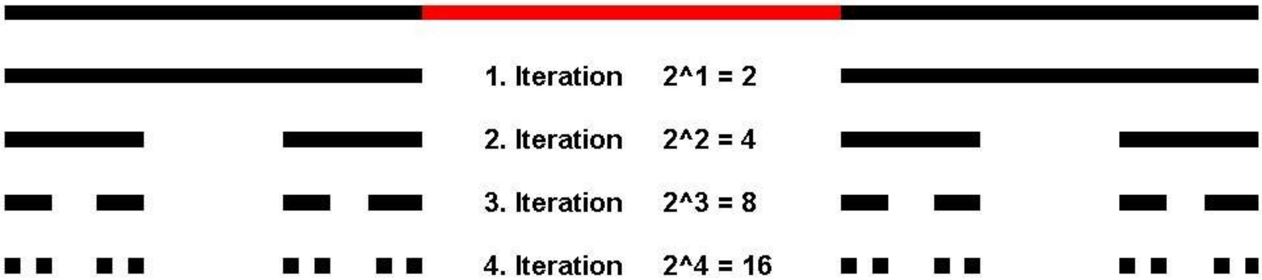
20 November 2003 Version 0.1

Inhalt

1.0 Kleiner Überblick der Fraktale

Fraktale sind eine relativ neue Disziplin in der Mathematik. Der Grundgedanke stammte Von Felix Hausdorf der 1919 die Idee von gebrochenen Dimensionen hatte.
Wie sich Fraktale entwickeln hängen sehr stark vom „Anfangswert“ ab. Je mehr Iterationen ausgeführt werde um so mehr muss berechnet werden. Aus diesem Grund kann man ein Fraktal konkret nur bis zu einer bestimmten Iterationstiefe darstellen. Marginale Veränderungen der Iterationstiefe können „jeden“ noch so Leistungsfähigen Computer an die Grenzen seiner Leistungsfähigkeit treiben.

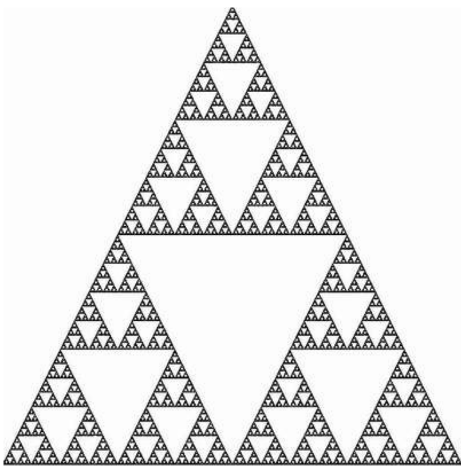
Wir kennen alle den zwei und den dreidimensionalen Raum. Im 2D Raum sind Linien, Kreise und beliebige andere „flache“ Figuren darstellbar. Im 3D Raum ist das äquivalent, nur das es sich eben um „Körper“ handelt. Aber gibt es auch Dimensionen die nicht gerade sind, also deren Dimension durch eine „gebrochene“ Zahl angegeben wird ?
Ja die gibt es und hierbei spricht man von einer fraktalen Dimension.
(Fragment = Bruchstück). Man spricht auch oft von „selbstähnlichen Systemen“, wenn man über Fraktale redet. Als kleines Beispiel möchte ich den Cantor Staub, auch als Cantor Menge oder Wischmenge bekannt, anführen der von Georg Cantor entdeckt wurde. Ihre Eigenschaft ist das sie mehr als ein Punkt, aber weniger als eine Linie ist. Stellen sie sich eine Linie vor die gleichmäßig gedrittelt wird, und wo das Mittelstück herausgenommen wird. Wir haben nun zwei gleichlange Linien deren Abstand genau so lang wie sie selber sind. Dieser Vorgang, der gleichmäßigen Drittelung wird jetzt Iterativ auf die zwei Linien angewandt. Im nächsten Schritt erhalten wir vier Linien, im übernächsten acht Linien und so weiter. Das Interessante ist, wenn man die Iterationen gegen unendlich gehen lässt, ist die Summe der einzelnen Teillinien gleich Null.

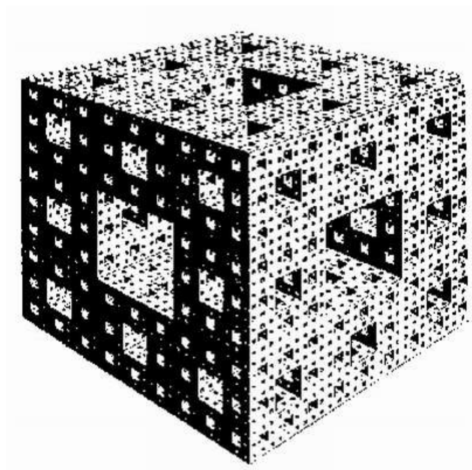


Beispiel : Cantor Staub

Die Anzahl der Linien lässt sich folgendermaßen berechnen, 2^Iterationstiefe.
Um die Länge der Linien zu erhalten rechnet man (2/3)^Iterationstiefe.
Der Cantor Staub hat eine Dimension die nach der Formel Dim = log(2) / log(3) = 0,6309 berechnet wird. (Diese Dimension wird auch Hausdorff-Dimension genannt).

Ähnlich wie beim Cantor Staub verhält es sich beim Sierpinski Dreieck, welches mehr als eine Linie ist aber nicht ganz eine Fläche, und beim Menger Schwamm welcher mehr als eine Fläche, aber weniger als ein Körper ist.


Sierpinski Dreieck


Menger Schwamm

Der Cantor Staub ist nur eines von unendlich vielen Fraktalen. Nachfolgend wird eine kleine Liste der bekanntesten Fraktale aufgezeigt

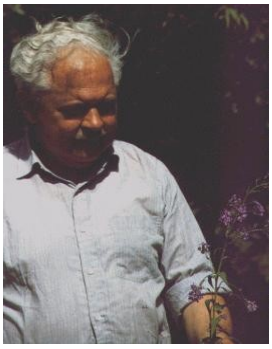
- Mandelbrot Menge (Apfelmännchen)
- Sierpinski's Flickenteppich, Sierpinski's Dreieck
- Cantor Staub
- Kochsche Kurve, Kochsche Schneeflocke
- Julia Menge
- Menger Schwamm
- Hilbert Kurve
- Lindenmayer System
- Paeno Kontinuum
- Drachen Kurve
- Pythagoras Baum

Diese Liste ließe sich nahezu endlos fortsetzen, nur das nicht alle in der Natur auftretenden Erscheinungen namentlich benannt sind. Die meisten Pflanzen folgen „selbstähnlichen" Wachstumsmustern.
Farne, Bäume, Korallen ja sogar Blumenkohl sind Fraktale der Natur.
Wichtiges zu Merken ist : L-Systeme sind keine Fraktale! Aber L-Systeme eigenen sich hervorragend um Fraktale Strukturen zu beschreiben.

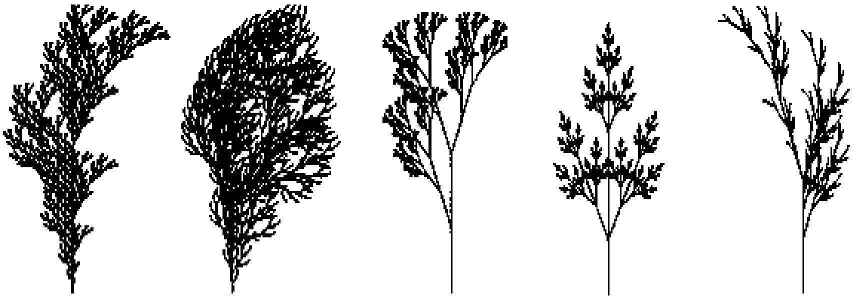
2.0 Das Lindenmayer System

2.1 Geschichtliches

Der dänische Biologe Aristid Lindenmayer (1925 - 1989) führte im Jahre 1968 die gleichnamigen Systeme, die auch kurz L-Systeme genant werden, ein. In der englischen Sprache nennt man sie „parallel rewriting systems“. Lindenmayers Ziel war es biologisches Wachstum zu simulieren. Um 1974 wurden L-Systeme von P.Hogeweg insofern erweitert dass sie eine grafische Interpretation erlaubten.
A.R Smith erklärte 1984 die konzeptionelle Ähnlichkeit von L-Systemen zu Fraktalen. Auch der Mathematiker Grzegorz Rozenberg und der Informatiker Premyszlaw Pruusinkiewicz leisteten auf dem Gebiet der L-Systeme Pionierarbeit.



Aristid Lindenmayer



Einigen nach einem L-System konstruierte Biologische Strukturen

2.2 Grundidee hinter L-Systemen

Der Grundgedanke von L-Systemen ist relativ einfach, er beruht auf einem Ersetzungs-System. Nicht ohne Grund spricht man hier auch von „selbstähnlichen“ Systemen.
Man hat ein Alphabet aus formalen Anweisungen (Linie zeichnen, um x Grad nach links drehen, um x grad nach rechts drehen, aktuelle Position speichern, aktuelle Position abrufen etc). Dann benötigt man einen Startwert der ein Element des Alphabetes ist, und eine Produktionsregel welche eine Zeichenkette aus Elementen des Alphabetes ist. Der Startwert wird nun iterativ durch die für ihn in der Produktionsregel festgelegte Zeichenkette substituiert.

L-Systeme bedienen sich einer formalen Grammatik, die der der Chomsky Grammatik sehr ähnlich ist. Die wesentlichen Unterschiede sind dass bei L-Systemen eine synchrone Verarbeitung der Produktionsregel stattfindet. Außerdem gibt es bei L-Systemen keine Differenzierung zwischen Terminal-, und Nicht-Terminal Symbolen.

Beispielhaft sei folgendes gegeben :
- Ein Alphabet **V** (endliche Menge formaler Symbole)
- Ein Startwert **w** (Axiom, Initiator)
- Eine Produktionsregel **P** (kann nur Elemente aus V enthalten)

Schauen wir uns folgendes Beispiel an:

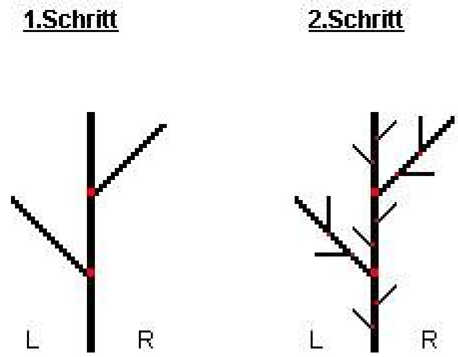
Zuerst stellen wir ein geeignetes Alphabet auf,
V = { F, + , - , [,] }
F = Strich,   + = Linksdrehung,   - = Rechtsdrehung, [= Position speichern,   ] = Position laden.

Der Startwert wird festgelegt, (Initiator)
w = F
und zu guter letzt die Produktionsregel. (Generator)
P = { F [+ F] F [-F] F }

Wenn wir diese Vorschrift nun in eine graphische Darstellung im 2D Raum überführen wollen müssen wir noch konkrete Werte für den Winkel, die Strichlänge, und die „Startrichtung“ angeben. Die Startrichtung ist in folgendem Beispiel = oben, der Winkel beträgt 45 Grad und die Strichlänge Ist circa ein Zentimeter lang. Dabei sei erwähnt dass sich die Strichlänge mit jedem weiteren Iterationsschritt ohnehin verändert.
Folgende Grafik zeigt ein deterministisches, zwei dimensionales, kontextfreies, nicht parametrisiertes L-System. Kurz ein 2D D0L System.
(diese Eigenschaften werden im Anschluss erläutert.)

Produktionsvorschrift

F	Strich (Stamm unten)
[	Speichere Position & Richtung
+	Linksdrehung
F	Strich (erste Verzweigung links)
]	Position & Richtung laden
F	Strich (Stamm mitte)
[	Speichere Position & Richtung
-	Rechtsdrehung
F	Strich (erste Verzweigung rechts)
]	Position & Richtung laden
F	Strich (Stam oben)



Beispiel für eine einfache Produktionsvorschrift

2.3 Arten von L-Systemen

L-Systeme können folgende Eigenschaften besitzen:

- Deterministisch, nicht deterministisch (stochastisch)
Stochastische L-Systeme enthalten Zufallselemente. Deterministische Deterministische enthalten keinen Zufall.
Allgemein steht D für deterministische L-Systeme.

- Kontext frei, Kontext sensitiv
Bei Kontext sensitiven L-Systemen beeinflussen die Nachbarn sich gegenseitig. Nicht so bei Kontext freien L-Systemen.
Allgemein steht 0L für Kontext freie L-Systeme.

- multidimensionale Interpretationen
Je nach Dimension für die das L-System gedacht ist, ist eine andere Produktionsregel notwendig.

- parametrisiert, nicht parametrisiert
Enthalten die einzelnen Elemente Parameter so spricht man von parametrisiert, ansonsten von nicht parametrisierten L-Systemen.
Da man nicht alle beliebigen Muster mit nicht parametrisierten L-Systemen darstellen kann ist man zu einer Parametrisierung übergegangen. Bei einer solchen wird ein Formales Symbol des Alphabetes mit einem numerischen Wert verbunden.

2.4 Möglichkeiten mit L-Systemen

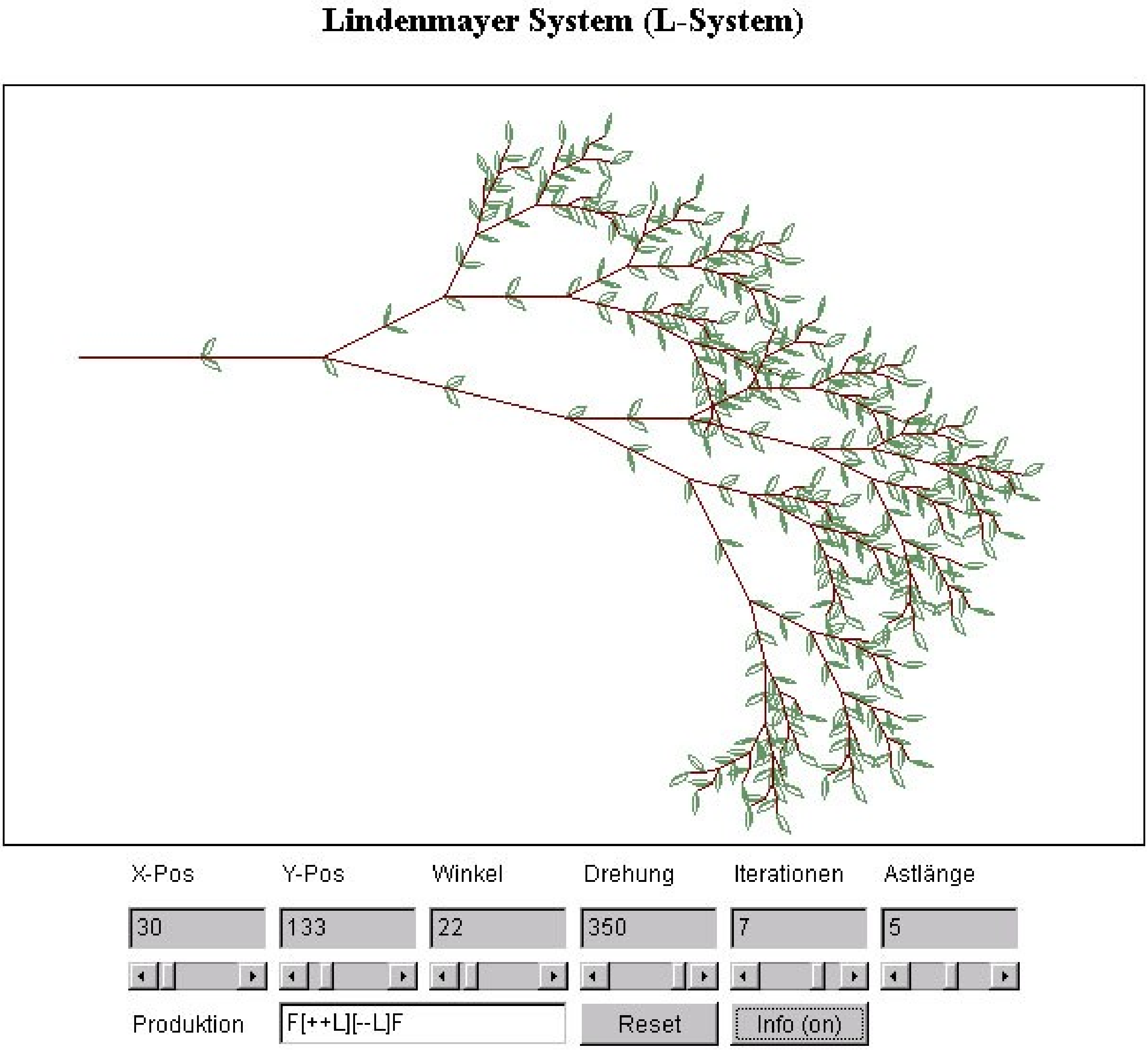
In der Fraktalen Welt gibt es Kurven mit sehr außergewöhnlichen Eigenschaften wie beispielsweise die Kochsche Schneeflocke die eine endliche Fläche besitzt aber einen unendlichen Umfang. Solche kurven nennt an wegen ihrer kuriosen Eigenschaften auch „Monsterkurven“. L-Systeme eigenen sich hervorragend um solche Kurven Darzustellen. Einige Beispiele von bekannten Fraktalen die sich durch L-Systeme relativ einfach darstellen lassen sind folgende (Liste erhebt keinen Anspruch auf Vollständigkeit):

- Sierpinski Dreieck
- Sierpinski’s Flickennteppich
- Cantor Staub
- Kochsche Schneeflocke

3.0 Das Projekt

3.1 Die Zielsetzung der Projektarbeit

Das Ziel des Projektes ist die grafische Ausgabe eines L-Systems.
Dieses L-System soll einen Ast bzw. einen Baum simulieren.
Verschiedene Parameter des Astes sind dabei einstellbar.
Die Tiefe der Iterationen, die relative Länge der einzelnen Äste,
die Ausrichtung (In welche Richtung der Baum „zeigt"), der Winkel
der den Abstand der einzelnen Äste und Blätter zueinander beeinflusst
und die Annordnung im 2D-Raum die durch die X bzw. Y Achsen Verschiebung
(Translation) bewerkstelligt wird.
Außerdem enthält das Programm einen „Rücksetzen" (reset) Knopf
der die ursprünglichen Startwerte wieder herstellt. Sowie ein Info Knopf
der in einer kleinen Legende die einzelnen Elemente des Alphabetes aufzeigt
und stichwortartig ihre Funktion beschreibt.
Die wichtigste und letzte Funktion ist ein Feld in dem die Produktionsvorschrift
steht, welche durch den Benutzer verändert werden kann.
So können neue Blätter, die in verschiedenen Winkeln abstehen, neue Äste die
sich je nach Formel „verbiegen" lassen, hinzugefügt werden.
Der Phantasie sind hierbei keine Grenzen gesetzt.



Das L-System Applet unter www.tolschock.de/java

3.2 Der Algorithmus

Der Algorithmus „turtle“ der folgend aufgelistet wird ist das eigentliche Kernstück. durch ihn wird festgelegt wie zu reagieren ist wenn die einzelnen Elemente des Alphabets durchlaufen werden. Der kürze halber ist nur dieser Algorithmus und nicht das ganze Programm aufgelistet.
Nachfolgend sind die Regeln sowie alle notwendigen Grundwerte erläutert:

Start: *T*
(Startwert der durch die Grundstruktur des Astes substituiert wird.)

Regeln: *T* = *R-[T]++[++L]R[--L]+[T]--T*
(Die Ast Grundstruktur.)
R = *F[++L][--L]F*
(Ast der in der Mitte ein Blatt links und rechts hat.)
L = *[+FX-FX-FX+FX | +FX-FX-FX | +FXFX]*
(Blatt)
FX *FX*
F *FF*

Winkel: 37 Grad

```
/*************************************************************************************/
public void turtle(Graphics g2D, String instruction, int depth)
{
	if(depth == 0) return;
	depth -= 1;

	Point aMark = new Point(0, 0);
	double directionMark = 0;

	for(int i=0; i<instruction.length(); i++)
	{
		char c = instruction.charAt(i);

		// produktionsregel iterieren, solange tiefe nicht erreicht ist
		if(c == 'T') turtle(g2D, "R-[T]++[++L]R[--L]+[T]--T", depth);
		else if(c=='R') turtle(g2D, produktion, depth);
		else if(c=='L') turtle(g2D,"[+FX-FX-FX+|+FX-FX-FX|+FXFX]", depth);
		else if(c=='F') // schritt vorwärts
		{
			if(i+1<instruction.length() && instruction.charAt(i+1)=='X')
			{
				turtle(g2D, "FX", depth);
			}
			else turtle(g2D, "FF", depth);

			// zeichne ab a in richt. direction einen step der länge lengthF
			if(depth == 0)
			{
				double rad = 2*Math.PI/360*direction;  // grad -> radiant
				int p = (int)(lengthF * Math.cos(rad));
				int q = (int)(lengthF * Math.sin(rad));

				b = new Point(a.x+p, a.y+q);

				// farben blätter und stamm
				if(i+1<instruction.length() && instruction.charAt(i+1) == 'X')
				{
					g2D.setColor(new Color(100, 150, 100));
				}
				else g2D.setColor(new Color(120, 0, 0));

				g2D.drawLine(a.x, a.y, b.x, b.y);
				a=b;                                    // Neuer Startpunkt
			}
		}
		else if(c=='+'){ direction += rotation;                    }
		else if(c=='-'){ direction -= rotation;                    }
		else if(c=='|'){ direction += 180;                         }
		else if(c=='['){ aMark = a;  directionMark = direction;    }
		else if(c==']'){ a = aMark;  direction = directionMark;    }
	}
}
/*************************************************************************************/
```

3.3 Die Technik

Aus Gründen der Plattform Unabhängigkeit und der einfach und sauber zu programmierenden GUI (Graphical User Inteface) Elemente wie Buttons, Scrollbars und Bereiche der grafischen Ausgabe ist die Wahl der Sprache in der dieses Projekt erstellt wird auf Java gefallen.
Hierbei wurde massiv auf die Funktionen des AWT (Abstract WIndow Toolkit) zurückgegriffen. Auf die Benutzung von „Swing“ wurde gänzlich verzichtet.
Das Programm liegt in zwei Versionen vor. Zum einen als Anwendung (im Rahmen der Entwicklung wurde ausschließlich als Anwendung (Application) kompiliert.), und zu anderen als Applet welches in meiner Homepage eingebettet ist.
Erreichbar unter folgender Adresse : www.tolschock.de/java

3.4 Hilfsmittel

Linux Mandrake 9.1

- Kate (zur Bearbeitung des Quelltextes)
- Shell (zum Ausführen und Testen der Anwendung)
- Java 2.0 SDK

Windows XP

- Microsoft Word 2002 (für die Dokumentation)
- Microsoft Paint (für die Bilder der Dokumentation)
- Adobe Photoshop 6.0 (für die Bilder der Dokumentation)
- PS-FTP (FTP Client zum Datei Transfer ins Internet)
-

Sonstiges

- HTML (zur Einbettung in meine Homepage)
- Hauptsächlich Recherche im WWW

3.5 Quellen:

[1] **„Chaosforschung“: Fraktale – Chaos – Ordnung** (PDF)
Jürgen Giesen.

[2] **Fraktale biologische Strukturen – Chaos und Ordung im Organismus** (PDF)
Manfred Sernetz, Institut für Biochemie und Endokrinoloie, Justus Liebig Universität Gießen

[3] **Fraktale** (PDF)
Thomas Peters, www.mathe-seiten.de

[4] **Internet** (Website)
http://olli.informatik.uni-oldenburg.de/lily/LP/flow1/page2.html#flow1Xpage14.html

[5] **Internet** (Website)
http://home.wtal.de/schwebin/lsys/lsystem.htm

[6] **Internet** (Website) Dr. Heinz Seiringer
http://www.naturewizards.com