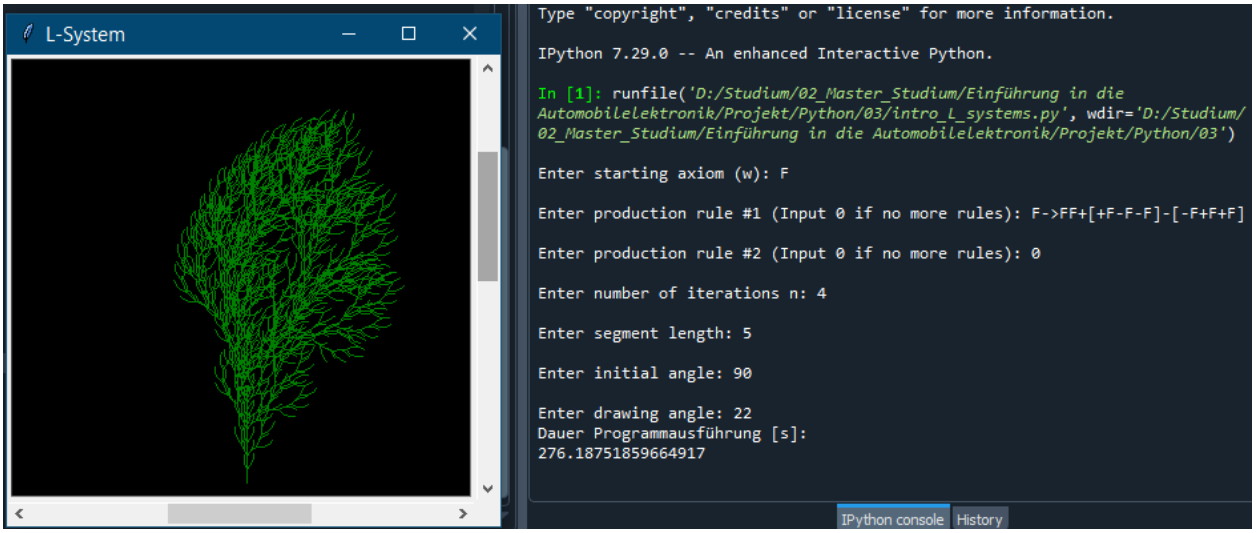


Abbildung 3: Ergebnis Programmende Test-ID P03

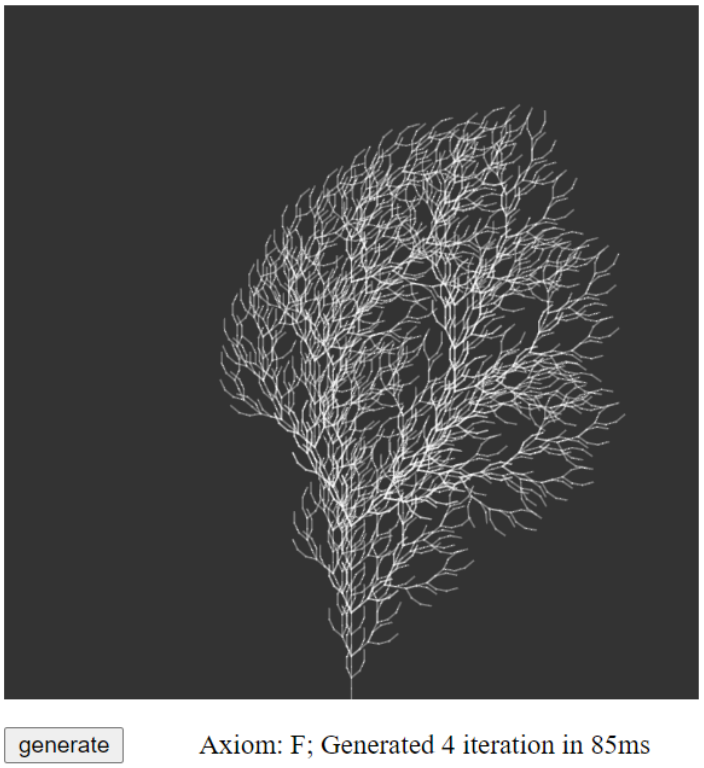


Abbildung 4: Ergebnis Programmende Test-ID Js01

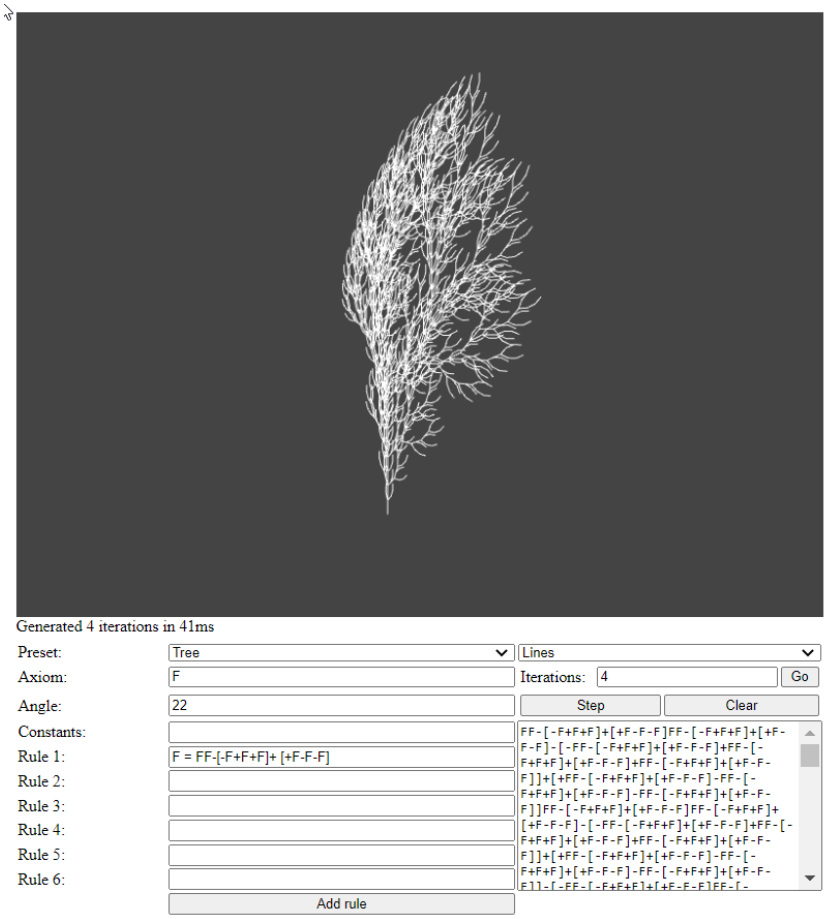


Abbildung 5: Ergebnis Programmende Test-ID Js-03

Auffällig ist in Abbildung 3 und Abbildung 5, dass die Winkel nicht als Datentype float angegeben werden können, sondern nur als integer. Dadurch ist das Ergebnis der Quellen P03 und Js03 nicht exakt mit dem aus Abbildung 1. Wodurch im weiteren Verlauf nur die Quellen mit der Test-ID P01 und Js01 verwendet werden, wie Tabelle 6 zeigt.

Tabelle 13: Vergleich der Ergebnisse der Quelle mit der Referenz aus Abbildung 1

Programmiersprache	**Test-ID**	Ergebnis mit Abb. 1 gleich?
Python (P)	P01	Ja
	P03	Nein
JavaScript (Js)	Js01	Ja
	Js03	Nein

3.1.3 Kriterium Übersichtlichkeit Quellcode

Beide Programmiersprachen haben eine Main-Datei, in der das Lindenmayer-System implementiert ist. Diese Main-Datei hat in beiden Fällen unter 100 Zeilen, wie in Tabelle 7 dargestellt.

Tabelle 14: Übersicht über die Anzahlen an Quellcodezeilen in den Quellen P01 und Js01

Test-ID	Dateiname	Anzahl Quellcodezeilen
P01	Lsystem.py	83
Js01	sketch.js	74

Des Weiteren ist die Lesbarkeit, aufgrund der gewählten Programmiersprachen, Python und JavaScript, sehr hoch, da diese Sprachen mit ihrer Syntax nahe an der englischen Sprache sind. (brainvestment.de, 2021)

3.1.4 Kriterium Performance

Für die Performance wird jede Quelle mit der in Tabelle 3 gezeigten Regel ausgeführt und die dafür benötigte Zeit gemessen. Die Ergebnisse sind in Tabelle 8 dargestellt.

Tabelle 15: Übersicht der Ausführzeiten

Programmiersprache	Test-ID	Dauer [s]
Python (P)	P01	253,250
JavaScript (Js)	Js01	0,085

Quelle: Eigene Darstellung

Der große Zeitunterschied zwischen Python und JavaScript ergibt sich daraus, dass bei Python die Zeichnung mit der turtle-Bibliothek Schrittweise erfolgt, wodurch das Ergebnis anwächst. Im Gegensatz dazu wird bei JavaScript das Ergebnis mit einem Mal gezeichnet.

3.1.5 Kriterium struktureller Aufbau vom Programm

Im folgenden Kriterium wird der Inhalt jedes Ordners betrachtet und beschrieben was dort enthalten ist.

Tabelle 16: Struktureller Aufbau der Test-ID P01

Name	Dateityp e	Beschreibung	löschbar
.idea	Ordner	Einstellungen von IDE IntelliJ	Ja
.vscode	Ordner	Einstellungen von IDE Visual Studio Code	Ja
venv	Ordner	Bibliothek für das ausführen des Programms	Nein
Lsystem.py	Datei	Beinhaltet den Main Quellcode	Nein
README.md	Datei	Beinhaltet kurze Beschreibung und die Lizenz	Nein

Wie in Tabelle 9 zuerkennen, können einige Ordner gelöscht werden, da dort lediglich

Einstellungen der IDE gespeichert sind, welche der Entwickler offenbar verwendet hat. Dagegen besteht die Quelle mit der Test-ID Js01 nur den notwendigsten Bestandteilen, welche in Tabelle 10 dargestellt sind.

Tabelle 17: Struktureller Aufbau der Test-ID Js01

Name	Dateityp e	Beschreibung	löschbar
sketch.js	Datei	Beinhaltet den Main Quellcode	Nein
index.html	Datei	Beinhaltet den Quellcode zum Aufrufen der sketch.js im Browser	Nein

Für die Quelle mit der Test-ID Js01 muss allerding noch die Bibliothek JsP5 in den entsprechenden Ordner eingefügt werden. Diese Bibliothek ist für die Zeichenroutine verantwortlich. Die entsprechende Quelle dieser Bibliothek ist in der index.html erwähnt.

3.1.6 Entscheidung

Abschließend kann gesagt werden, dass im Rahmen dieser Arbeit wird die Programmiersprache Python (Test-ID P01) verwendet wird. Diese Programmiersprache ist im Vergleich zu JavaScript aus Tabelle 8 zwar langsamer, aber dafür in der Forschung und Software-Entwicklung verbreiteter als JavaScript. Wohingegen JavaScript eher dafür geeignet ist Webanwendungen zu programmieren. (brainvestment.de, 2021)

4 Wartungshandbuch

4.1 Wie funktionieren Lindenmayer-Systeme?

Das L-System ist ein elementarer Baustein des zu dieser Dokumentation gehörigen Algorithmus. In diesem Kapitel soll ein leicht verständlicher Überblick über L-Systeme gegeben werden, wie sie aufgebaut sind und wie sie funktionieren.

4.1.1 Allgemein

Bei Lindenmayer- oder L-Systemen handelt es sich um einen mathematischen Formalismus, der der Erzeugung von Fraktalen dient. Aus diesen Fraktalen können Computergrafiken erzeugt und Pflanzen können realitätsnah modelliert werden.

Tatsächlich ist ein Großteil der Natur aus Fraktalen aufgebaut. Auffallend sind Symmetrien von Blättern, der Blick in einen aufgeschnittenen Blumenkohl oder die spiralenartige Anordnung von Schuppen auf Kiefernzapfen. Pflanzen sind sogenannte Multifraktale, sie besitze Fraktale verschiedener Dimensionen (Stamm, Äste, Blätter, Knospen, Früchte, etc.). Fraktale scheinen also eine gute mathematische Grundlage für die Simulation von Pflanzen (und deren Wuchses) zu bieten.

L-Systeme gehören zu den sog. Ersetzungssystemen: Wesentliches Prinzip des L-Systems ist das sukzessive Ersetzen eines einfachen Objekts durch bestimmte vordefinierte Regeln. Diese Ersetzungssysteme werden rekursiv durchgeführt. Ein rekursives System zeichnet sich dadurch aus, dass es an sich ein unendlicher Vorgang ist, der durch sich selbst definiert wird oder sich selbst als Teil enthält. Die Vorgänge in rekursiven L-System sind in der Regel kurz beschrieben und bestehen aus relativ kurzen Anweisungen. Um Rekursion entstehen zu lassen, ruft im L-System eine Funktion sich selbst auf, bis eine im Programm enthaltene Abbruchbedingung greift und damit den unendlichen Vorgang beendet. In unserem Fall ist diese Abbruchbedingung eine endliche Zahl von Iterationen. Fraktale entstehen also durch Wiederholung bzw. Ersetzung in unendlich vielen Durchläufen bzw. Iterationen einer bestimmten Schleife mit vorgegebenen Formeln. Folgen und Reihen bilden das mathematische Gerüst für Fraktale. Es werden die durch die Formeln (den Algorithmus) erzeugten Zeichenketten von einer sog. Turtle-Grafik in eine grafische Darstellung umgewandelt. Ergebnis einer Simulation ist somit immer eine abstrakte Liste oder Zeichenkette und eine grafische Umsetzung dieser Liste. Berechnung und Darstellung können beim Programmieren zwei unabhängige Schritte sein.

4.1.2 Erzeugung von L-Systemen

Um ein L-System zu erzeugen, wird die Anzahl n der Iterationen festgelegt, d.h. es werden Ersetzungsschritte n-mal wiederholt. In jedem Ersetzungsschritt wird der aktuelle Wert w Zeichen für Zeichen abgearbeitet und jedes Zeichen durch das neue, in den Ersetzungsregeln festgelegte Zeichen ersetzt. L-Systeme enthalten Befehle p, die auf jeden Wert w n-mal angewandt werden. Die Ersetzungsbefehle p ordnen dabei einem Zeichen eine andere Zeichenfolge zu. Diese Zuordnung wird immer mit einem Pfeil ausgedrückt: A → B bedeutet, dass A durch B ersetzt wird. Wird für ein Zeichen kein Befehl angegeben, ersetzt sich das Zeichen durch sich selbst (also z.B. A → A).

4.1.3 Beispiel

Ein einfaches Beispiel für ein L-System findet sich auf der Webseite von Michael Szell [Michael Szell] und soll hier wiedergegeben werden:

Ein Beispiel dafür ist die blaugrüne Algenart Anabaena Catenula. Das Wachstum der Alge vollzieht sich in fadenförmigen Zellketten. Jede Zelle wird mit einem Symbol beschrieben. Dabei gibt es in der vereinfachten Ausführung kleine Zellen, die mit A, und große, die mit B bezeichnet werden. Weiters beschreibt ein Suffix l oder r, ob es sich um linke oder rechte Tochterzellen handelt. Somit umfasst das die kleine Zelle der Alge A vier verschiedene Werte *w*:

$A = \{A_r, A_l, B_r, B_l\}$

Folgende Befehle *p* werden festgelegt:

Tabelle 18: Ersetzungsbefehle Algensimulation

$A_l \rightarrow A_lB_r$
$A_r \rightarrow B_lA_r$
$B_l \rightarrow A_lB_r$
$B_r \rightarrow B_lA_r$

Somit lautet das L-System:

$G = (\{A_r, A_l, B_r, B_l\}, A_l, \{A_l \rightarrow A_lB_r, A_r \rightarrow B_lA_r, B_l \rightarrow A_lB_r, B_r \rightarrow B_lA_r\})$.

Führt man das System mit n=5 Iterationsschritten aus, ergibt sich für jedes Element A_r, A_l, B_r, B_l eine eindimensionale Zeichenkette. Jedes Element wird der Reihe nach überprüft und mittels der Befehle *p* wird entschieden, wie es ersetzt wird. Die ersten fünf Zustände des Wertes A_l sehen nach Anwendung der Regeln wie folgt aus:

Tabelle 19: Zeichenkette 5 Iterationen Algensimulation

n	Zeichenkette
1	A_l
2	A_lB_r
3	$A_lB_rB_lA_r$
4	$A_lB_rB_lA_rA_lB_rB_lA_r$
5	$A_lB_rB_lA_rA_lB_rB_lA_rA_lB_rB_lA_rA_lB_rB_lA_r$

4.1.4 Zeichenketten grafisch darstellen (Turtle-Grafik)

Bei einer Turtle-Grafik handelt es sich um eine Bildbeschreibungssprache. Man muss sich vorstellen, dass ein stifttragender Roboter (hier die Turtle) sich auf der Zeichenebene bewegt und mit einfachen Kommandos gesteuert werden kann. Diese Kommandos sind in der Funktion des L-Systems hinterlegt. In unserem Falle sollen Bilder von Pflanzen beschrieben werden.

Folgende vom Nutzer vorgegebenen Kommandos kann die Turtle ausführen:

Tabelle 20: Befehle für die Turtle

F	Bewege dich um eine bestimmte festgelegte Schrittweite l vorwärts und zeichne eine Linie der Anfangs- zur Endposition
f	Bewege dich nach vorne wie bei F, hebe aber den Schwanz (zeichne keine Linie)
+	Drehe dich um einen festgelegten Winkel φ gegen den Uhrzeigersinn
-	Drehe dich um den Winkel φ im Uhrzeigersinn

Die Turtle kann sich nur gradlinig bewegen.

In fast allen Anwendungsfällen ist die Turtle-Grafik von koordinatenbasierter Grafikbeschreibung abgelöst worden. Für die Darstellung von Fraktalen mittels des L-Systems findet sie jedoch im professionellen Bereich noch Anwendung. Die Turtle besitzt keinen Speicher und führt Anweisungen sofort aus.

Die grafische Umsetzung des mathematischen Modells ist sehr wichtig, um Theorien mit Modellen zu bestätigen und reale Strukturen mit dem Modell zu vergleichen.

Verknüpfen wir das Wissen zu Turtle-Grafiken (F,f,+,-) mit den Begriffen der Werte *w* und Befehle *p,* so können wir mittels Lindenmayer-Systeme verschiedenstes zeichnen.

Zur Veranschaulichung stellen wir an dieser Stelle sog. Koch-Konstruktionen vor: Es wird der Wert w, der Ersetzungsbefehl p und der für die Turtle-Grafik notwendige Winkel φ festgelegt. Dieser Winkel wird von der Turtle gegangen wird, wenn ein + oder ein – vorkommen:

Tabelle 21: Beispiel Koch-Konstruktion

Startwert	w	F
Ersetzungsbefehl	p	F → F+F--F+F
Winkel	φ	60°
Anzahl der Iterationen	n	4

Das Ergebnis sieht wie folgt aus:

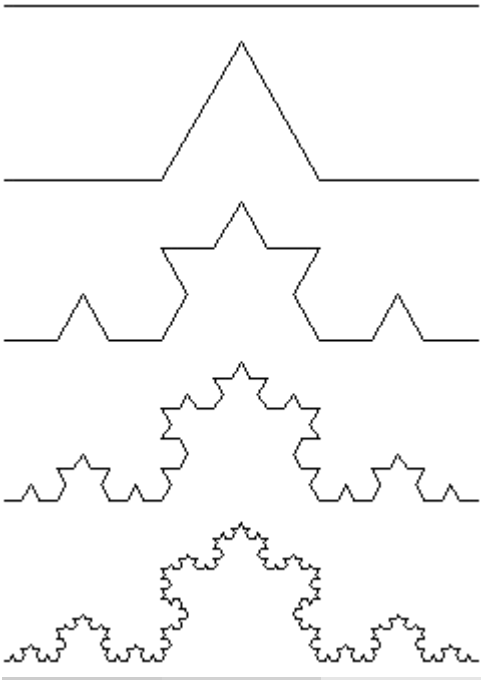


Abbildung 6: Bildbeschreibung: Koch-Kurve (von oben nach unten): Initiator (F), Generator (F+F--F+F) und drei weitere Iteratonsschritte.

4.1.5 Verzweigte Lindenmayer-Systeme

Die bisher vorgestellten Befehle haben für fadenförmige Organismen ausgereicht. Da Pflanzen jedoch üblicherweise Verzweigungen enthalten, müssen neue Turtle-Befehle eingeführt werden, um diese definieren zu können:

Tabelle 22: Turtle-Befehle verzweigte L-Systeme

[	Beginne eine Verzweigung: Merke dir den momentanen Zustand (= Position und Blickrichtung, optional andere Attribute wie Dicke und Farbe)
]	Beende eine Verzweigung: Lade den gespeicherten Zustand und fahre am Verzweigungspunkt mit der Konstruktion fort

Diese Art der L-Systeme bezeichnet man als geklammerte L-Systeme. Hier einige Beispiele für geklammerte L-Systeme:

Abbildung 7: Ergebnisse geklammerte L-Systeme

All diese L-Systeme liefern immer dasselbe Produkt: Ein Startwert wird durch schrittweise angewandte, vordefinierte Befehle immer zum selben Endprodukt verändert. Eine Anhäufung von Pflanzen kann so nur aus einer Kopie von vielen gleichen Pflanzen aufgebaut werden, was von der Realität weit entfernt ist.

Um realitätsnäher zu arbeiten, gibt es sog. **stochastische L-Systeme**:

Bei stochastischen L-Systemen gibt es eine Auswahl von Formeln, Werten und dazugehörigen Wahrscheinlichkeiten, die bestimmen, welcher Befehl wann und wie häufig angewendet wird. Das hat eine Änderung von Topologie, geometrischer Struktur und Dimension der Pflanze zur Folge. Es gibt verschiedene Befehle für einen Ausgangswert, aus denen vor jeder Iteration zufällig ausgewählt wird. So verändert sich die endgültig generierte Darstellung der Pflanze. Als Beispiel soll hier das „stochastische Gras“ (http://michael.szell.net/fba/kapitel2) dienen):

Tabelle 23: Beispiel „stochastisches Gras“

Startwert	w	FF
Ersetzungsbefehl 1	p1	F → F[+F]F[-F]F
Ersetzungsbefehl 2	p2	F → F[+F]F
Ersetzungsbefehl 3	p3	F → F[-F]F
Winkel	φ	25.7°
Anzahl der Iterationen	n	5

Ergebnis:

Abbildung 8: Ergebnis "stochastisches Gras"

4.2 Struktur der Software

Um den Aufbau und die Struktur der abgegebenen Software bestmöglich zu verstehen, ist diese ausführlich kommentiert. Für das Verständnis der Abläufe bei der Eingabe und der vorgenommen Rechenschritte, liegt in Anhang B ein Ablaufdiagramm ab.

5 Benutzerhandbuch

5.1 Voraussetzung

Das Programm verwendet die Bibliotheken „ghostscript“, „pillow“ und „tkinter“. Sollten die Bibliotheken nicht vorhanden, können diese über die Kommandozeile installiert werden. Die Installation wird am Beispiel von Anaconda 3 gezeigt.

Tabelle 24: Benötigte Bibiliotheken

Bibliothek	Kommando
ghostscript	`conda install -c conda-forge ghostscript`
pillow	`conda install -c conda-forge pillow`
tkinter	`conda install -c anaconda tk`

Die hier genannten Befehle können in der Anaconda CMD.exe ausgeführt und installiert werden. Beim Erstellen des Skriptes wurde die Python Version 3.9.7 verwendet. Es wird empfohlen mindestens Python v3.9.7 oder höher zu verwenden.

5.2 Layout und Funktion des Programms

Das Layout des Programms wird anhand der nachfolgenden Abbildung gezeigt.

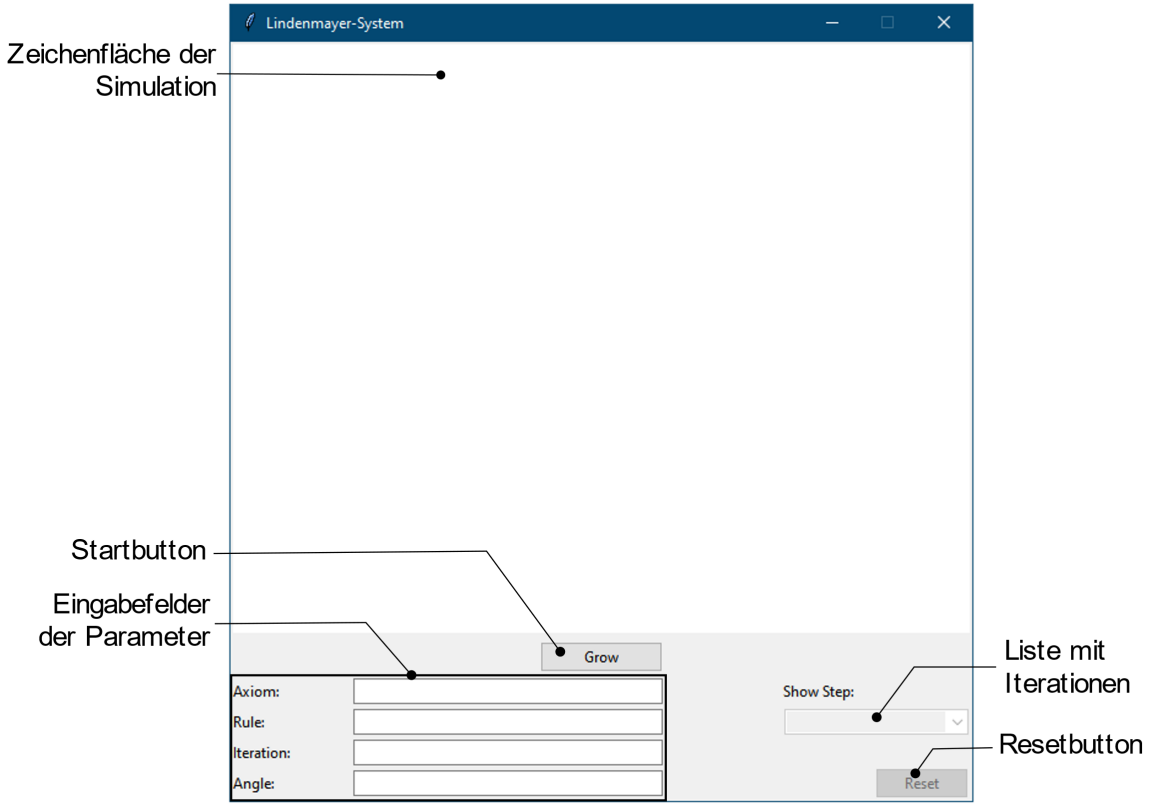


Abbildung 9: Layout des Programms (GUI)

Bevor eine Simulation gestartet werden kann, müssen bestimmte Parameter eingegeben werden. Diese Parameter sind in der nachfolgenden Tabelle erklärt.

Tabelle 25: Parameter für das Pflanzenwachstum

Parameter	Beschreibung	Verwendbare Werte/ Format
Axiom	Gibt den Startwert an	Ein Element von [„F“, „G“, „R“, „L“]
Rule	Beschreibt die Ersetzungsregel	Besteht aus Elementen des Axioms und der Seperator ist ein „=“ Bsp. F=F+F--F+F
Iteration	Gibt die Anzahl der Iterationen an	Nur positive Werte, welche größer sind als 0
Angle	Gibt den Winkel an mit dem sich der gezeichnete Strich bei „+“ und „-“ verdreht	Nur positive Dezimalzahlen mit Punkt (.) anstatt eines Kommas (,), kann größer oder gleich 0 sein

Nachdem die Parameter eingeben wurden, kann die Simulation über den Startbutton („Grow“) gestartet werden. Nach jeder durchgeführten Iteration wird der aktuelle Stand der Zeichenfläche im selben Verzeichnis gespeichert, in dem das Skript liegt. Die gespeicherten Bilder besitzen folgendes Format „Output_LS_yyyy-mm-dd_hh-mm-ss-ms.png“. Außerdem lassen sich die einzelnen Iterationsschritte über eine Liste auswählen. Dies ist allerdings nur möglich, nachdem die Simulation fertig ist.

Um die Zeichenflächen zurückzusetzen kann der Resetbutton verwendet werden. Im Anschluss kann eine neue oder geänderte Simulation gestartet werden.

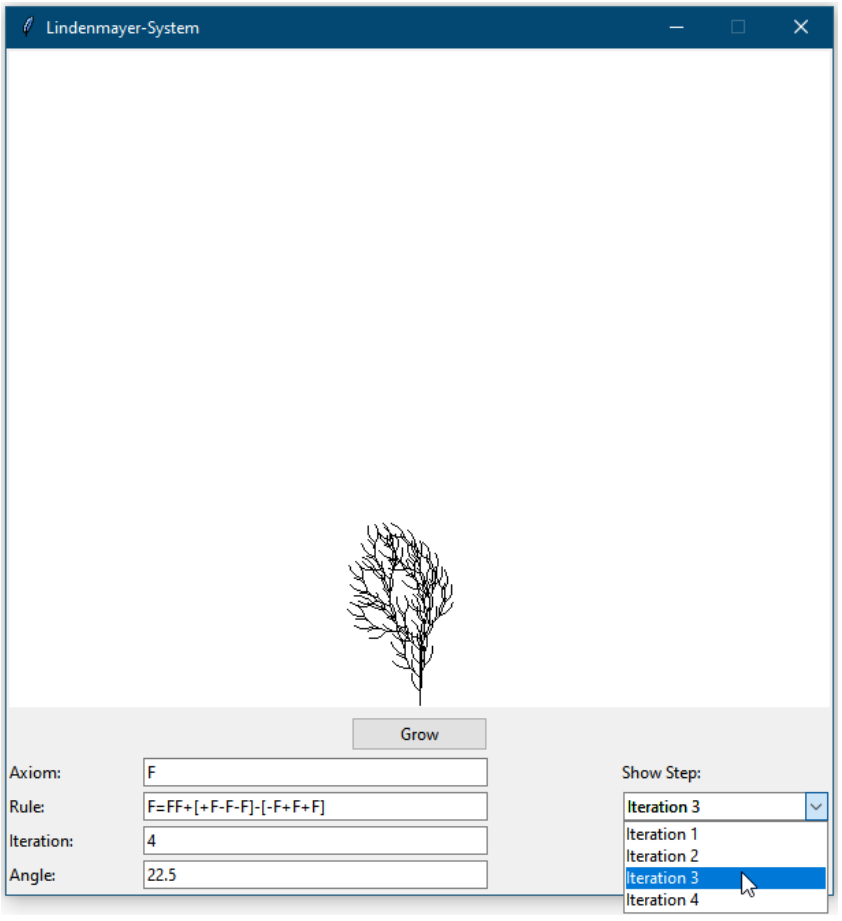


Abbildung 10: Auswählen von abgespeicherten Bildern der zugehörigen Iteration

Das Zurücksetzen der Zeichenfläche beinhaltet auch das Leeren der Liste mit den gespeicherten Iterationen. Die gespeicherten Bilder bleiben aber im Verzeichnis vorhanden.

Die in Anhang B abgelegte Prozessübersicht verdeutlicht die Arbeit des Programmcodes. Darüber hinaus ist der Programmcode ausführlich kommentiert. In der nachfolgenden Testdokumentation sind Anwendungsbeispiele des Programmcodes dargestellt und beschrieben.

6 Testdokumentation

In diesem Abschnitt wird die Testdokumentation des entwickelten L-System-Codes zur Simulation des Pflanzenwachstums vorgestellt. Zunächst wird die Parametereingabe für die L-System-Simulation getestet. Dann werden die entwickelten Features der GUI analysiert und präsentiert. Abschließend werden zwei Pflanzenarten nach (Prusinkiewicz & Lindenmayer, 2004, S. 25) vorgestellt und simuliert.

6.1 Parametereingabe der GUI

Die Parameter des L-System-Algorithmus müssen wie in Tabelle 25 ausführlich beschrieben eingegeben werden. Wenn nicht, erscheint eine Fehlermeldung auf dem Bildschirm.

Erstens, wenn versehentlich ein oder mehrere Parameter nicht in das Eingabefeld eingegeben wurden und die Grow-Button gedrückt wird, erscheint eine Fehlermeldung wie in Abbildung 11 zu sehen.

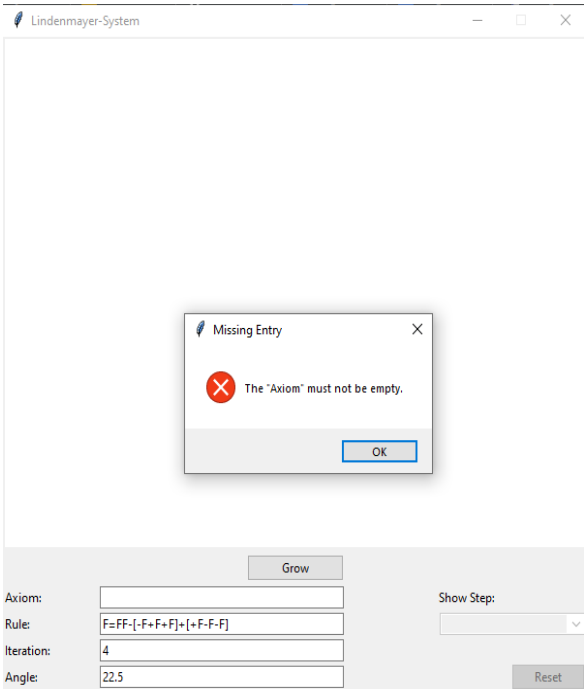


Abbildung 11: Leeres Eingabefeld bei der Parametereingabe

Wird dann die für die Simulation des Pflanzenwachstums zuständige L-System-Regel falsch in das Eingabefeld eingetragen und der Grow-Button gedrückt, erscheint erneut eine Fehlermeldung auf dem Bildschirm (siehe Abb. 12).

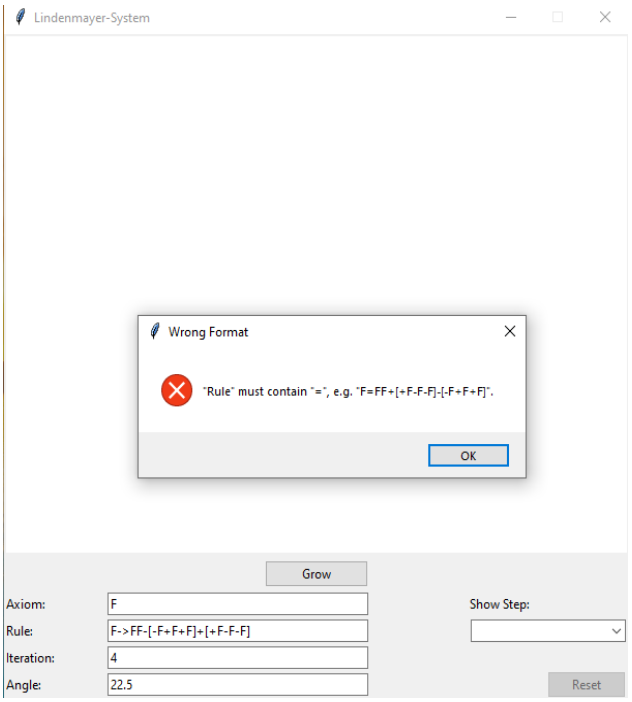


Abbildung 12: Falsches Format bei der L-System Regeleingabe

Wenn schließlich entweder die Anzahl der Iterationen oder der Winkel falsch in das Eingabefeld eingegeben wird (falsches Format oder negative Werte) und der Grow-Button gedrückt wird, erscheint erneut eine Fehlermeldung auf dem Bildschirm (siehe Abb. 13 und Abb.14).

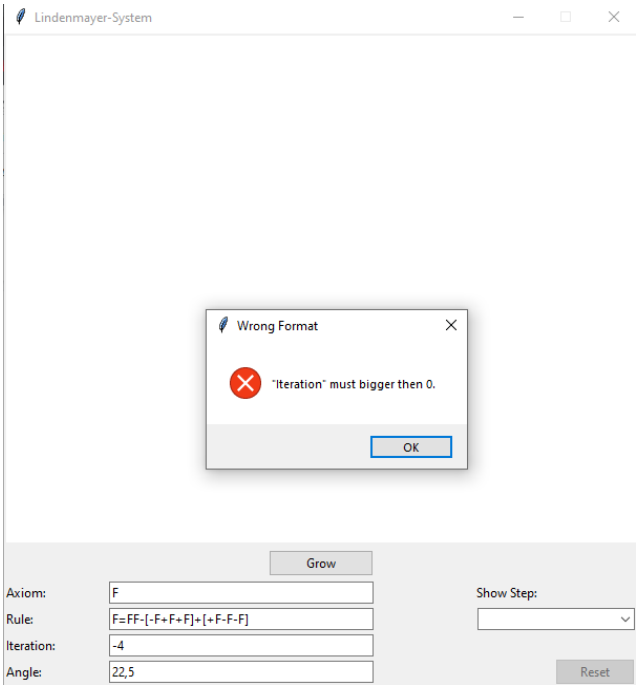


Abbildung 13: Negative Zahl bei der Iterationseingabe

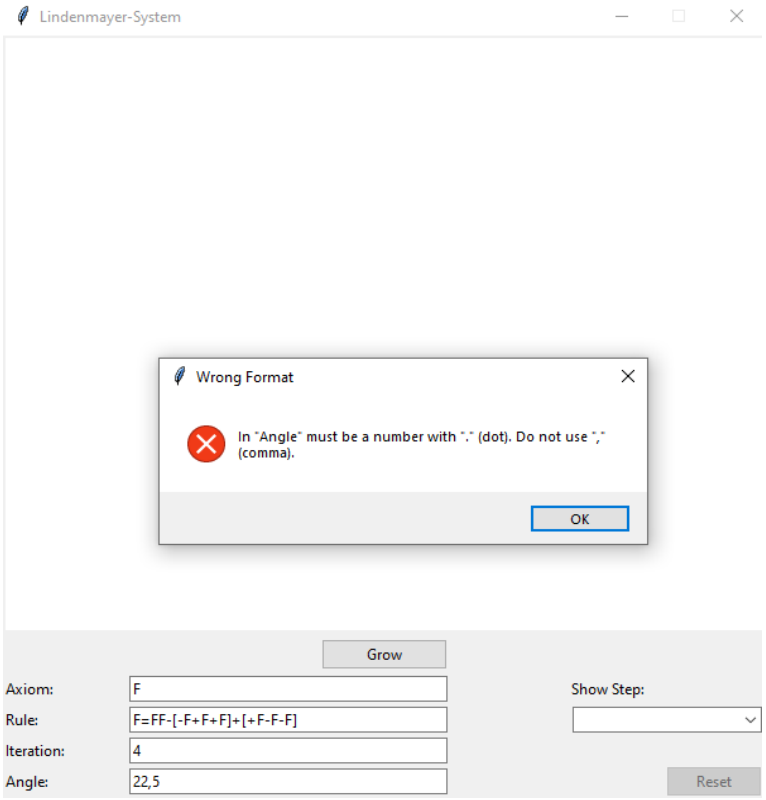


Abbildung 14: Falsches Format bei der Winkeleingabe

Wenn alle Parameter korrekt in das Eingabefeld eingegeben wurden und der Grow-Button gedrückt wird, beginnt das Programm erst dann mit der kontinuierlichen Simulation des Pflanzenwachstums.

6.2 Features der GUI

- **Prozentsatz des simulierten kontinuierlichen Pflanzenwachstums:**

  Dies ist eine wichtige Funktion, die der GUI hinzugefügt wurde. Es zeigt den Prozentsatz des simulierten kontinuierlichen Pflanzenwachstums. Wenn ein großer Iterationswert gewählt wird (<5), kann die kontinuierliche Simulation des Pflanzenwachstums viel Zeit in Anspruch nehmen. Daher bietet das Hinzufügen dieser Funktion einen guten Überblick über das bereits simulierte Pflanzenwachstum. Sie ist im oberen Teil der GUI zu sehen (siehe Abb. 15).

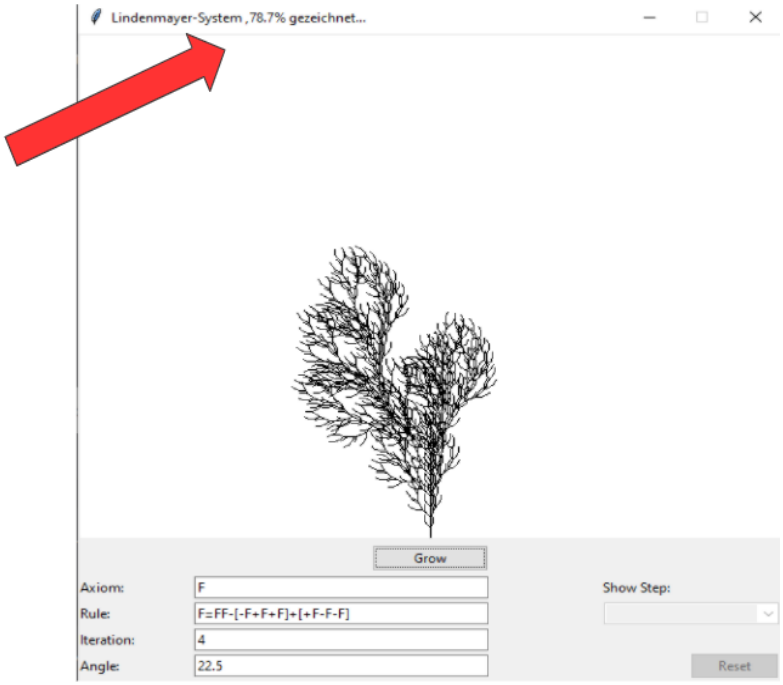


Abbildung 15: Prozentsatz des simulierten kontinuierlichen Pflanzenwachstums

- **Show-Step: Einsehen des Wachstums nach jeweiliger Iteration**

  Sobald die kontinuierliche Simulation des Pflanzenwachstums abgeschlossen ist, kann jede einzelne Iteration des Pflanzenwachstums angezeigt werden, indem auf Show Step geklickt wird und den bevorzugten Iterationswert ausgewählt wird, der angezeigt werden soll (siehe Abb. 10). Dies ist ein wichtiges Feature, weil man das Pflanzenwachstum nach einer bestimmten Iteration visualisieren kann. Dies ist eine entscheidende Funktion im Hinblick auf die Beantwortung des zentralen Forschungsproblems (“wann die Bepflanzung das Lichtraumprofil entlang von Bahnstrecken erstmalig verletzen wird”).

6.3 Anwendungsbeispiele

Schließlich werden mit Hilfe der entworfenen Software zwei Arten von Pflanzen simuliert, um eine visuelle Darstellung von Pflanzenwuchs in verschiedenen Wachstumsstadien zu gewährleisten. Die Parameter für die Simulation stammen aus (Prusinkiewicz & Lindenmayer, 2004, S. 25).

- **Simulation des Pflanzenwachstums (Beispiel 1):**

Tabelle 26: Parameter der 1. Pflanzenart

Parameter	**Wert**
Axiom	F
Rule	F=FF+[+F-F-F]-[-F+F+F]
Iteration	4
Angle	22.5

Die Simulation der ersten Pflanzenart ist in Abbildung 16 zu sehen.

Abbildung 16: Simulation der ersten Pflanzenart

Wie deutlich zu sehen ist, nehmen mit zunehmender Anzahl der Iterationen auch die Höhe und Dichte der Pflanze zu.

- **Simulation des Pflanzenwachstums (Beispiel 2):**

Tabelle 27: Parameter der 2. Pflanzenart

Parameter	**Wert**
Axiom	F
Rule	F=F[+F]F[-F]F
Iteration	4
Angle	25.7

Die Simulation der zweiten Pflanzenart ist in Abbildung 17 zu sehen.

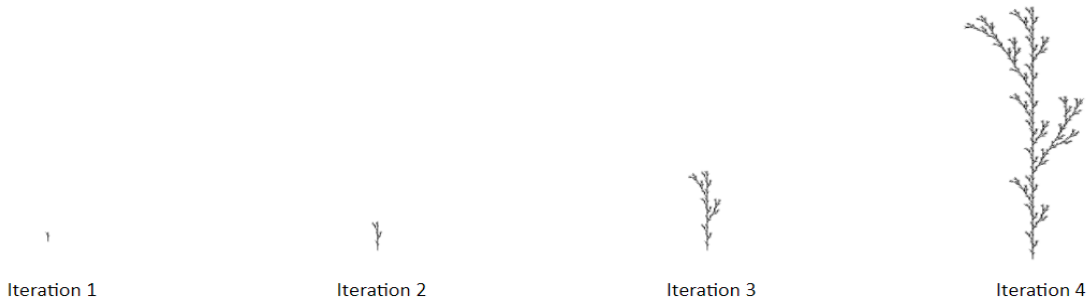


Abbildung 17: Simulation der zweiten Pflanzenart

In Abb. 16 und 17 ist eine visuelle Darstellung des Pflanzenwachstums mit Hilfe des entwickelten L-Systems zu sehen. Somit wurde das Hauptziel dieses Projekts erfolgreich abgeschlossen.

7 Fazit und Ausblick

Ziel dieses Forschungsprojektes war die Implementierung und Simulation des Pflanzenwachstums mit Hilfe eines einfachen L-Systems. Weiterhin sollte das Gesamtwachstum als kontinuierliche Simulation realisiert werden sowie die visuelle Darstellung des Pflanzenwachstums in verschiedenen Wachstumsstadien. Schließlich musste eine GUI für eine einfachere und übersichtlichere Benutzererfahrung erstellt werden. Um diese Ziele zu erreichen, musste das Projekt zunächst erfolgreich geplant werden (siehe 1. Pflichtenheft + 2. Projektplanung). Dann musste allgemein über die Simulation des Pflanzenwachstums recherchiert werden (siehe 3. Wartungshandbuch). Danach sollte den Entwicklungsprozess und Entscheidungen dokumentiert werden (4. Analyse und Designdokumentation). Zum Schluss sollte die Software dokumentiert, beschrieben und getestet werden. Damit sind die Grundlage für weitere Gruppen und weitere Recherche im Bereich der Simulation des Pflanzenwachstums mittels eines einfachen L-Systems gelegt.

Ausblick

In diesem Projekt wurde ein einfaches Lindenmayer-System implementiert, um die Visualisierung des Pflanzenwachstums zu erzielen. Diese Arbeit kann in mehreren Bereichen weiter ausgebaut werden. Wir schlagen fünf grundlegende Bereiche vor, in denen zukünftige Forschung betrieben werden kann.

- *Mehrere Regeleingabe*

  In diesem Projekt wurden Simulationen des Pflanzenwachstums mit Hilfe von nur einer “Regel” erreicht. Um das Pflanzenwachstum von komplizierteren und ausgefeilteren Pflanzen zu simulieren, muss die Software leicht verändert werden, damit der Benutzer mehr als eine Regel bei der Parametereingabe eingeben kann.

- *Mehrere Pflanzenwachstum in einem Bild*

  In diesem Projekt wurde die Simulation des Pflanzenwachstums einer Pflanze in einem Bild mit Hilfe der Turtle-Grafik erreicht. Zukünftige Forschungen können durchgeführt

werden, um die Möglichkeit zu untersuchen, das Wachstum mehrerer Pflanzen in einem Bild zu simulieren.

- *Stochastische L-Systeme*

  In der realen Welt ist die Topologie, geometrische Struktur und Dimension der Pflanze viel komplizierter als das simulierte Pflanzenwachstum mit Hilfe einfacher L-Systeme. Um das Pflanzenwachstum realistischer zu simulieren, können sogenannte stochastische L-Systeme implementiert werden.

- *3D-Pflanzenwachstum*

  Darüber hinaus ist das Pflanzenwachstum in der Natur dreidimensional. Daher kann die aktuelle Software erweitert werden, um 3D-Pflanzenwachstum zu simulieren.

- *Integration Bahnverkehr und Regellichtraum*

  Hauptziel des erweiterten Forschungsvorhabens ist es, das Pflanzenwachstum realistischer zu simulieren, um zu untersuchen, wann die Bepflanzung entlang von Bahnstrecken erstmals das Lichtraumprofil verletzen wird. Daher ist mit dem Abschluss dieses kleinen Projekts und der Implementierung eines einfachen L-Systems zur Simulation des Pflanzenwachstums die Basis für zukünftige Forschung gelegt, um das endgültige Ziel im Bereich der Bahnverkehr zu erreichen.

8 Quellen

brainvestment.de (2021) *JavaScript oder Python: Die beliebtesten Programmiersprachen zum Einstieg - brainvestment.de* [Online]. Verfügbar unter https://brainvestment.de/javascript-oder-python/#2-javascript-oder-python-welche-sprache-ist-anf%C3%A4ngerfreundlicher (Abgerufen am 12 Dezember 2021).

Claudia Fröhling (2014) *Software ohne Lizenz: Der Anfang vom Ende der Open-Source-Ära?* [Online], Frankfurt, Software & Support Media GmbH. Verfügbar unter https://entwickler.de/open-source/software-ohne-lizenz-der-anfang-vom-ende-der-open-source-ara/ (Abgerufen am 10 Dezember 2021).

Prusinkiewicz, P. & Lindenmayer, A. (2004) „The Algorithmic Beauty Of Plants“, S. 1–228 [Online]. Verfügbar unter https://link.springer.com/book/10.1007/978-1-4613-8476-2 (Abgerufen am 26 November 2021).

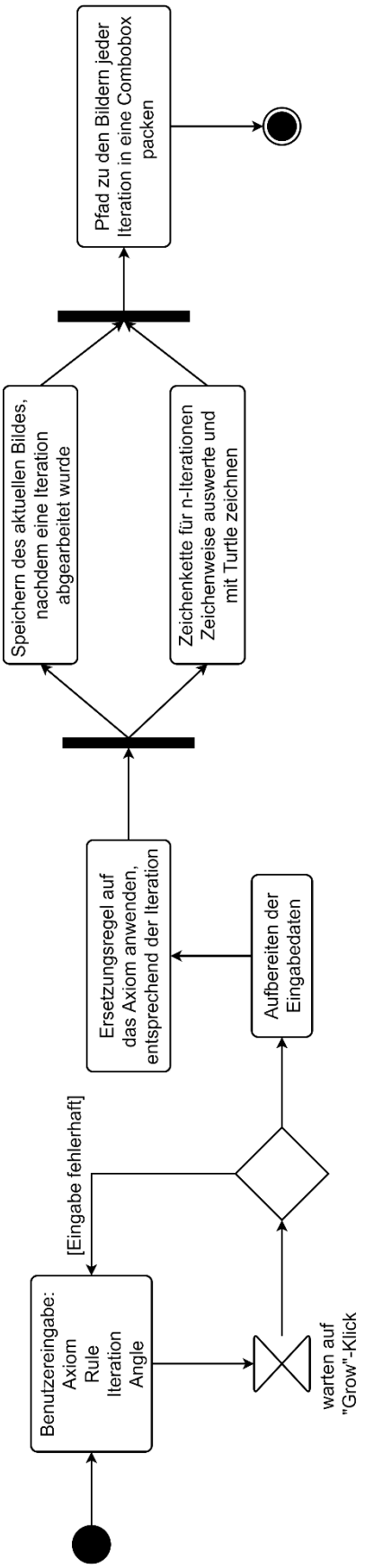
Statistisches Bundesamt (2021) „Die beliebtesten Programmiersprachen wiltweit laut PYPL-Index im November 2021“ [Online]. Verfügbar unter https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/ (Abgerufen am 26 November 2021).

Michael Szell *Rekursive Strukturen – Fraktale Wachstumsmodelle von Pflanzen* *http://michael.szell.net/fba/* [Online] (Aberufen 13.01.2022)

Anhang A

GANTT project
2021
2022
Kick-Off Projekt
Durchsprache Pflichtenheft und Zeitpla...
Durchsprache Recherheergebnisse und ...
Abschlusspr
Woche 44
Woche 45
Woche 46
Woche 47
Woche 48
Woche 49
Woche 50
Woche 51
Woche 52
Woche 1
Woche 2
Woche 3
Woche 4
Woche 5
Woche 6
Vorgang
Anfang
Ende
Kick-Off Projekt 05.11.21 05.11.21
3.1 Erstellung Pflichtenheft 08.11.21 14.11.21
3.2 Erstellen des Projektplans 15.11.21 01.12.21
Durchsprache Pflichtenheft und Zeitplanung 02.12.21 02.12.21
3.3 Informationssammlung 08.11.21 01.12.21
3.3 Zusammenstellung der Ergebnisse 02.12.21 20.12.21
3.3 Dokumentation der Rechercheergebnisse 20.12.21 02.01.22
Durchsprache Recherheergebnisse und Software 07.01.22 07.01.22
3.4 Erstellen/Anpassen der Software 20.12.21 10.01.22
Weihnachtspause 23.12.21 27.12.21
3.5 Test der erstellten Software 03.01.22 21.01.22
3.6 Dokumentation der erstellten Software 17.01.22 10.02.22
Abschlusspräsentation 07.02.22 07.02.22
Reflexion Projektablauf, Management 08.11.21 14.02.22

Anhang B


Benutzereingabe:
Axiom
Rule
Iteration
Angle
warten auf
"Grow"-Klick
[Eingabe fehlerhaft]
Aufbereiten der
Eingabedaten
Ersetzungsregel auf
das Axiom anwenden,
entsprechend der Iteration
Zeichenkette für n-Iterationen
Zeichenweise auswerte und
mit Turtle zeichnen
Speichern des aktuellen Bildes,
nachdem eine Iteration
abgearbeitet wurde
Pfad zu den Bildern jeder
Iteration in eine Combobox
packen