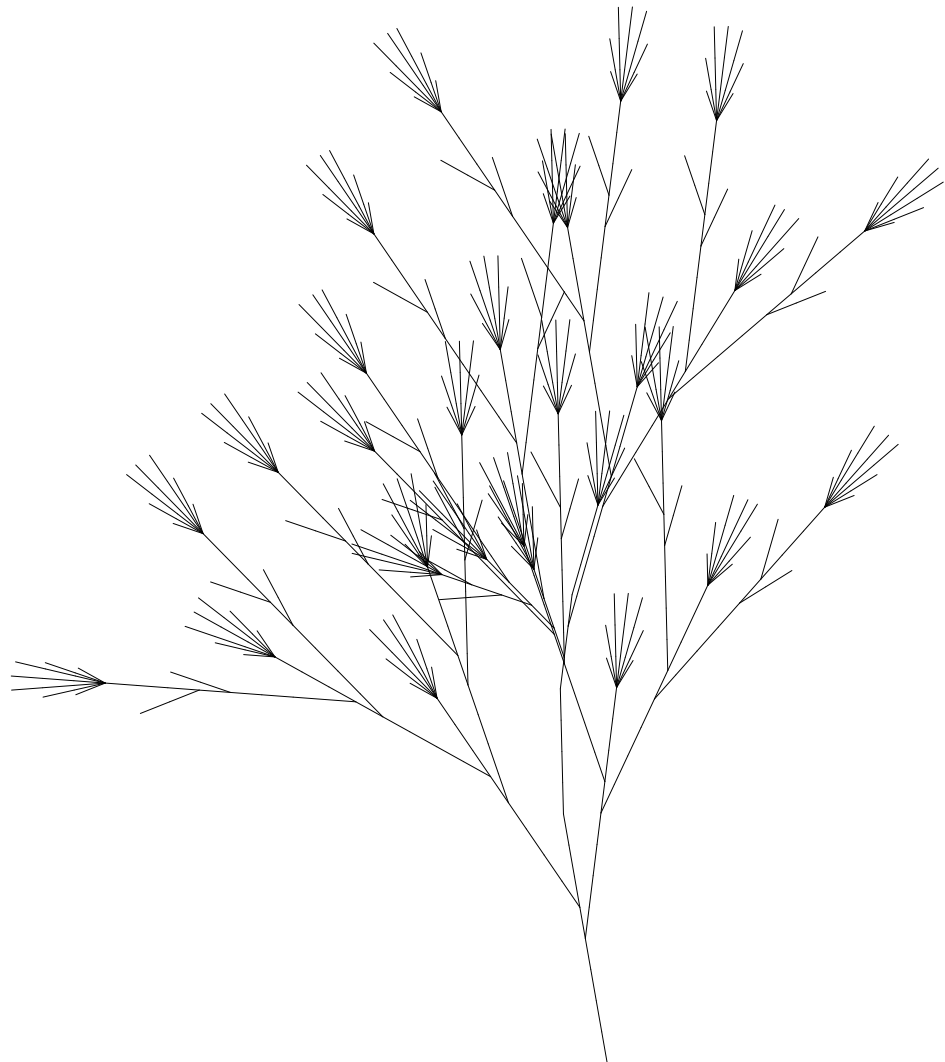

Modellierung und Visualisierung geometrischer Strukturen mit Hilfe von Lindenmayer-Systemen



Maturitätsarbeit

verfasst von

Philipp Stucki

betreut durch

Andi Stöckli, Physiklehrer, KME

eingereicht am 10. November 2006 in Zürich an der

Kantonalen Maturitätsschule für Erwachsene

Copyright ©2006 Philipp Stucki. All rights reserved.

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Inhaltsverzeichnis

1	Einführung	1
2	L-Systeme als Zeichenketten	3
3	Visualisierung	5
3.1	Die Schildkröte	5
3.2	Beispiele	6
4	Die Kochsche Schneeflocke	7
4.1	Das L-System	7
4.2	Die Visualisierung	8
5	L-Systeme mit Speicher	9
5.1	Der Speicher	9
5.2	Befehle für den Speicher	9
5.3	Beispiel 1	9
5.4	Beispiel 2	10
6	Zusammenfassung	12
6.1	Übersicht der Befehle	12
6.1.1	Standardbefehle	12
6.1.2	Speicherbefehle	12
7	Das Programm LSys/JS	13
7.1	Übersicht	13
7.2	Der Parser	13
7.2.1	Syntax	14
7.3	Der Compiler	14
7.4	Der Interpreter	15
7.5	Bedienung	15
8	Fazit	16
	Literaturverzeichnis	17
A	Ausgesuchte L-Systeme	18
A.1	Pflänzchen I	18
A.2	Pflänzchen II	19
A.3	Pflänzchen III	20
A.4	Alge	21
A.5	Spirale	22
A.6	Gosper-Kurve	23
A.7	Sierpinski-Dreieck	24
B	Kopie LSys/JS	25

Abstract

Lindenmayer-Systeme bieten eine einfache Möglichkeit, sehr schöne geometrische Strukturen zu erzeugen, welche in den meisten Fällen fraktale Eigenschaften (Selbstähnlichkeit) aufweisen. Sehr leicht können mit L-Systemen auch pflanzenähnliche Strukturen erzeugt werden.

In dieser Arbeit wird erläutert, wie L-Systeme definiert, generiert und anschliessend visualisiert werden. Die angewendeten Methoden zum Generieren und Visualisieren eines L-Systems werden ausführlich erklärt. Für die Visualisierung der in der Arbeit gezeigten Beispiele wurde ausserdem eigens das Programm LSys/JS entwickelt, welches in einem Kapitel vorgestellt und dokumentiert wird.

Ebenfalls enthält diese Arbeit einen Anhang, in welchem einige ausgesuchte L-Systeme gezeigt werden.

1 Einführung

Ein L-System ist eine Beschreibung einer geometrischen Struktur mit Hilfe einer speziell dafür entworfenen formalen Sprache. Formal heisst dabei, dass die Sprache eine Menge von Wörtern umfasst und dass jedes Wort eine Folge von Zeichen aus der Menge der verfügbaren Zeichen - dem Alphabet - darstellt.

Eine Programmiersprache beispielsweise ist eine formale Sprache, deren Wörter aus dem Alphabet - hier alle verfügbaren Schlüsselwörter und Funktionen, welche die Programmiersprache umfasst - bestehen. Ein Programm, welches in dieser Programmiersprache verfasst ist, stellt eine Menge von Wörtern dar.

Ein L-System besteht aus den folgenden Komponenten:

- Variablen
- Konstanten
- Axiom
- Produktionsregeln

Mathematisch lassen sich diese vier Komponenten in einem Quadrupel zusammenfassen. Im Gegensatz zu einer Menge spielt die Anordnung der einzelnen Komponenten in einem Quadrupel eine Rolle. Die Menge $\{1,2,3,4\}$ ist dieselbe Menge wie $\{4,3,2,1\}$. Das Quadrupel $(1,2,3,4)$ ist aber nicht dasselbe Quadrupel wie $(4,3,2,1)$. In einer Menge ist auch die Anzahl der Elemente nicht festgelegt, während ein Quadrupel immer genau vier Komponenten umfasst. Ein L-System lässt sich also schreiben als das Quadrupel $G = (V, S, \omega, P)$ wobei

- V die Menge der Zeichen ist, welche Variablen sind
- S die Menge der Zeichen ist, welche Konstanten sind
- ω das Axiom des L-Systems ist
- P Die Menge aller Produktionsregeln des L-Systems ist.

Vereinfacht ausgedrückt lässt sich das in etwa wie folgt erklären. V ist die Menge aller Zeichen, welche als Variablen gelten. Variablen können verschiedene Werte annehmen, sind also nicht konstant und können je nach L-System verschieden sein. S ist die Menge aller Zeichen, welche als Konstanten gelten. Die Werte von Konstanten sind unveränderlich und in allen L-Systemen gleich. Alle Befehle zur Steuerung der Schildkröte (siehe Kapitel 3.1) sind beispielsweise Konstanten.

V und S bilden zusammen das Alphabet der formalen Sprache für die Definition eines L-Systems. Alle Wörter der formalen Sprache setzen sich aus Zeichen zusammen, welche in diesen beiden Mengen enthalten sind.

Das Axiom ω legt die Ausgangsbedingung für das L-System fest. Es ist ebenfalls ein Wort der formalen Sprache und besteht somit aus Zeichen der beiden Mengen V und S .

Die Produktionsregeln oder Ersetzungsregeln P bestehen jeweils aus einem Zeichen aus der Menge V und aus Wörtern der formalen Sprache. Das Zeichen wird bei jedem Auftreten durch das zugeordnete Wort ersetzt. Tritt z.B. das Zeichen A auf und ist für dieses Zeichen die Regel $A \rightarrow B$ definiert, so wird das Zeichen A durch das Wort B ersetzt.

Das ist auch das eigentliche Prinzip eines L-Systems. Bei jedem Auftreten eines Zeichens, für welches eine entsprechende Regel definiert ist, wird das Zeichen durch das Wort ersetzt, welches die Regel definiert. Dieser Vorgang wird wiederholt durchgeführt, wodurch bei jedem neuen Durchgang - auch Iteration genannt - jedes Zeichen, für welches eine Regel ein Wort definiert, durch das jeweilige Wort ersetzt wird. Die Anzahl Iterationen bestimmt dabei die Komplexität des L-Systems.

Durch den Ersetzungsvorgang können sehr leicht fraktale Strukturen¹ entstehen, weil ein Zeichen einer Regel jeweils wieder im Wort vorkommen kann, durch welches es ersetzt wird. Die Beispiele in Kapitel 2 demonstrieren dieses Prinzip ausführlich.

¹Eine fraktale Struktur ist eine Struktur, welche Selbstähnlichkeit besitzt. Der Romanesco (auch Pyramidenblumenkohl, ein Gemüse) beispielsweise besitzt sehr schöne fraktale Strukturen.

2 L-Systeme als Zeichenketten

Mit Hilfe von Zeichenketten kann der Prozess des Entstehens eines L-Systems einfach visualisiert werden. Das ist auch die Methode, welche später für das Generieren der L-Systeme angewendet wird. Das folgende Beispiel soll anhand eines einfachen L-Systems zeigen, wie dieser Prozess funktioniert.

Das System umfasst ein Axiom und eine Regel. Das Axiom dieses Systems ist A und die Regel definiert, dass für jede Iteration das Zeichen A durch das Wort AA ersetzt wird.

Axiom	A
Regel 1	$A \rightarrow AA$
<hr/>	
$n = 0$	A
$n = 1$	AA
$n = 2$	$AAAA$
$n = 3$	$AAAAAAAA$
$n = 4$	$AAAAAAAAAAAAAAAA$

Mit jeder Iteration verdoppelt sich die Anzahl der Zeichen. Die Anzahl der Zeichen a welche die Zeichenkette umfasst, kann durch die geometrische Folge $a_n = 2^n$ ausgedrückt werden.

Im nächsten Beispiel wächst das Mittelglied B bei jedem Schritt um jeweils 2 Zeichen. Die Zeichen A und C am linken und rechten Ende bleiben dabei aber unverändert, weil sie zwar ersetzt werden, ihre Anzahl aber, bedingt durch die beiden Regeln 1 und 2, konstant bleibt. Tritt eines der beiden Zeichen A oder C auf, wird es durch das jeweilige Zeichen und ein neues Zeichen B , welches entweder links oder rechts des alten Zeichens zu liegen kommt, ersetzt. Setzt man die angewendete Regel und das zu ersetzende Zeichen jeweils in Klammern, wird aus $(A)B \rightarrow (AB)B$ und anschliessend aus $(A)BB \rightarrow (AB)BB$.

Axiom	ABC
Regel 1	$A \rightarrow AB$
Regel 2	$C \rightarrow BC$
<hr/>	
$n = 0$	ABC
$n = 1$	$ABBBC$
$n = 2$	$ABBBBBC$
$n = 3$	$ABBBBBBBC$

Das L-System aus dem letzten Beispiel lässt sich auch um eine Regel erweitern, so dass das Axiom nur noch ein Zeichen umfasst. Die neue Regel muss so konstruiert sein, dass sie bei der ersten Iteration das Axiom ABC aus dem letzten Beispiel erzeugt. Die Variable für die neue Regel darf aber nicht eine sein, welches in einer Regel des L-Systems bereits als Ausgabe vorkommt, weil die Regel ansonsten in den nachfolgenden Iterationsschritten wiederum zutreffen würde. Eine Regel für die Variable B zu formulieren, würde eine andere Zeichenkette

erzeugen, weil die Variable B in den Regeln 2 und 3 bereits verwendet wird.

Axiom	D
Regel 1	$D \rightarrow ABC$
Regel 2	$A \rightarrow AB$
Regel 3	$C \rightarrow BC$
<hr/>	
$n = 0$	D
$n = 1$	ABC
$n = 2$	$ABBBC$
$n = 3$	$ABBBBBC$
$n = 4$	$ABBBBBBBC$

3 Visualisierung

Es wird klar, dass sich so bereits mit wenigen Regeln optisch ansprechende Strukturen erzeugen lassen, deren Darstellung sich in den vorangegangenen Beispielen aber alleine auf Zeichenketten beschränkt hat.

Um ein L-System auch grafisch in einer Ebene darzustellen, wird eine Methode benötigt, welche es erlaubt, mit einfachen Mitteln Zeichenoperationen auszuführen. Eine Lösung ist die Verwendung von sogenannten *Turtle-Graphics*. Eine virtuelle Schildkröte bewegt sich dabei auf einer Ebene ϵ , die gleichzeitig das Papier darstellt, auf welches gezeichnet wird. Die Schildkröte verfügt über einen Zeichenstift und malt mit diesem Linien auf die Ebene ϵ , wenn sie diese durchfährt. Die Schildkröte kann sich auf der Ebene ϵ aber auch bewegen, ohne dass sie dabei gleichzeitig eine Linie zeichnet.

3.1 Die Schildkröte

Die Schildkröte wird in der Ebene ϵ mit ein paar wenigen Befehlen gesteuert, welche jeweils genau ein Zeichen lang sind. Grossbuchstaben entsprechen dabei einer Vorwärtsfahrt mit Zeichnen einer Linie, Kleinbuchstaben einer Vorwärtsfahrt ohne Zeichnen einer Linie und die Befehle $+$ und $-$ verändern die Orientierung α der Schildkröte im Koordinatensystem der Ebene ϵ jeweils um den Winkel δ .

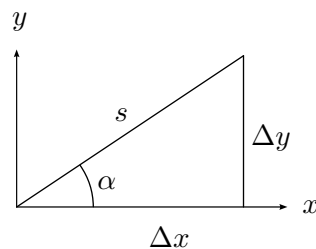


Abbildung 1: Bewegung der Schildkröte relativ zur aktuellen Position. Der Winkel α stellt dabei die Orientierung der Schildkröte dar.

Befehle zur Steuerung der Schildkröte:

- F Vorwärtsfahrt um die Strecke s , wobei die zurückgelegte Strecke s auf der Ebene ϵ als Linie eingezeichnet wird. $\Delta x = s \sin \delta$, $\Delta y = s \cos \delta$ (siehe auch Abb. 1)
- f Vorwärtsfahrt um die Strecke s .
- + Drehung nach links um den Winkel δ . (Addiert δ zu α)
- Drehung nach rechts um den Winkel δ . (Subtrahiert δ von α)

Zu Beginn ist die Schildkröte nach oben, also in Richtung 90° orientiert. (Abb. 2) Mit den Befehlen in der vorangehenden Liste und den beiden Parametern s und δ kann die Schildkröte bereits einfache Zeichnungen erzeugen. Die auszuführenden Befehle für die Schildkröte werden vor dem Start in einer Zeichenkette abgelegt, welche dann von der Schildkröte zeichenweise interpretiert wird.

3.2 Beispiele

Die Abbildungen 3a und 3b zeigen zwei Beispiele von Zeichnungen, welche durch die Schildkröte durch Interpretation der jeweiligen Zeichenkette erzeugt werden. Der Winkel δ beträgt bei beiden Beispielen 90° , unterschiedlich ist aber der Parameter s , welcher in Abbildung 3a 5 und in Abbildung 3b 1 beträgt. Das Quadrat in Abbildung 3a bekommt dadurch eine Seitenlänge von 5, die Seitenlänge beim L in Abbildung 3b ist gleich gross wie das zugrunde liegende Raster.

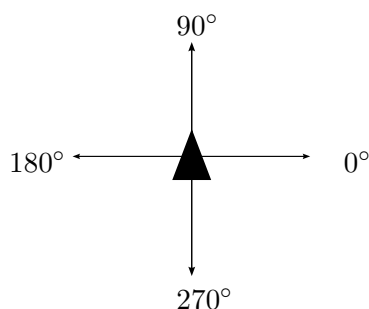
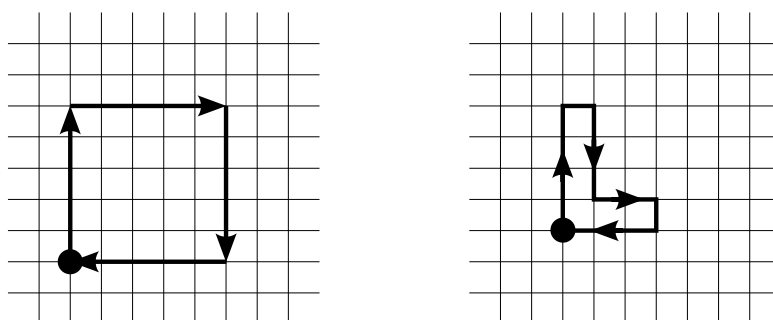


Abbildung 2: Die Schildkröte vor dem Start



(a) Ein Quadrat ($\delta = 90, s = 5$,
Befehle: $F-F-F-F$)

(b) Ein L ($\delta = 90, s = 1$, Befehle:
 $FFFF-F-FFF+FF-F-FFF$)

Abbildung 3: Zwei Beispiele für von der Schildkröte erzeugte Zeichnungen

4.2 Die Visualisierung

Wiederholt man das Ersetzen jeder Seite und ersetzt sie jeweils wiederum durch das vorgegebene Muster, ergibt sich nach ein paar Iterationsschritten eine einer Schneeflocke ähnliche Struktur. Die Entstehung der Kochschen Schneeflocke ist in Abbildung 6 zu sehen. Es sind die ersten vier Iterationsschritte für $0 < n < 5$ zu sehen.

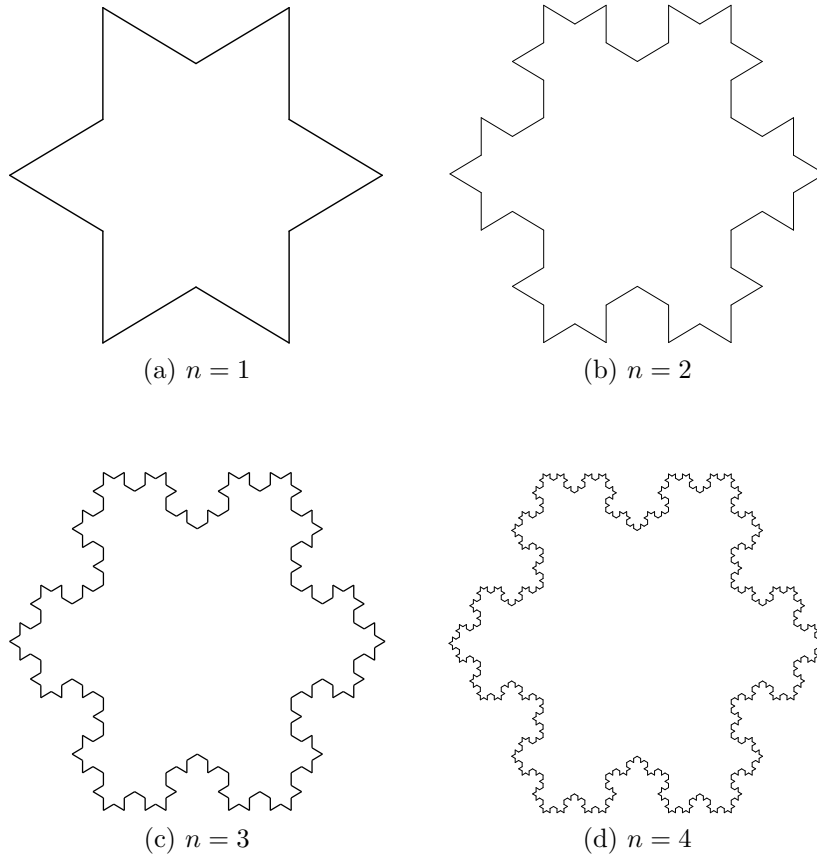


Abbildung 6: Die Entstehung der Kochschen Schneeflocke

5 L-Systeme mit Speicher

Um sich verzweigende Strukturen erzeugen zu können, muss die Schildkröte mit einer Möglichkeit ausgestattet werden, ihre aktuelle Position zu speichern, so dass diese anschliessend wieder abgerufen werden kann, und die Schildkröte zu ihrer ursprünglichen Position zurückkehrt. Damit wird es möglich, ausgehend von der aktuellen Position der Schildkröte einen Ast zu modellieren und die Schildkröte nach Abschluss des Astes wieder an ihre Startposition, also an den Beginn des Astes, zurückkehren zu lassen.

5.1 Der Speicher

Die Position der Schildkröte wird auf einem Stapelspeicher abgelegt, so dass sich an der obersten Stelle des Speichers immer die zuletzt abgelegte Position der Schildkröte befindet. Der Stapelspeicher kann beliebig viele Positionen der Schildkröte speichern. Dadurch ist es möglich, innerhalb eines Astes einen neuen Unterast zu starten und nach dessen Abschluss an die Ausgangsposition des Unterastes zurückzukehren.

Bei Abruf einer Position vom Speicher wird immer die zuletzt gespeicherte Position zurückgegeben. Es ist also nicht möglich, einen Ast A zu starten, dann einen Unterast B zu starten und anschliessend an die Startposition des Astes A zurückzukehren, bevor der Unterast B abgeschlossen ist. Eine Position auf dem Stapelspeicher, welche nicht an oberster Stelle liegt, kann also nicht abgerufen werden.

5.2 Befehle für den Speicher

Zur Steuerung des Speichers werden zwei neue, ebenfalls ein Zeichen umfassende, Befehle eingeführt. Einer zum Speichern und Ablegen der aktuellen Position auf dem Stapel, und einer zum Abrufen und Wiederherstellen der Position, welche sich an oberster Stelle des Stapels befindet. Die Liste 5.2 zeigt die beiden neuen Befehle im Überblick.

- [Startet einen neuen Ast. Dazu wird die aktuelle Position der Schildkröte zuoberst auf den Stapelspeicher abgelegt.
-] Beendet einen Ast, indem die an oberster Stelle des Stapelspeichers liegende Position abgerufen und die Schildkröte anschliessend an diese zurückgesetzt wird. Die abgerufene Position wird dabei vom Stapelspeicher gelöscht.

5.3 Beispiel 1

Abbildung 7 zeigt eine Anwendung der neuen Befehle zum Starten und Beenden eines Astes. Die Struktur verfügt über zwei Äste, von welchen der erste bei Punkt A und der zweite bei Punkt C beginnt. Die Befehle für die Schildkröte zur Erzeugung der Abbildung 7 sind $F[+F]F[-F]F$. Diese Befehle werden von der Schildkröte zeichenweise interpretiert und dann visualisiert. Im Detail läuft dieser Vorgang wie folgt ab:

Start Die Schildkröte befindet sich in Punkt O und ist nach oben in Richtung 90° orientiert.

F Vorwärtsfahrt zu Punkt A . Zeichnet die Linie \overline{OA}

- [Starten eines neuen Astes. Die Position und die Orientierung der Schildkröte wird auf den Speicher gelegt.
- +F Zeichnet eine Linie von Punkt A zu Punkt B . Der Ast ist, bedingt durch den Befehl +, nach links geneigt.
-] Beendet den aktuellen Ast. Dazu wird vom Speicher die oberste Position abgerufen und die Schildkröte an diese zurückversetzt. Die Schildkröte kehrt also zum Punkt A zurück und ist wieder nach 90° orientiert.
- F Vorwärtsfahrt zu Punkt C . Zeichnet die Linie \overline{AC}
- [Starten eines neuen Astes. Position und Orientierung der Schildkröte werden auf dem Speicher abgelegt.
- F Zeichnet eine Linie von Punkt C zu Punkt D . Dieser zweite Ast ist nach rechts geneigt.
-] Beendet den aktuellen Ast und kehrt zu Punkt C zurück.
- F Vorwärtsfahrt zu Punkt E . Zeichnet die Linie \overline{CE}

5.4 Beispiel 2

Das folgende L-System ist ein einfaches System mit Ästen und einer Regel. Die Regel erzeugt dieselbe Struktur wie sie in Abbildung 7 zu sehen ist. Der Unterschied ist einzig, dass sich die Struktur innerhalb der Äste wiederholt. In Abbildung 8 sind die ersten vier Iterationen dieses L-Systems zu sehen.

Axiom	F
Regel 1	$F \rightarrow F[+F]F[-F]F$
δ	45°

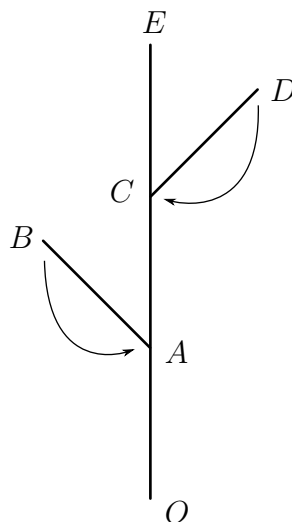


Abbildung 7: Beispiel einer Struktur mit zwei Ästen

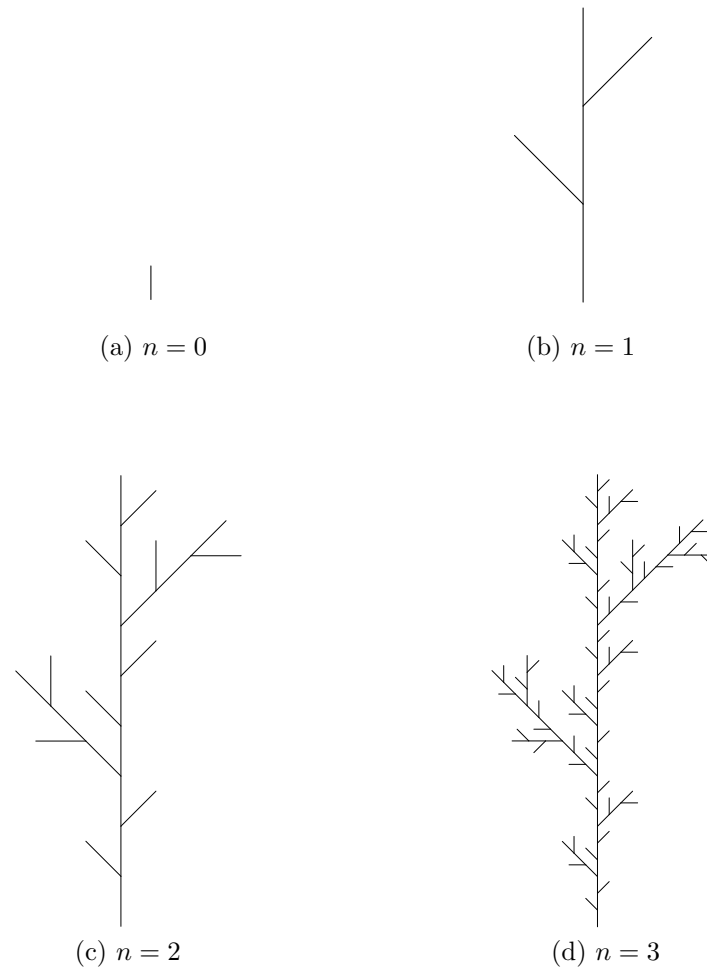


Abbildung 8: Einfaches verzweigtes L-System mit einer Regel ($\delta = 45^\circ$)

6 Zusammenfassung

Mit den bis jetzt vorgestellten Befehlen können bereits vielfältige Strukturen erzeugt werden. Der Stapelspeicher für die Position der Schildkröte bringt neue Möglichkeiten zur Gestaltung von Strukturen welche echten Pflanzen ähnlich sehen. Mit ein wenig Übung und Geduld ist es sogar möglich, existierende Arten mit L-Systemen nachzubilden. Das ist genau der Zweck, für den Lindenmayer die L-Systeme ursprünglich entworfen hatte. (Siehe dazu auch [1]) Selbstverständlich werden die künstlich erzeugten Strukturen einer echten Pflanze immer nur ähnlich sehen und sie nicht perfekt nachbilden können.

Die folgende Übersicht fasst sämtliche vorgestellten Befehle zur Steuerung der Schildkröte noch einmal zusammen und soll auch als Referenz für das Programm LSys/JS dienen, welches in Kapitel 7 vorgestellt wird.

6.1 Übersicht der Befehle

6.1.1 Standardbefehle

- F Bewegung um die Strecke s in Richtung der aktuellen Orientierung. Die zurückgelegte Strecke wird dabei durch eine Linie visualisiert.
- A-Z Jeder Grossbuchstabe von $A - Z$ entspricht standardmässig dem Befehl F. In einem L-System kann jedem Grossbuchstaben eine neue Regel zugewiesen werden.
- f Bewegung um die Strecke s in Richtung der aktuellen Orientierung.
- a-z Jeder Kleinbuchstabe von $a - z$ entspricht standardmässig dem Befehl f. In einem L-System kann jedem Kleinbuchstaben eine neue Regel zugewiesen werden.
- + Änderung der Orientierung nach links im Gegenuhrzeigersinn um den Winkel δ
- Änderung der Orientierung nach rechts im Uhrzeigersinn um den Winkel δ

6.1.2 Speicherbefehle

- [Speichert die aktuelle Position und Orientierung der Schildkröte und startet dadurch einen neuen Ast.
-] Ruft die zuletzt gespeicherte Position und Orientierung der Schildkröte ab und beendet damit den aktuellen Ast.

7 Das Programm LSys/JS

Mit dem Programm LSys/JS können L-Systeme im Browser² editiert und gleichzeitig visualisiert werden. Die Ausgabe erfolgt direkt im Browser. Damit müssen keine zusätzlichen Komponenten oder Programme installiert werden. LSys/JS unterstützt alle in Kapitel 6.1 aufgelisteten Befehle. Alle in dieser Arbeit abgebildeten L-Systeme wurden ausschliesslich mit LSys/JS erzeugt und visualisiert. Für den Export von Abbildungen im Vektor-Format bietet LSys/JS einen speziellen Treiber, welcher direkt Dateien im XML-Format SVG³ erzeugen kann.

7.1 Übersicht

LSys/JS besteht vereinfacht aus drei wesentlichen Komponenten.

Der Parser liest das eingegebene L-System und extrahiert daraus die relevanten Informationen wie Regeln, Axiom und Einstellungen.

Der Compiler erzeugt aus dem geparsten L-System die Zeichenkette mit den Befehlen für die Schildkröte.

Der Interpreter interpretiert zeichenweise die vom Compiler erzeugten Befehle, steuert damit die Schildkröte und erzeugt so eine Visualisierung des L-Systems.

7.2 Der Parser

Der Parser verarbeitet das im Textfeld eingegebene L-System und stellt die darin enthaltenen Informationen den nachfolgenden Komponenten zur Verfügung. Dazu wird das L-System zeilenweise eingelesen. Der Parser erzeugt aus den eingelesenen Informationen anschliessend eine Datenstruktur welche alle Regeln, das Axiom und alle weiteren Einstellungen wie die Länge der Strecke s , den Winkel δ und die gewünschte Anzahl der auszuführenden Iterationen n enthält.

Alle Zeilen, welche mit dem Zeichen # beginnen, werden vom Parser verworfen. Damit können Kommentare an jeder Stelle in einem L-System eingefügt werden.

²LSys/JS wurde ausschliesslich für den Browser *Firefox* entworfen und auch ausschliesslich auf diesem Browser getestet. *Firefox* ist für alle verbreiteten Plattformen und Betriebssysteme kostenlos verfügbar. Zur Ausführung von LSys/JS wird mindestens die Version 1.5 benötigt.

³Der Ausgabetreiber für SVG wurde vor allem deshalb implementiert, weil für den Import von visualisierten L-Systemen in diese Arbeit Pixelgrafiken, wie sie der Ausgabetreiber für den Browser erzeugt, qualitativ nicht genügt hätten.

7.2.1 Syntax

Die Syntax für die Definition eines L-Systems in LSys/JS ist in der folgenden Übersicht dargestellt.⁴

```
axiom=( [a-zA-Z\-\+\[\]]+ )
```

Definiert das Axiom des L-Systems. Akzeptiert alle Möglichen Kombinationen der Zeichen a-z, A-Z, -, +, [, und]

```
( [A-Za-z]? ) := ( [a-zA-Z\-\+\[\]]+ )
```

Definiert eine neue Regel für die links stehende Variable. Allen Variablen von a-z und A-Z kann eine Regel zugewiesen werden. Akzeptiert alle Möglichen Kombinationen der Zeichen a-z, A-Z, -, +, [, und] Beispiel: $F := F + F + F + F$

```
iterations=( [0-9]+ )
```

Definiert die Anzahl Iterationen n für das L-System. Akzeptiert nur ganze Zahlen. Beispiel: $iterations = 3$ (Standardwert: $n = 1$)

```
angle=[0-9]+(\.[0-9]+)?
```

Definiert den Winkel δ für das L-System im Gradmass. Akzeptiert nur positive reelle Zahlen. Beispiele: $angle = 12.5$, $angle = 90$ (Standardwert: $\delta = 90$)

```
length=( [0-9]+ )
```

Definiert die Länge der Stecke s in Pixeln. Akzeptiert nur ganze Zahlen. (Standardwert: $s = 10$)

7.3 Der Compiler

Der Compiler erzeugt aus den vom Parser gelieferten Informationen eine Zeichenkette, welche die Befehle zur Visualisierung des L-Systems mit Hilfe der Schildkröte enthält. Ausgangsbedingung für den Compiler ist das Axiom, auf welches die definierten Regeln angewendet werden. Die entstandene Zeichenkette wird dann wiederum als Eingabe verwendet, auf welche die Regeln erneut angewendet werden. Dieser Schritt wird solange wiederholt, bis die gewünschte Anzahl Iterationen n erreicht ist.

In jeder Iteration geht der Compiler die gesamte Zeichenkette zeichenweise durch. Ist für den jeweiligen Buchstaben eine Regel vorhanden, wird der Buchstabe durch diese Regel ersetzt. Andere Zeichen werden vom Compiler ignoriert.

Am Ende liefert der Compiler eine Zeichenkette die alle Befehle zur Visualisierung des L-Systems enthält. Dieser Vorgang ist vergleichbar mit den L-Systemen in Kapitel 2 auf Seite 3.

⁴ Die einzelnen Zeilen eines L-Systems werden vom LSys/JS-Parser mit Hilfe von regulären Ausdrücken geparkt. Die Notation in dieser Syntax-Übersicht ist entsprechend gehalten. Mehr Informationen zu regulären Ausdrücken und deren Syntax finden sich bei [2].

7.4 Der Interpreter

Der Interpreter interpretiert anschliessend zeichenweise die vom Compiler erzeugte Zeichenkette. Er kennt alle im Kapitel 6.1 auf Seite 12 aufgelisteten Befehle. Entsprechend den Befehlen der Zeichenkette wird vom Interpreter die Schildkröte gesteuert, welche dann das L-System im Browser visualisiert. Der Interpreter verändert die vom Compiler gelieferten Befehle also nicht mehr, sondern führt diese nur noch aus.

7.5 Bedienung

Die Bedienung von L-System/JS ist grundsätzlich einfach gehalten. Im Textfeld wird zuerst das zu erzeugende L-System eingegeben. Das L-System kann durch Klick auf 'Redraw' visualisiert werden. Ist die Checkbox 'Draw Start Position' zusätzlich aktiviert, wird die Startposition der Schildkröte mit einem kleinen roten Kreis markiert.

Um das L-System an einer neuen Position darzustellen, wird einfach mit der Maus an die gewünschte neue Startposition in der Zeichenfläche geklickt. L-System/JS löscht dann die Zeichenfläche und stellt das L-System an der angeklickten Position neu dar.

Mit 'width' und 'height' kann die Grösse der Zeichenfläche in vorgegeben Grenzen verändert werden. Mit einem Klick auf '+' oder '-' wird die Zeichenfläche entsprechend vergrössert oder verkleinert. 'Max.' vergrössert die Zeichenfläche auf die maximal mögliche Grösse.

Die Auswahlliste 'Presets' bietet eine Auswahl von vorgefertigten L-Systemen, die bei Auswahl geladen und dargestellt werden.



Abbildung 9: Das Programm L-System/JS nach dem Start

8 Fazit

L-Systeme, wie sie in dieser Arbeit vorgestellt wurden, bieten die Möglichkeit, mit wenig Aufwand komplexe und optisch ansprechende und verblüffende Strukturen zu erzeugen. Durch Verändern weniger Befehle oder durch Justieren eines einzigen Parameters können bereits neue, sich nicht mehr ähnlich sehende Strukturen generiert werden. Meistens wirkt sich eine kleine Änderung z.B. am Winkel δ so aus, dass sich dadurch bereits ein völlig neues Ergebnis ergibt. Mit ein wenig Geduld und Experimentiertrieb lassen sich mit dem Programm L`Sys`/JS faszinierende Bilder kreieren. Meistens ergeben sich durch Probieren und Editieren von bestehenden Regeln zufällig neue Ergebnisse. Einige der in dieser Arbeit abgebildeten L-Systeme sind alleine durch Zufall während dem Experimentieren mit einem bestehenden L-System entstanden.

Leider erschöpfen sich die Möglichkeiten aber auch bald, denn die hier gezeigte Methode zur Visualisierung beschränkt sich auf eine Ebene und kann beispielsweise keine L-Systeme in 3D visualisieren. Der Grund dafür ist der relativ grosse Aufwand, welche die Visualisierung in 3D bei der Implementation mit sich gebracht hätte. Die Visualisierung in einer Ebene bringt keinen grossen Zusatzaufwand, weil sie als Komponente bereits im Browser integriert ist.

Die Implementation des Programms L`Sys`/JS hat aber dennoch einiges an Zeit verschlungen, der Aufwand hat sich aber gelohnt, steht damit doch ein Programm zur Verfügung, welches, nicht wie die meisten bestehenden Lösungen, eine lokale Installation erfordert. Ausserdem kann ein L-System einfach editiert und dann durch einen einzigen Mausklick visualisiert werden. Dadurch werden die gemachten Änderungen an einer Regel oder an einem Parameter direkt sichtbar. In vielen bestehenden Lösungen muss für Änderungen an den Regeln ein separates Editor-Fenster geöffnet werden, in welchem die Änderungen dann vorgenommen werden.

Die Welt der L-Systeme ist eine faszinierende und verblüffende und vermutlich wird mich dieses Thema auch weiterhin beschäftigen. Nach der Implementation von L`Sys`/JS liegt das Erstellen eines Programms nahe, dass eine Visualisierung auch in 3D bieten würde. Ein Nachteil von L`Sys`/JS ist die relativ niedrige Geschwindigkeit, bedingt durch die Programmierung in Javascript und die Tatsache, dass das Programm im Kontext des Browsers läuft. So führen bereits L-Systeme mit mehr als fünf bis sechs Iterationen zu sehr langen Rechenzeiten bei der Visualisierung. Dies ist aber vor allem durch die verwendete Canvas-Komponente⁵ bedingt, welche bei einer grossen Anzahl von zu zeichnenden Pfaden offenbar sehr langsam wird. Eine nächste Implementation würde ich deshalb wohl in einer anderen Programmiersprache und vor allem unter Verwendung beschleunigter Grafikfunktionen durchführen.

⁵ Die Canvas-Komponente ist Teil der WHATWG (Web Hypertext Application Technology Working Group) Spezifikation und bietet unter anderem die Möglichkeit, Grafiken mit Hilfe von Pfaden zu erzeugen. (Vgl. auch [3])

Literatur

- [1] Przemyslaw Prusinkiewicz and Aristid Lindenmayer: *The Algorithmic Beauty of Plants*. 2nd Printing. New York: Springer Verlag, 1996
- [2] Jeffrey E Friedl: *Mastering Regular Expressions*. 2nd Edition. Sebastopol: O'Reilly & Associates, 2002
- [3] Web Hypertext Application Technology Working Group: "Web Applications 1.0 Working Draft – 4 November 2006". URL: <http://www.whatwg.org/specs/web-apps/current-work/> [Stand: 5.11.2006]
- [4] Wikipedia: "Lindenmayer-Systeme" URL: <http://de.wikipedia.org/wiki/Lindenmayer> [Stand: 5.11.2006]
- [5] Wikipedia: "L-system" URL: <http://en.wikipedia.org/wiki/L-system> [Stand: 5.11.2006]
- [6] Wikipedia: "Formal Language" URL: http://en.wikipedia.org/wiki/Formal_language [Stand: 5.11.2006]
- [7] Wikipedia: "Formal Grammar" URL: http://en.wikipedia.org/wiki/Formal_grammar [Stand: 5.11.2006]
- [8] SVG Working Group: "Scalable Vector Graphics (SVG) 1.1 Specification". URL: <http://www.w3.org/TR/2003/REC-SVG11-20030114/> [Stand: 5.11.2006]

A Ausgesuchte L-Systeme



A.1 Pflänzchen I

$$n = 4, \delta = 8.5$$

$$+A-B-C-D+E$$

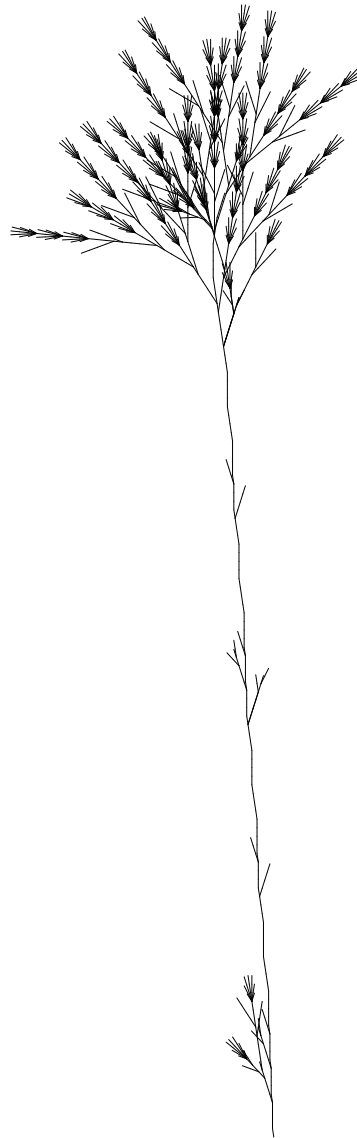
$$A \rightarrow FFFF [-AE] F [+++AE] FFF$$

$$B \rightarrow FFFF$$

$$C \rightarrow F [+++CE] F [-AE] F$$

$$D \rightarrow FFF$$

$$E \rightarrow [---F] [--FF] [-FFF] [+++F] [++FF] [+FFF] FFF$$



A.2 Pflänzchen II

$n = 4, \delta = 8.5$

+ZA-B-C-D+E

Z→F-FFFY+ZZX

Y→[++FFC]

X→[---FFFC]

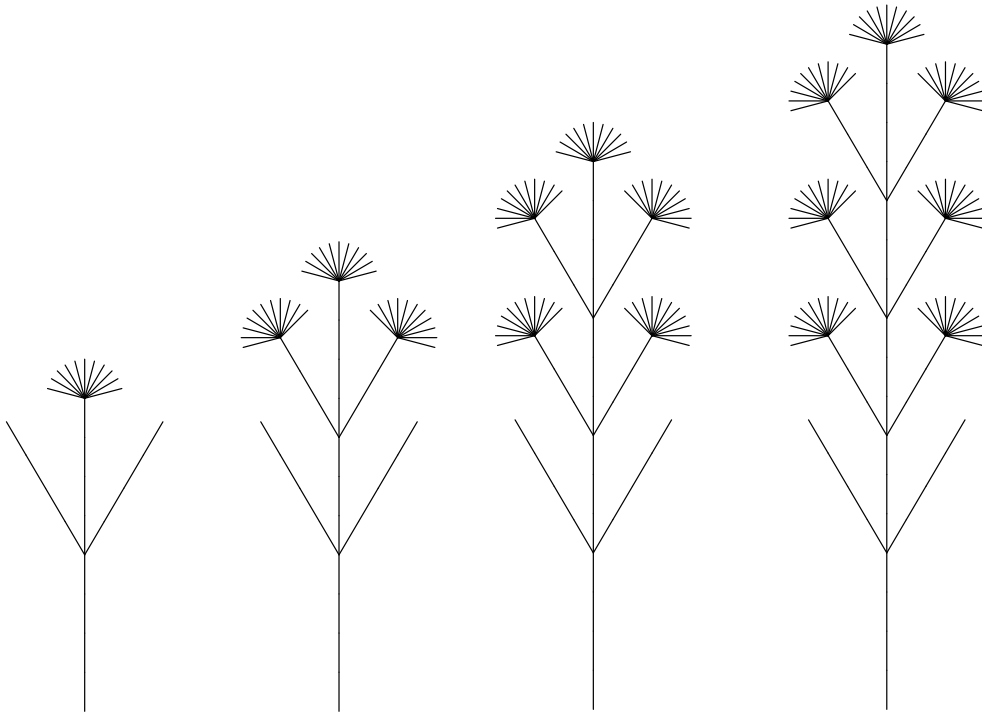
A→FFFF [--AEEE] F [+++AEEE] FFF

B→FFFF

C→F [+++CE] F [-AE] F

D→FFF

E→[---F] [--FF] [-FFF] [+++F] [++FF] [+FFF] FFF



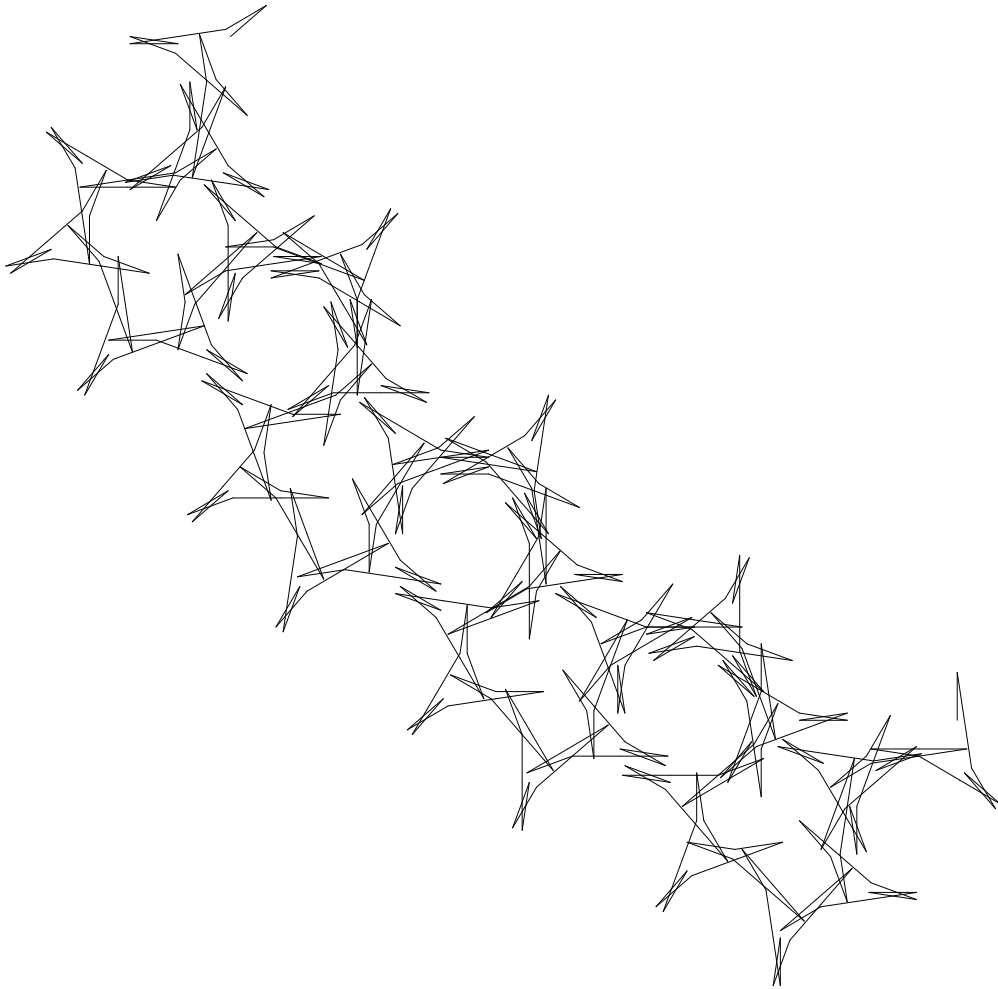
A.3 Pflänzchen III

$n = 4, \delta = 15$

AFFFFB

$A \rightarrow AFFF [--FFFB] [++FFFB]$

$B \rightarrow [+++++F] [++++F] [+++F] [++F] [+F] [F] [-F] [--F] [---F] [----F] [-----F]$

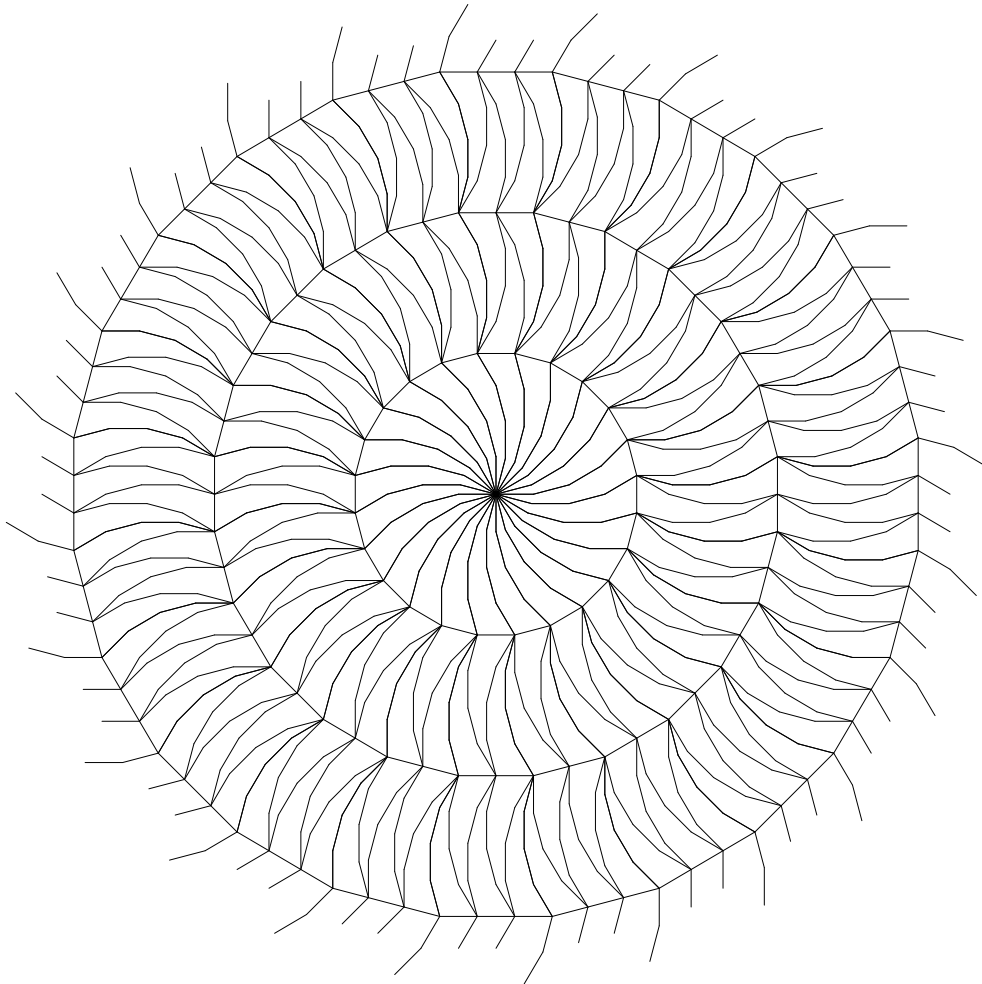


A.4 Alge

$n = 3, \delta = 170$

F-F-F-F

F→F-FF--F-F



A.5 Spirale

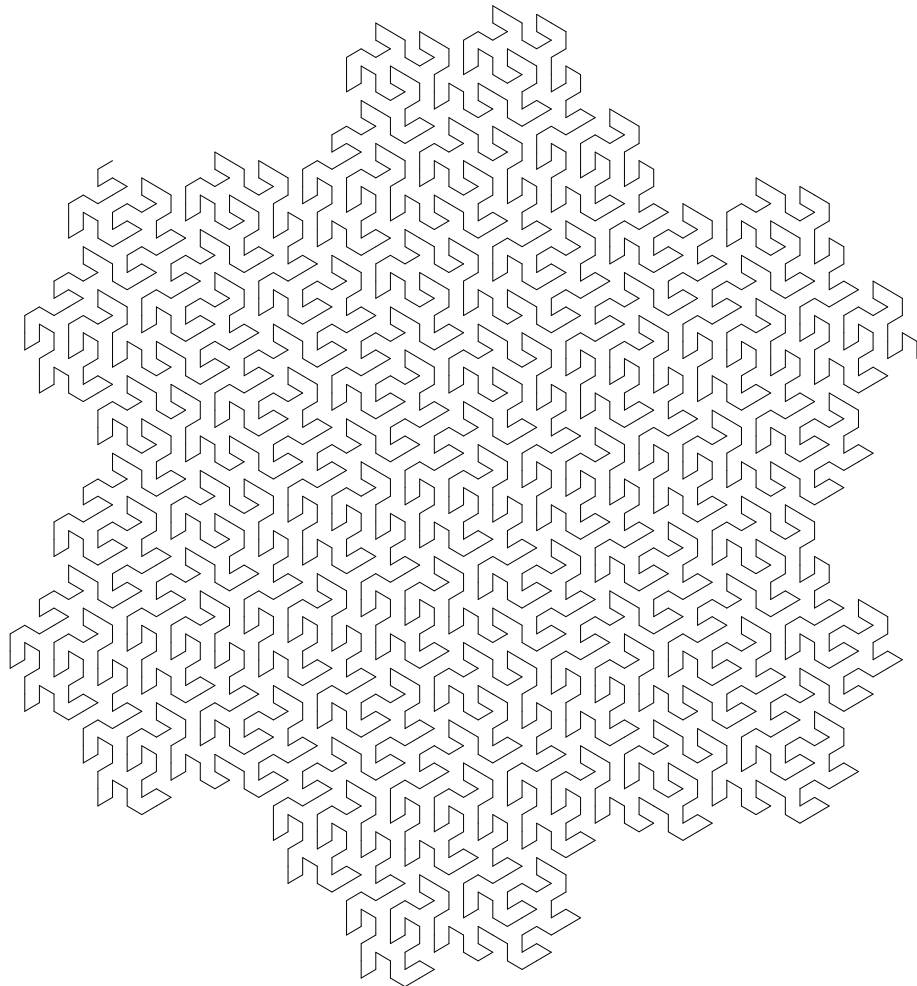
$n = 4, \delta = 15$

AAAA

$A \rightarrow B+B+B+B+B+B+$

$B \rightarrow [F+F+F+F [---B-C] +++++F++++++F-F-F-F]$

$C \rightarrow [F+F+F+F [---C] +++++F++++++F-F-F-F]$



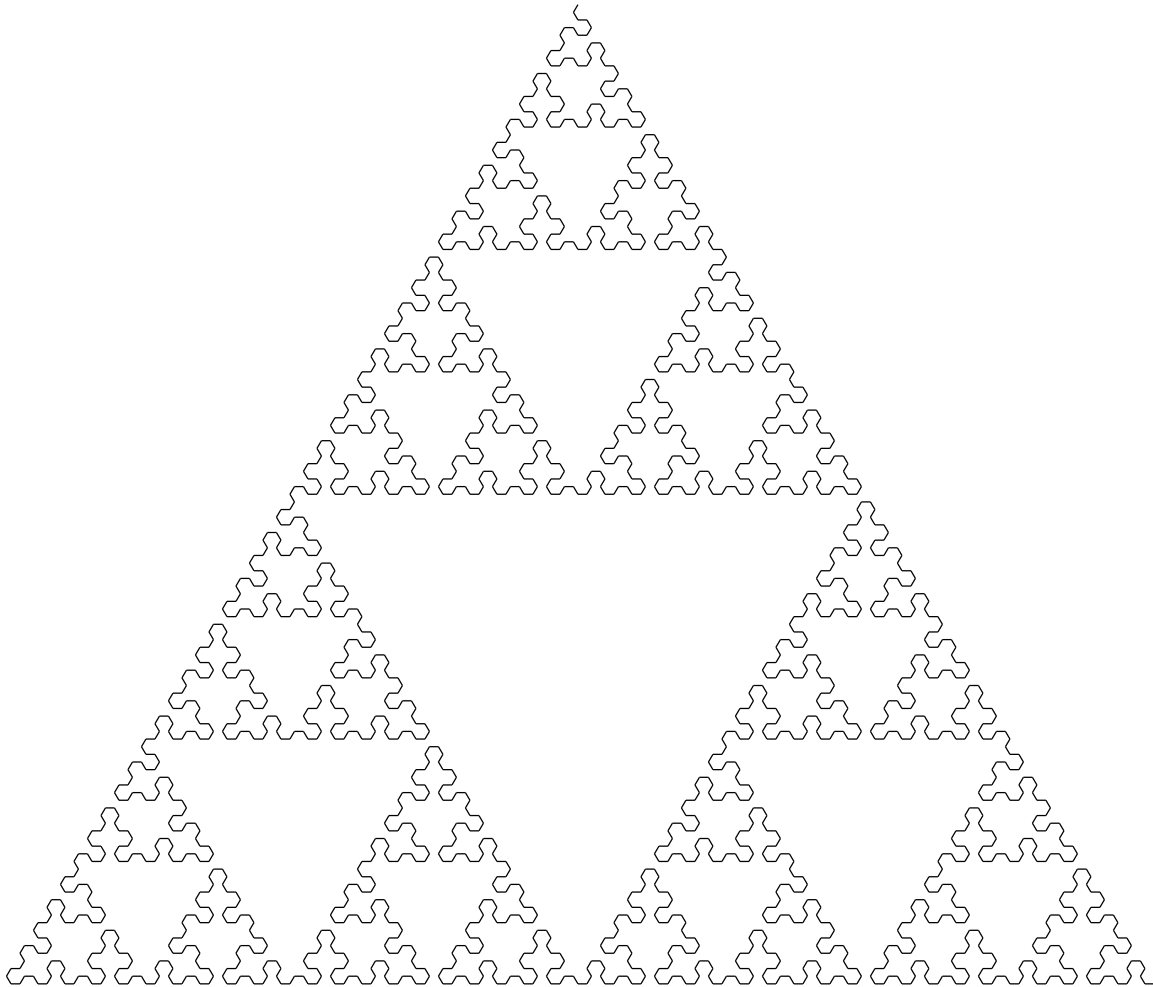
A.6 Gosper-Kurve

$n = 4, \delta = 60$

A

$A \rightarrow A+B++B-A--AA-B+$

$B \rightarrow -A+BB++B+A--A-B$



A.7 Sierpinski-Dreieck

$n = 7, \delta = 60$

A

$A \rightarrow B-A-B$

$B \rightarrow A+B+A$

B Kopie LSys/JS

Der auf dieser Seite beiliegende Datenträger enthält eine Kopie der Webseite <http://lsysjs.qwert.ch> mit Stand 10.11.2006. Auf der Webseite findet sich jeweils die aktuelle Version des Programms LSys/JS. Die Kopie wurde nur der Vollständigkeit halber beigelegt, da das Programm LSys/JS Teil dieser Maturarbeit ist.

C Unterzeichnete Erklärung

Hiermit erkläre ich, vorliegende Maturitätsarbeit selbständig und mit Hilfe der ausdrücklich deklarierten Quellen verfasst zu haben.

Zürich, 10. November 2006

Philipp Stucki