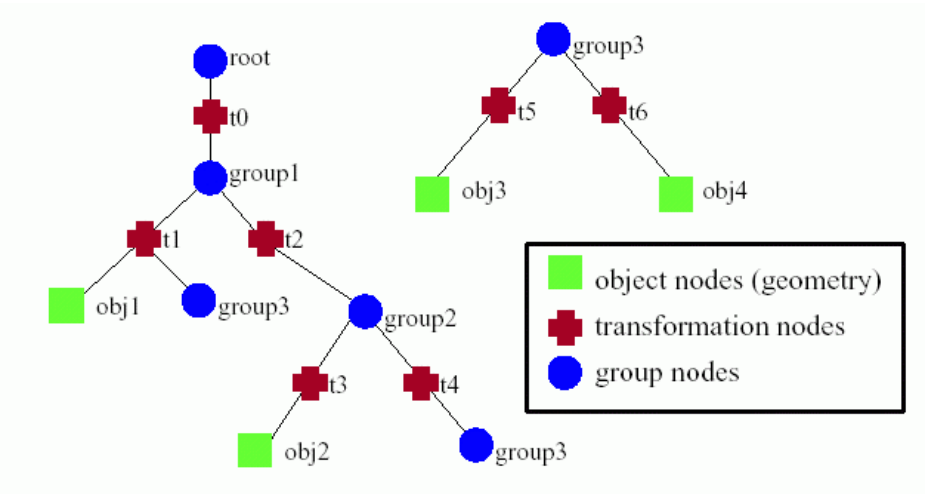
Erinnerung an Kapitel 6:
Szenengraphen als Möglichkeit der Modellierung komplexer Strukturen (vgl. Übung, VRML)



Kreuze: Transformationsknoten,
Transformation wird auf alle darunterliegenden Gruppen und Objekte angewandt
Kreise: Gruppenknoten (bzw. **`children`**-Feld in VRML)
Quadrate: Primitivobjekte

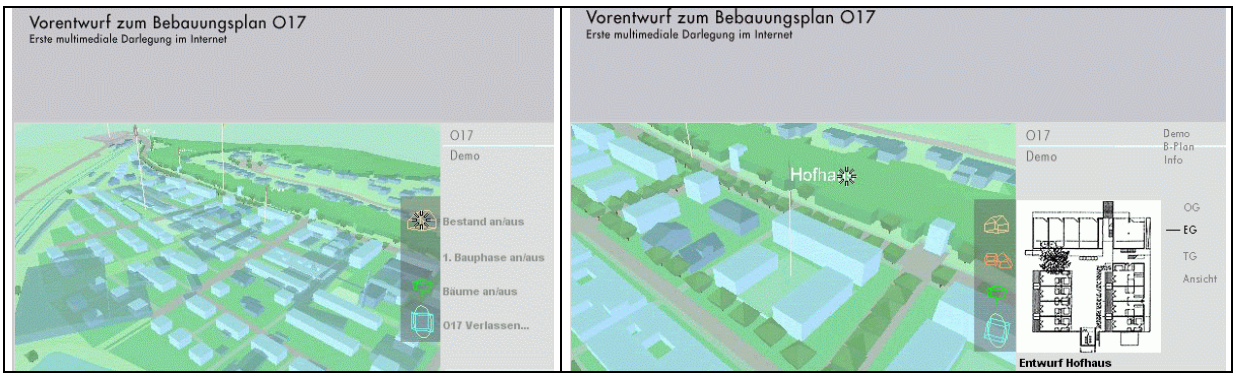
Realisierung geschachtelter **`Transform`**-Knoten in VRML:

```
Transform
  {
  center ...
  scale ...
  rotation ...
  translation ...
  children
    [
    Transform
      {
      center ...
      scale ...
      rotation ...
      translation ...
      children
        [   .......
            ...... [ Shape    # Primitiv-Knoten
                       { appearance ...
                         geometry ....
```

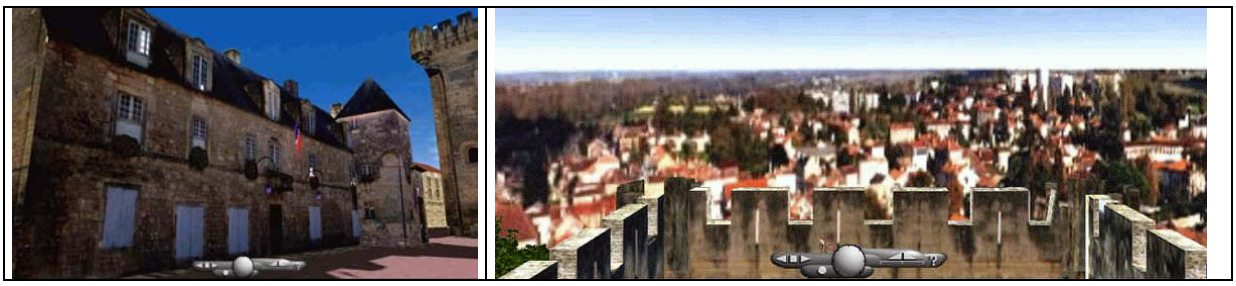
Anwendungen:

- Gebäudemodellierung, Architektur

Beispiel: interaktiver Bebauungsplan mit VRML
(Pilotprojekt in Darmstadt)
http://www.citimage.de/o17



- virtuelle Museumsbesuche, Stadtbesuche, Landschaften...
http://www.villes-3d.com



- medizinische Visualisierung
- Molekülmodelle
- astronomische Modelle
- Spiele

...

Lindenmayer-Systeme (L-Systeme)

Ansatz aus der Theorie formaler Grammatiken

analog zu Chomsky-Grammatiken (regulär, kontextfrei, kontext-sensitiv etc.)
aber: in jedem Ableitungsschritt *parallele* Ersetzung aller Zeichen, auf die eine Regel anwendbar ist

von Aristid Lindenmayer (Botaniker) 1968 zur Modellierung des Wachstums von fadenförmigen Algen eingeführt (Verwandtschaft zum Ansatz der *zellulären Automaten*)

- Verwendung hauptsächlich für Vegetationsmodelle
  aber auch: Webmuster, Gebäude, Roboter, Tiere (Vermehrung, Nahrungsaufnahme), Melodien
- 2D- und 3D-Varianten
- Emulation von IFS möglich
- volle Mächtigkeit einer Programmiersprache
- *dynamische* Simulationen ($\Rightarrow$ Möglichkeit der Animation)
- Anbindung physikalisch oder biologisch begründeter Simulationsmodelle möglich (Wechselwirkung der Struktur und Funktion von Pflanzen)

Fortgeschrittene Softwaresysteme zur Umsetzung:
- cpfg / LStudio `http://www.cpsc.ucalgary.ca/projects/bmv/index.html`
- Grogra `http://www.uni-forst.gwdg.de/~wkurth/grogra.html`
- Graphtal
- LParser `http://www.xs4all.nl/~ljlapre/`

L-Systeme arbeiten *stringbasiert*.
Erweiterungen: Wörter aus parametrisierten Zeichen (Modulen); Graph-Grammatiken; *map*-L-Systeme und *cellwork*-L-Systeme.

Grundversion gut für alle Strukturen mit *lokal 1-dimensionalem Grundgerüst* (Verzweigungssysteme).

Definition:
Ein (kontextfreies, nichtparametrisches) *L-System* ist ein Tripel (Σ, α, *R*), darin ist

- Σ eine nichtleere Menge von Zeichen (das *Alphabet*),
- α ein Element von Σ^*, das *Startwort* oder *Axiom*,
- *R* eine nichtleere Teilmenge von $\Sigma \times \Sigma^*$, die Menge der *Produktionsregeln* (generative Regeln).

Ein *Ableitungsschritt* eines Wortes $\beta \in \Sigma^*$ besteht aus der Ersetzung aller Zeichen in β, die in linken Regelseiten von *R* vorkommen, durch die entsprechenden rechten Regelseiten. Man vereinbart: Zeichen, auf die keine Regeln anwenbar sind, werden unverändert übernommen.

Ergebnis zunächst nur:
Ableitungskette von Wörtern, die sich durch iterierte Anwendung des *rewriting*-Vorgangs aus dem Startwort ergeben.

$$\alpha \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \sigma_3 \rightarrow ....$$

In der theoretischen Informatik betrachtet man die Sprachen, die von den so erhältlichen σ_i gebildet werden, ihre Abschlusseigenschaften, Relationen zur Chomsky-Hierarchie...

was für die Grafik noch fehlt:
eine Semantik (= *geometrische Interpretation*)

füge zu obiger Def. hinzu:
eine Abbildung, die jedem Wort aus Σ^* eine Teilmenge des R^3 zuordnet

dann: "interpretierte" L-System-Abarbeitung

$$\begin{array}{ccccccc} \alpha & \rightarrow & \sigma_1 & \rightarrow & \sigma_2 & \rightarrow & \sigma_3 \rightarrow .... \\ & & \downarrow & & \downarrow & & \downarrow \\ & & S_1 & & S_2 & & S_3 \quad .... \end{array}$$

S_1, S_2, S_3, ... können als Generationen oder als Entwicklungsstufen eines belebten Objekts (Pflanze, Biotop...) interpretiert werden.

Als Interpretationsabbildung wird meistens gewählt:

Turtle geometry ("Schildkrötengeometrie")

befehlsgesteuertes, lokales Navigieren im 2D- oder 3D-Raum

- Abelson & diSessa 1982
- vgl. Sprache "LOGO"

Verwandtschaft des Ansatzes zu "Kettencode-Bildsprachen"
(dort aber meist Befehle mit globaler Bedeutung: "gehe nach unten", "gehe nach rechts" etc.)
in turtle geometry in "Reinform" nur lokale Information und Orientierung
aber: "verwässert" durch Zusatzbefehle, z.B. für Tropismen (Geotropismus, Heliotropismus...): Ausrichtung der Orientierung an festen Richtungen oder Objekten

"Turtle": Zeichen- oder Konstruktionsgerät (virtuell)
vgl. virtual pen in PostScript

- speichert (grafische und nicht-grafische) Informationen
- mit Stack assoziiert
- aktueller Zustand enthält z.B. Information über aktuelle Liniendicke, Schrittweite, Farbe, weitere Eigenschaften des als nächstes zu konstruierenden Objekts

Befehle (Auswahl):

F	"Forward", mit Konstruktion eines Elements (Linienstück, Segment, Internodium einer Pflanze...) benutzt wird die aktuelle Schrittweite für die Länge
f	forward ohne Konstruktion (move-Befehl)
L(x)	ändere die aktuelle Schrittweite (Länge) zu *x*
L+(x)	inkrementiere die aktuelle Schrittweite um *x*
L*(x)	multipliziere die aktuelle Schrittweite mit *x*
D(x), D+(x), D*(x)	analog für die aktuelle Dicke
RU(45)	Drehung der *turtle* um die "up"-Achse um 45°

RL(...), RH(...) analog um "left" und "head"-Achse

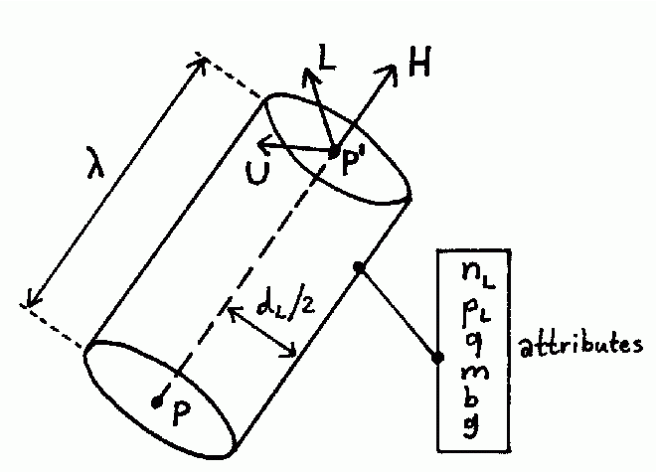
up-, *left*- und *head*-Achse bilden ein orthonormales Rechtssystem, das von der turtle mitgeführt wird

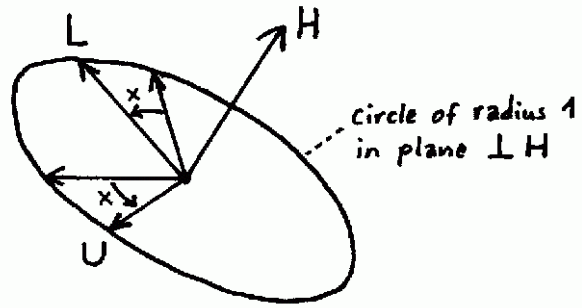
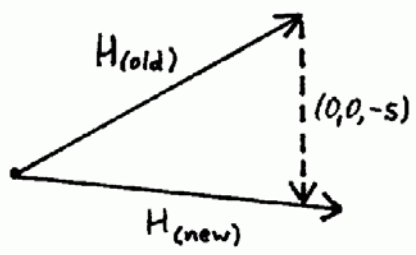
RV(x) Rotation "nach unten" mit durch *x* vorgegebener Stärke

+, – Abkürzung für **RU**(φ) und **RU**(–φ) mit einem festen Winkel φ

(Prusinkiewicz und Lindenmayer verwenden noch weitere Kurzformen für Befehle)

Wirkung des **F**-Befehls (aktuelle Schrittweite ist λ):



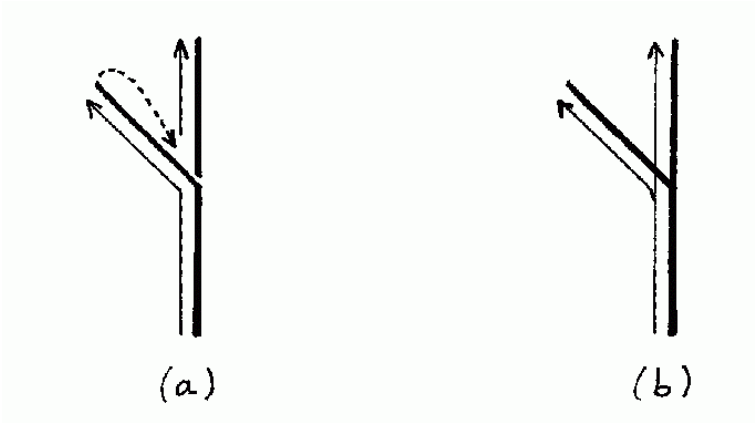
Wirkung von **RH**:	Wirkung von **RV** auf die *head*-(Vorwärts-) Richtung:
	

Strings aus diesen Symbolen werden sequentiell abgearbeitet.

Verzweigungen: Realisierung mit Stack-Befehlen

[	lege aktuellen Zustand auf Stack
]	nimm Zustand vom Stack und mache diesen zum aktuellen Zustand (Ende der Verzweigung)

Interpretation der Klammern sequentiell und parallel möglich (Turtle steht z.B. für pflanzliches, teilungsaktives Gewebe = Meristem)



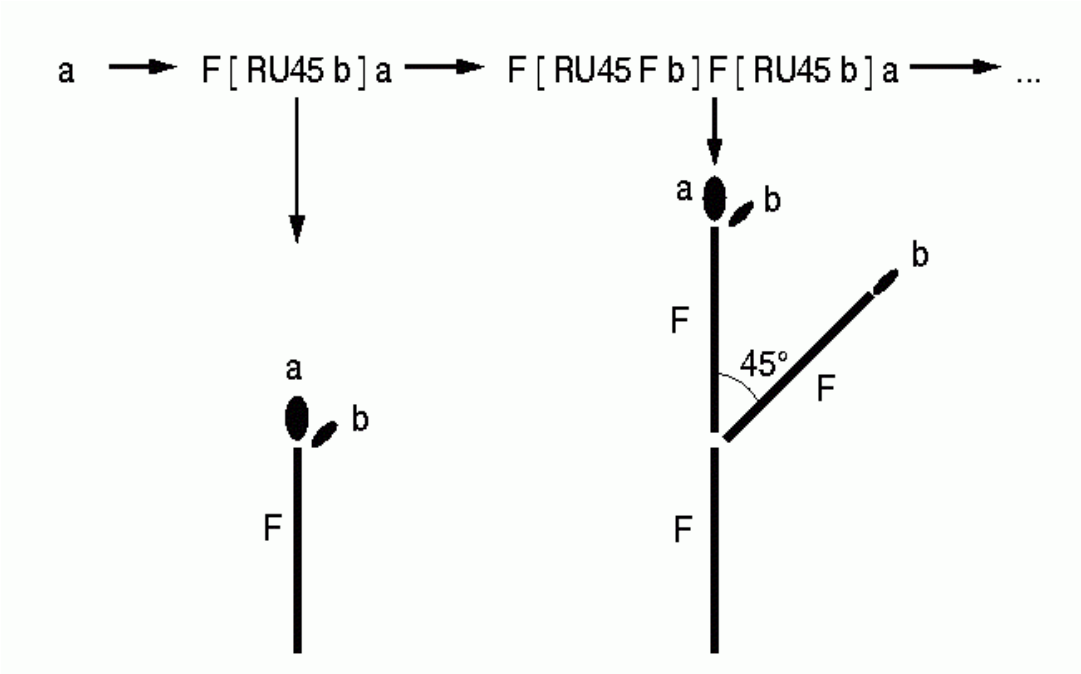
Der Turtle-Befehlsvorrat wird zu einer Untermenge der Symbolmenge Σ des L-Systems.
Zuerst Abarbeitung des L-Systems (String-Erzeugung), dann Interpretation der erzeugten Wörter durch die Turtle. Symbole, die nicht Turtle-Befehle sind, werden von der Turtle ignoriert.

Beispiel:

Regeln

a → F [RU45 b] a,
b → F b

Startwort **a**



(**a** und **b** werden normalerweise nicht geometrisch interpretiert.)

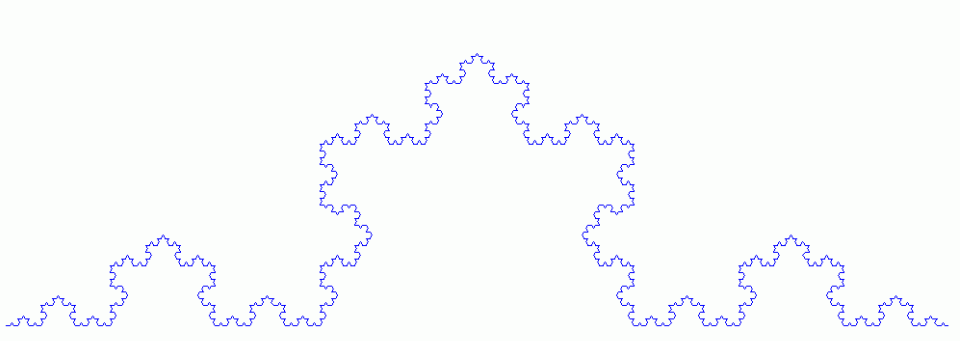
Weitere Beispiele:

Koch-Kurve:

```
\angle 60,
* → RU90 a F,
a → a L*0.3333,  /* Skalierung */
F → F – F + + F – F
```

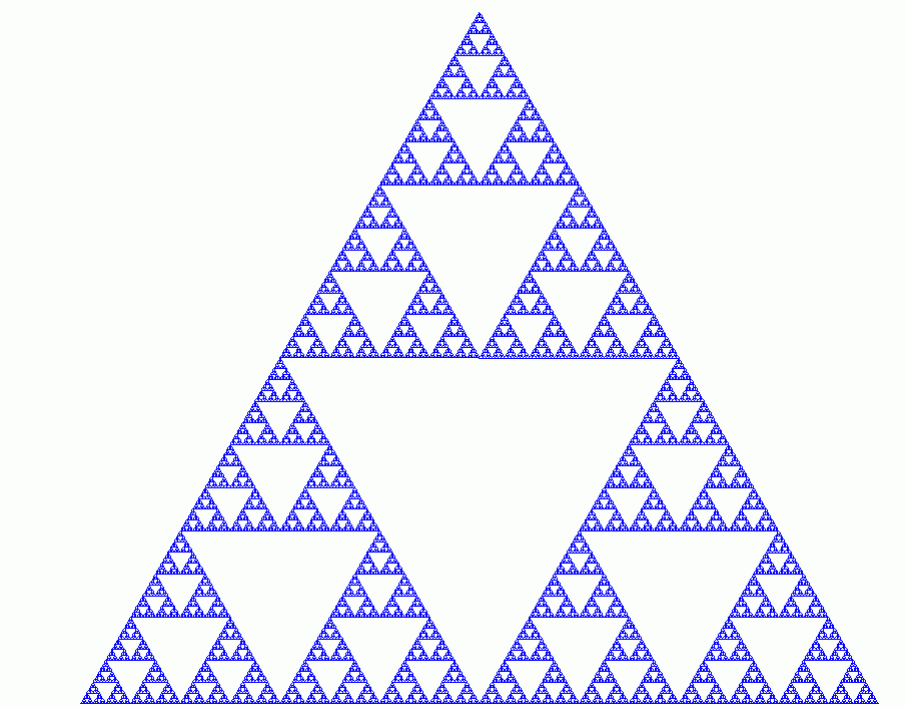
jedes Linienstück wird durch 4 neue Linienstücke ersetzt (3. Regel); Skalierung durch Hilfssymbol, welches sich in jedem Schritt reproduziert (2. Regel).
Das Startwort ist hier " ***** ".
"**\angle**" spezifiziert den Winkel für "**+**" und "**–**".

Ausgabe nach 6 Schritten:



Sierpinski-Dreieck (Realisierung als geschlossene Kurve, Verwendung von Hilfssymbol **x** für Insertion des inneren Dreiecks):

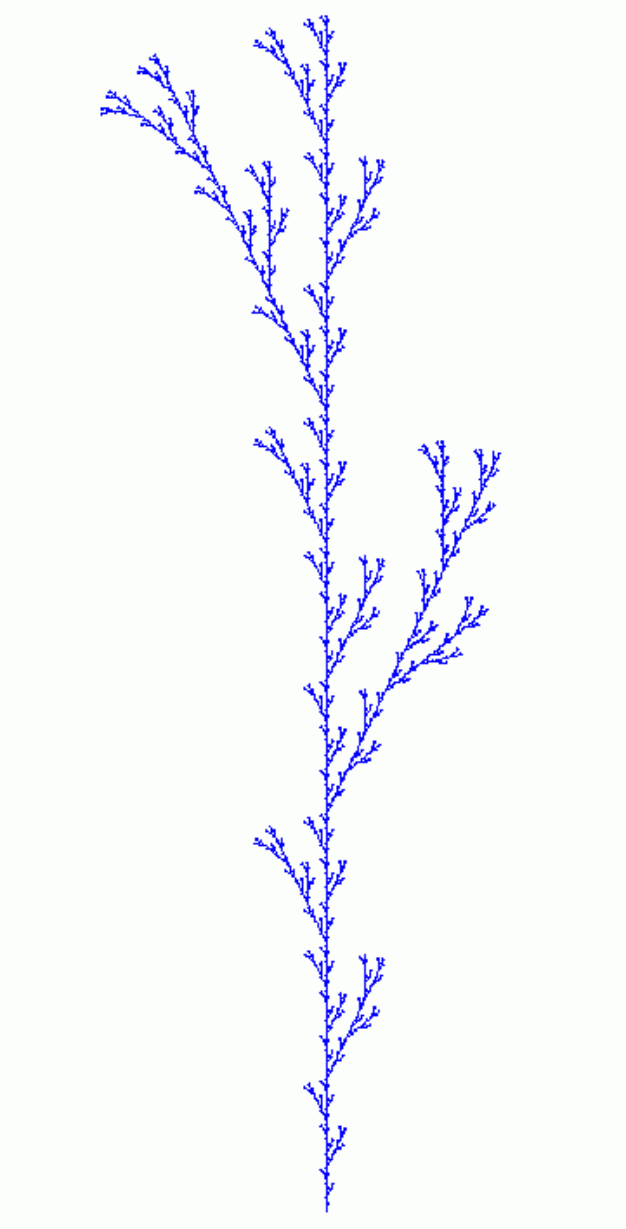
```
\angle 60,
* → RU90 b F x F – - F F – – F F,
F → F F,
x → – – F x F + + F x F + + F x F – –,
b → b L*0.5
```



Verzweigung, "Pseudo-Pflanze":

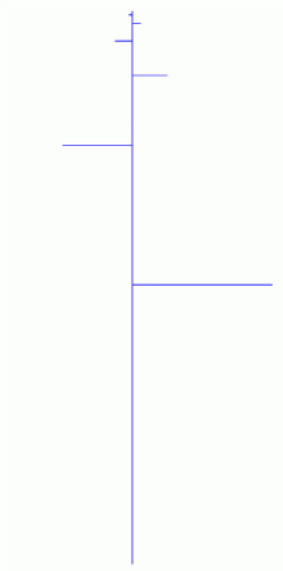
\angle 25.7,
F → F [+ F] F [– F] F

Ergebnis nach 7 Schritten:



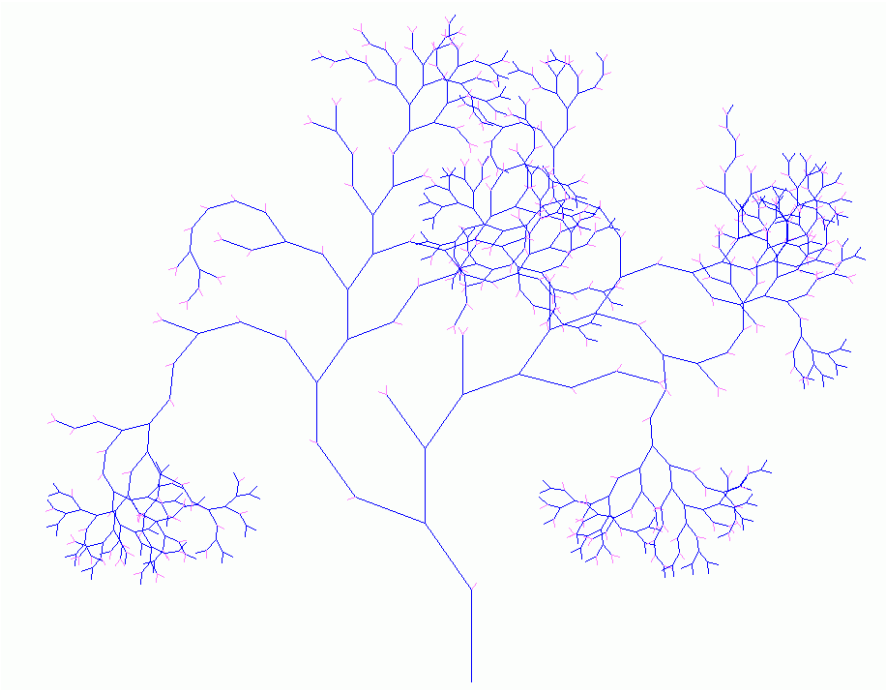
Verzweigung, alternierende Zweigstellung und Verkürzung:

```
* → F a,
a → L*0.5 [ RU90 F ] F RH180 a
```



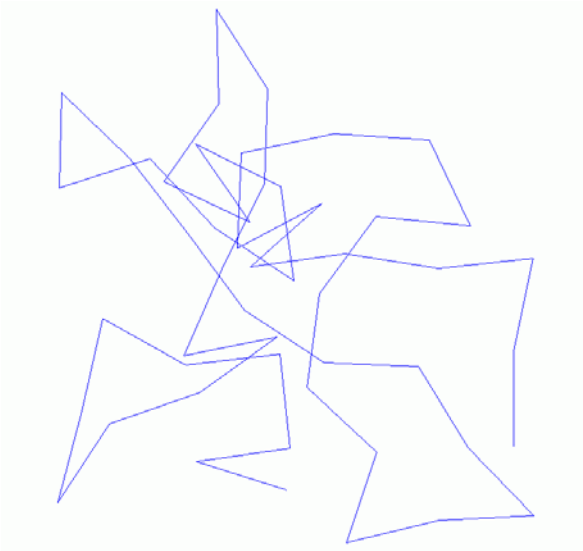
Verzweigung, Absterbemöglichkeit (stochastisches L-System):

```
* → F L*0.9 [ RU35 * ] RU-35 * ?0.65,
* → P2 L10 F ?0.35
```



Irrflug (Brownsche Bewegung in 2D), Verwendung einer gleichverteilten Zufallsvariablen:

```
\var x uniform 0 360,
* → F a,
a → RU(x) F a
```



Erweiterung des Konzepts:
Lasse reellwertige Parameter nicht nur bei Turtle-Kommandos wie "**RU45**" zu, sondern bei allen Symbolen
→ *parametrische L-Systeme*, operieren auf "Moduln" statt auf Wörtern
beliebig lange, endliche Parameterlisten
Parameter werden bei Regel-Matching mit Werten belegt

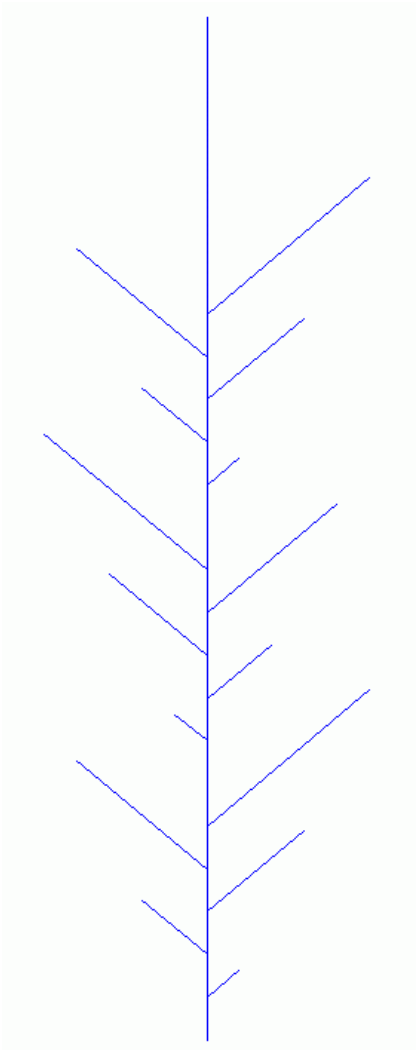
Beispiel:
Regel **`a(x, y)`** → **`F(x^3+10) b(2*y)`**
vorliegendes Wort "**`a(2, 3)`**"
nach einer Regelanwendung: **`F(18) b(6)`**

Parameter können in Konditionen abgeprüft werden (Konditionen mit C-Syntax):
`(x >= 17 && y != 0) a(x, y)` → ....

Beispiele:

Verwendung des Wiederholungsoperators "&" und von parametrisierten Symbolen
Länge des abzweigenden Astes abhängig von Position (Bauprinzip bei vielen Gehölzen!) – "Akrotonie":

```
\var i index,
* → a(5),
a(n)→ &(n) < F [ RU50 lat(i) ] RH180 > F a(n),
lat(j) → L+(j*100) F
```



Beispiel für ein L-System mit mehreren Hilfssymbolen, die unterschiedliche Meristem-Typen einer Pflanze repräsentieren:

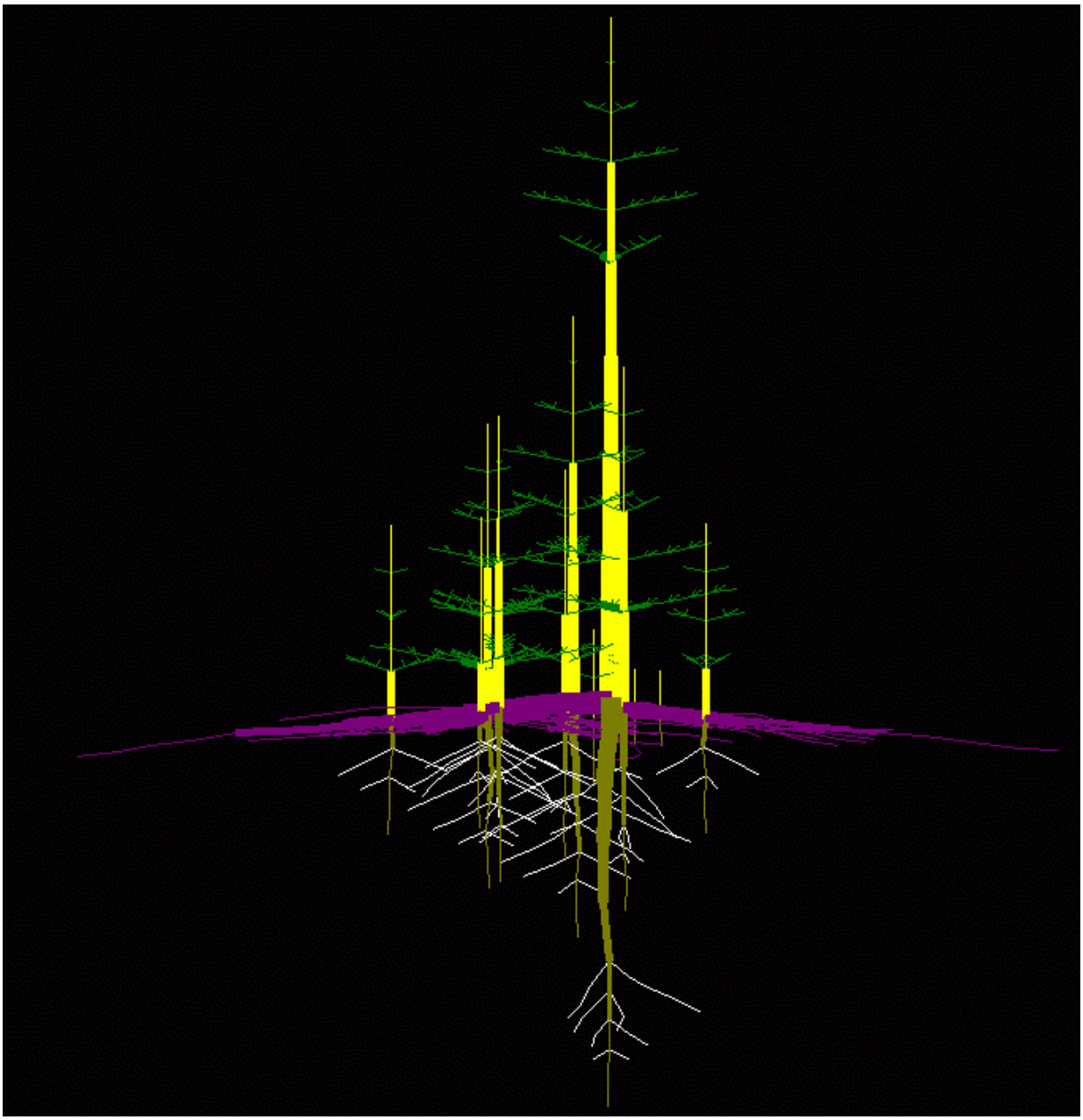
```
\var x0 uniform 0 360,
\var x1 normal 0 15,
\var x2 uniform -10 0,
\var x3 uniform -25 25,

* # [ P14 t0 ] P4 &6 < [ RH(x0) P5 ul' ] > L*0.6 u0,
t0 # xt F D RH137.5 [ RL80 L*0.5 P2 k(1) s1 ]
     [ RH180 RL80 L*0.5 P2 k(1) s1 ] t0,
s1 # xs F D [ RH25 RU60 $ L*0.7 s2 ] [ RH-25 RU-60 $ L*0.7 s2 ] s1,
s2 # xs F D,
xt # xt D+3,
xs # xs D+2,
(t < 6) k(t) # k(t+1),
(t = 6) k(t) # %,
u0 # RG RH(x0) RU(x1) xt F D
     [ L*1.1 k(1) P15 u1 ] [ L*1.1 k(1) P15 u1' ] u0,
ul' # ul,
ul # RG RL90 RL(x2) RU(x3) xs F D a1 ul,
u1 # u2,
u1' # u2',
u2 # RG RL70 RL(x2) RU(x3) xs F D u2,
u2' # RG RH180 RL70 RL(x2) RU(x3) xs F D u2,
a1 # a2,
a2 # a3,
a3 # [ RG RU180 * ] ?0.2,
a3 # z ?0.8
```

Beachte:

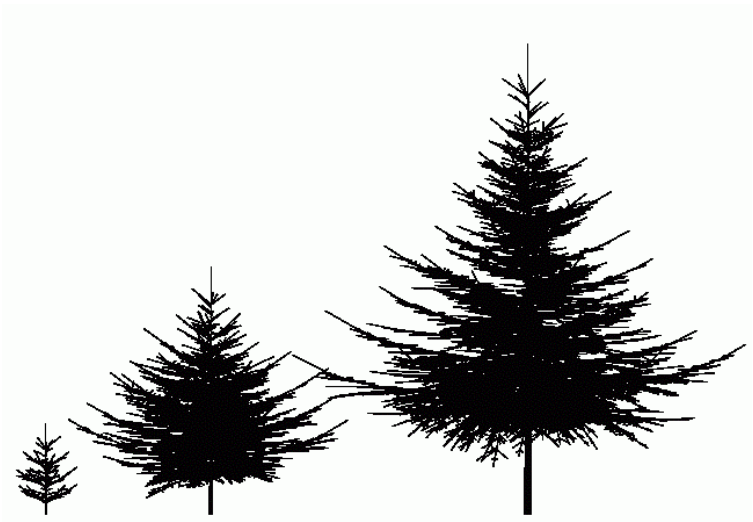
- **`k(t)`** als "Uhrensymbol" (**`t`** wird in jedem Schritt um 1 inkrementiert)
- Verwendung eines "Cut-Operators" **`%`**
- Wiederauftauchen des Startsymbols "**`*`**" in der vorletzten Regel: "Reiteration", d.h. Reproduktion des gesamten Bauplans der Pflanze (in diesem Fall aus Wurzel-Meristemen, "Wurzelbrut")

Austriebs- und Absterbewahrscheinlichkeiten können in der Praxis aus botanischen Messungen entnommen werden

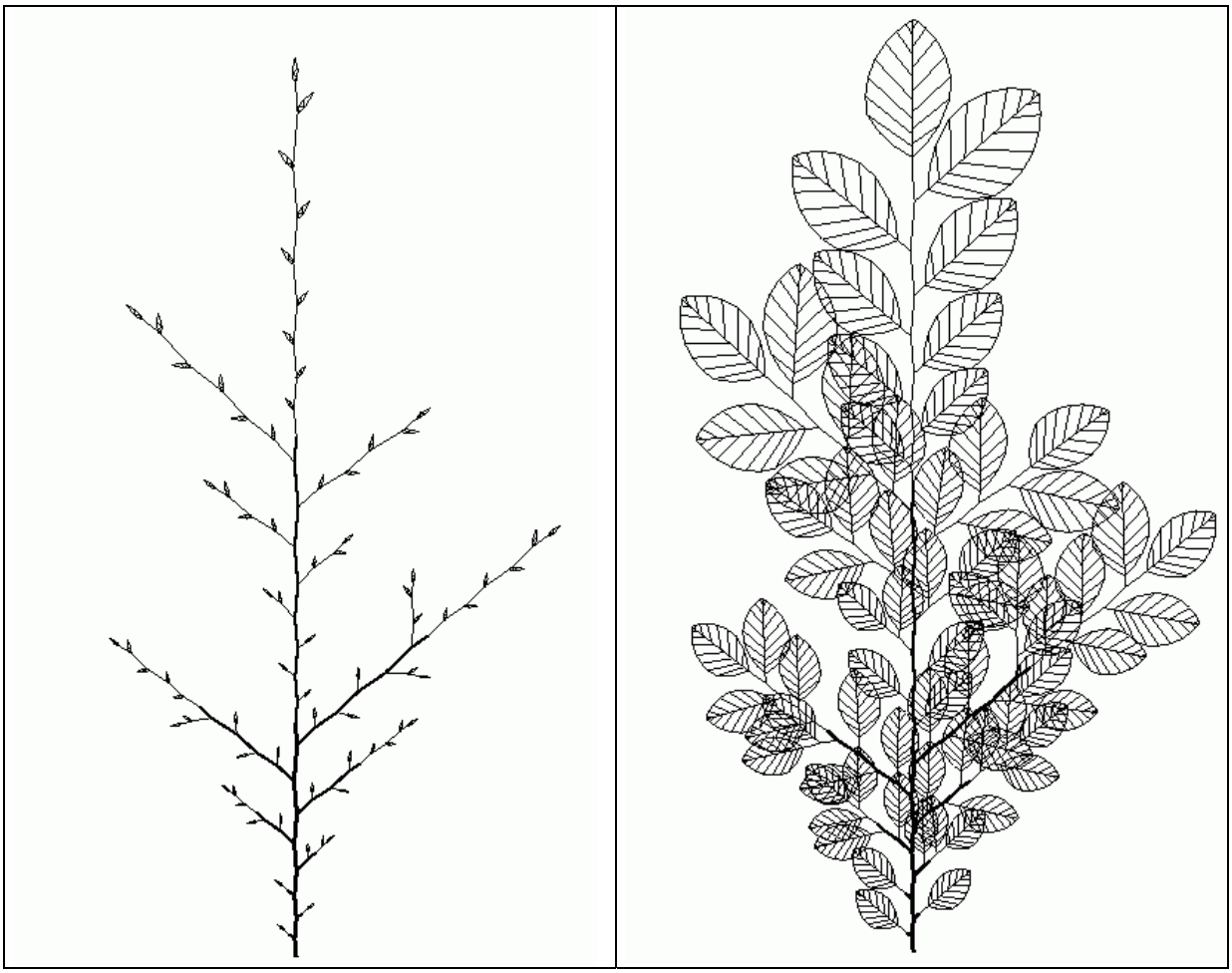
Ergebnis nach 15 Schritten:



Beispiel Fichte (basierend auf Messungen am realen Objekt):



Beispiel Buchenzweige:



Nützliche Erweiterung des Formalismus:
Einführung einer zusätzlichen Regelmenge
"Interpretationsregeln"
wirken nicht auf den String der nächsten Generation
Verwendung zum Zeichnen: Vorstufe der Turtle-Interpretation
z.B. für die Knospen und Blätter beim obigen Buchenzweig

$$\begin{array}{lllll} \alpha \rightarrow \sigma_1 & \rightarrow \sigma_2 & \rightarrow \sigma_3 & \rightarrow .... \\ \quad \Downarrow & \Downarrow & \Downarrow & \\ \quad \sigma_1' & \sigma_2' & \sigma_3' & .... \\ \quad \downarrow & \downarrow & \downarrow & \\ \quad S_1 & S_2 & S_3 & .... \end{array}$$

Doppelpfeile ($\Downarrow$) bedeuten Anwendung der Interpretationsregeln

Nachteil der bisher vorgestellten L-Systeme:
Kontrolle nur durch Vorgänger-Symbol ("lineage control") oder stochastisch

⇒ fehlende Interaktion innerhalb des modellierten Objekts oder mit der Umwelt

⇒ Determinismus oder stochastische Modelle ohne kausale Komponenten

Abhilfe: Einführung von *Sensitivität* bei der Regelanwendung

(a) Kontextsensitivität

(schon altes Konzept, Beispiele bereits bei Lindenmayer...):

Abhängigkeit einer Regelanwendung vom linken und / oder rechten Kontext im String:
leftcontext **<** **a** **>** *rightcontext* $\rightarrow$ β.
stringbasiert!

Verwendung:

- Weiterleitung von Signalen innerhalb der modellierten Struktur

  s < a > → a s,

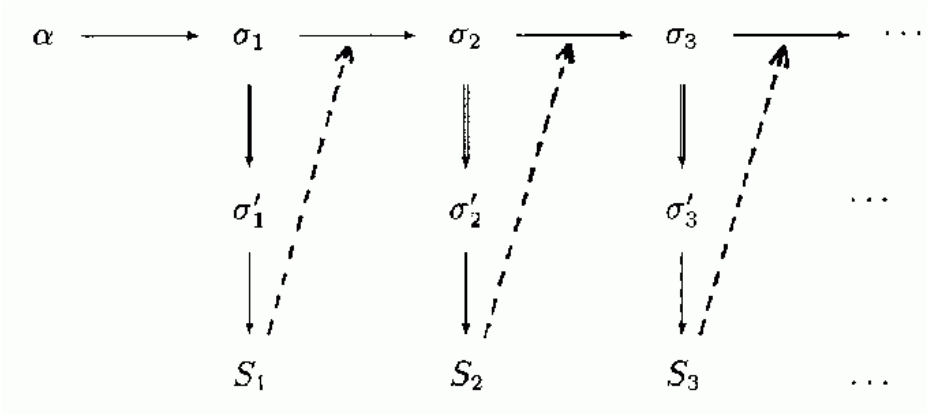
  s → **,** /* leeres Wort */
- Konzentration von Substanzen (Hormonen)
- Bewegung von Objekten (z.B. Insekten auf der Pflanze)

Beispiel: Entwicklung von Blütenständen häufig hormonal gesteuert

(b) globale Sensitivität
(auch: *environmentally sensitive L-systems*)

- Kommunikation mit der Umgebung über spezielle Kommunikationsmodule oder über sensitive Funktionen
- Regelanwendung hängt (potentiell) von der gesamten, aktuell vorhandenen Struktur im Objektraum (und von eventuellen externen Eingriffen) ab (nicht nur von der Stringrepräsentation)
- Schnittstelle zu physikalisch oder biologisch basierten Simulationsmodellen

Prinzip:



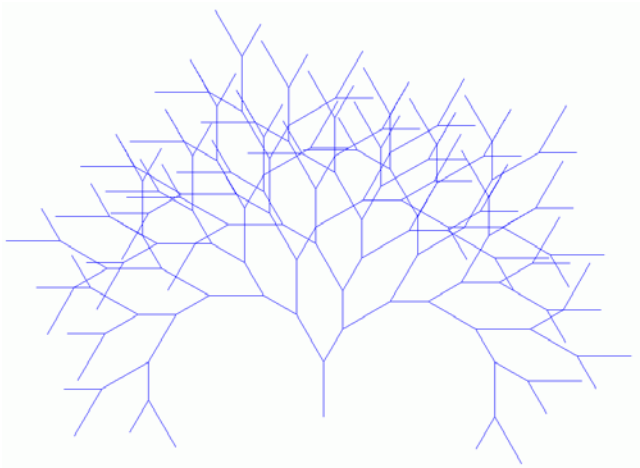
einfaches Beispiel:

Nichtsensitives und global sensitives L-System im Vergleich

nichtsensitive, dichotome Verzweigung:

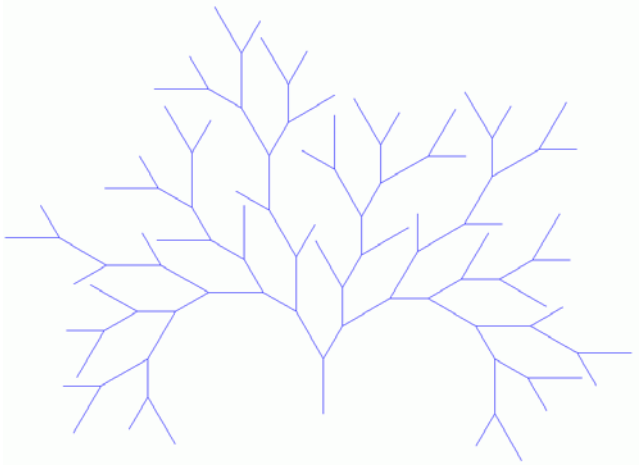
```
\axiom a 1–8,
\angle 30,
a → RH180 F100 [ – b ] + a,
b → RH180 F70 [ – b ] + a
```

Ergebnis:



sensitive Verzweigung mit Abhängigkeit vom Abstand zum nächsten Nachbar-Element (im Objektraum!):

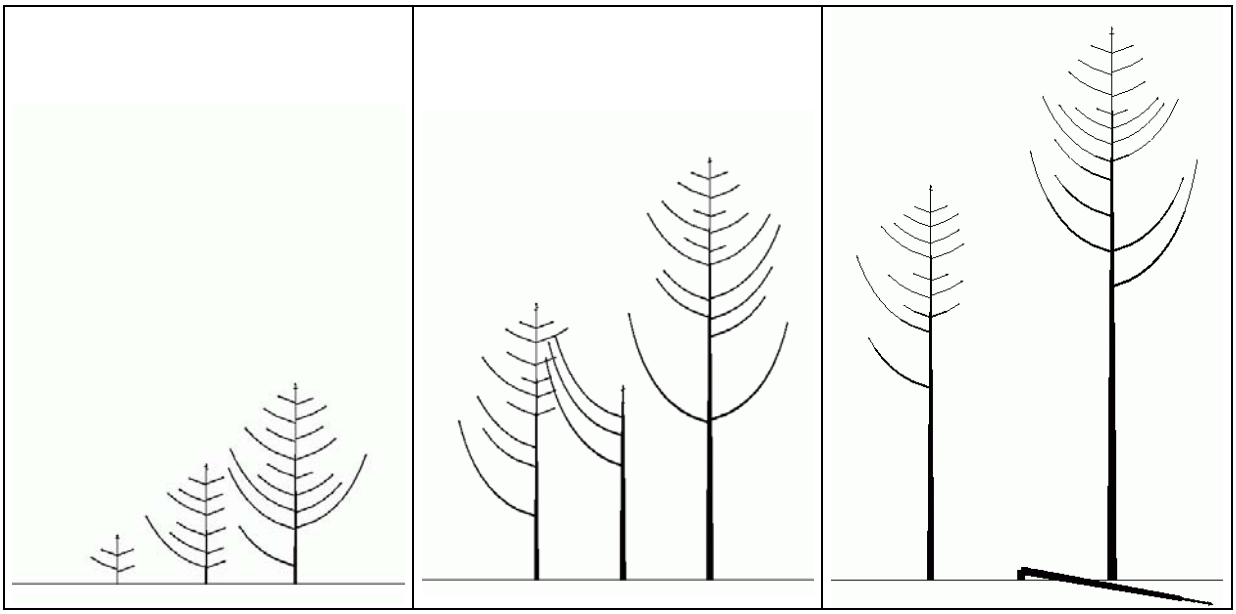
```
\axiom a 1-8,
\angle 30,
\var f function 2 1,
(f(1) > 60) a → RH180 F100 [ - b ] + a,
(f(1) > 60) b → RH180 F70 [ - b ] + a
```



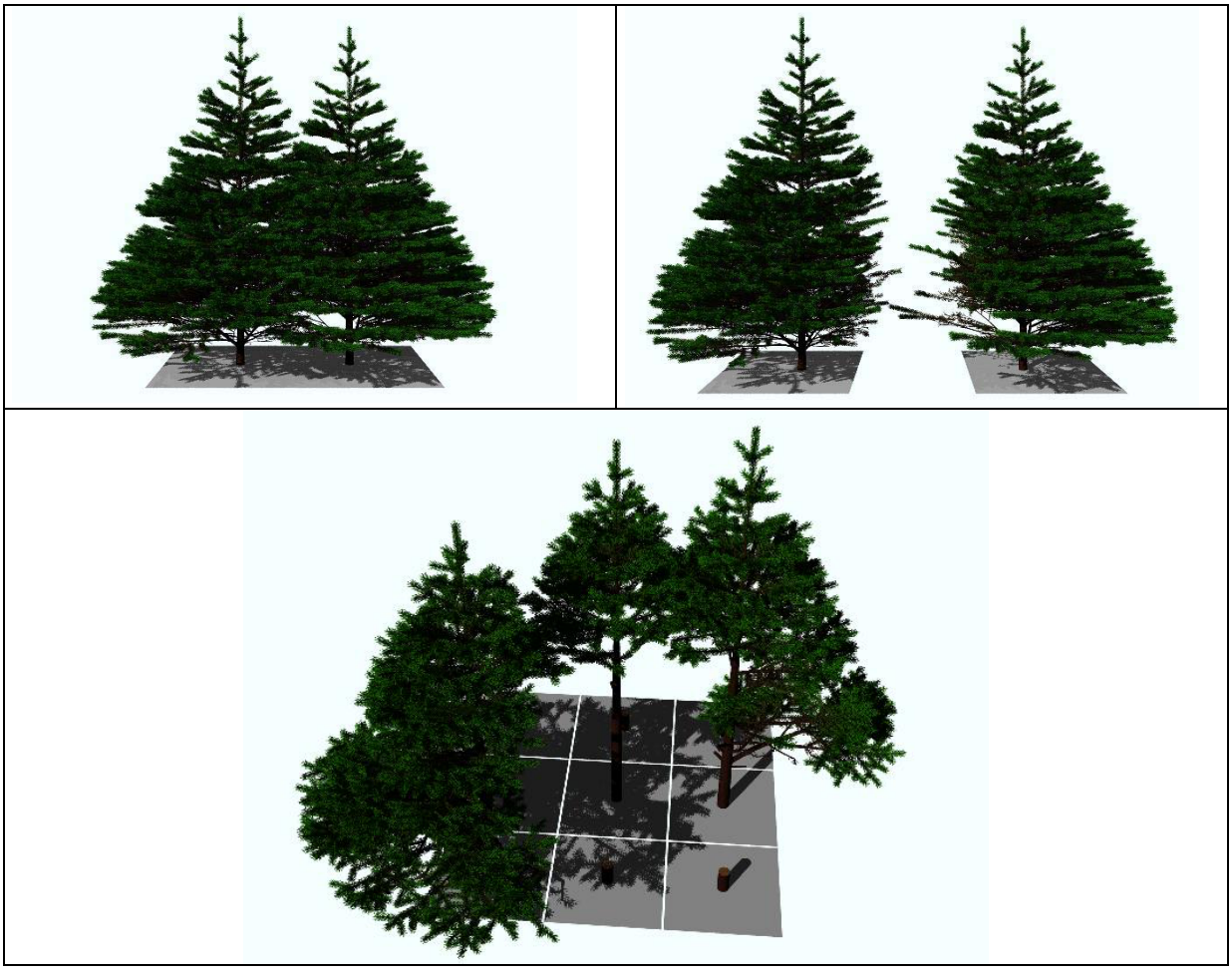
Anwendung in der Pflanzenmodellierung:

- Dichteabhängigkeit des Wachstums
- Einfluss des Neigungswinkels eines Astes auf den Neuaustrieb
- Einfluss der Beschattung
- Wechselwirkung mit Herbivoren (Tiere)

Beispiel: einfaches Überschattungsmodell (2D)



Anwendung: Wachstum von Bäumen unter Konkurrenz-bedingungen
(P. Prusinkiewicz et al.)



Weitere Ansätze zur Modellierung komplexer Objekte

Prozedurales Modellieren

Prozedur (Generierungsvorschrift) spezifiziert die Objekt-geometrie
("erzeuge regelmäßiges Polyeder mit 20 Seitenflächen", anstatt die Seitenflächen alle einzeln anzugeben)

Verwendung prozeduraler oder objektorientierter Programmiersprachen oder spezieller Softwarewerkzeuge (z.B. MATHEMATICA)

Vorteile:

- Platzersparnis
- Objekt kann ja nach Anforderung dargestellt werden
- wie bei L-Systemen Verknüpfung mit physikal./biol. Simulationsmodellen möglich

Ähnlichkeit zum Szenengraphen-Ansatz (Instanzierung von geometr. Primitiven), aber: hier kann auch die Topologie verändert werden

Beispiel: Hängebrücke

- Prozeduren beschreiben Aufbau der Straße, Seile, Pfeiler etc.
- Teile beeinflussen sich gegenseitig
- Möglichkeit, ein physikalisches Modell der Statik (oder der Schwingungen der Brücke) zu integrieren
- Modellwechsel: Nachts besteht Brücke nur aus Punktlichtern

Partikelsysteme

kleine Partikel als Objekte, Erzeugung und Verhalten zum großen Teil stochastisch bestimmt

Darstellung von Bäumen über Partikelsysteme

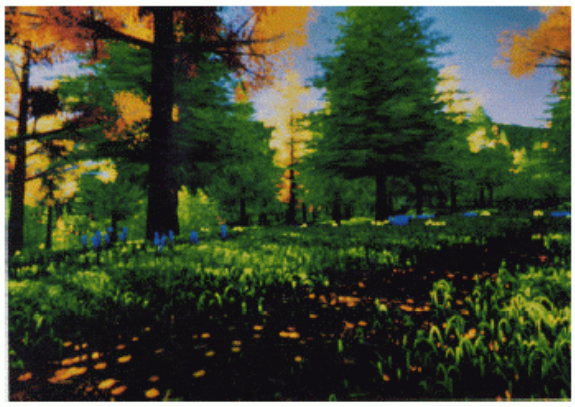
→ visuelle Modelle, für Film

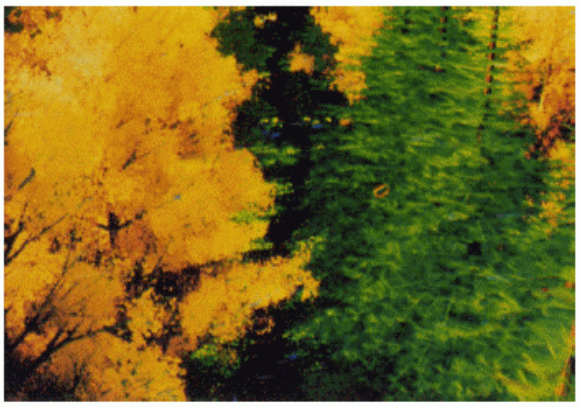
→ primitives rekursives Verzweigungmodell

Postprocessing: Zufälligkeiten werden eingebaut
Blätter: kleine Kugel mit Farbe und Ausrichtung

→ wichtig: korrekte Farben sowie Licht/Schatten

→ Ergebnis trotz botanischer Inkorrektheit brauchbar



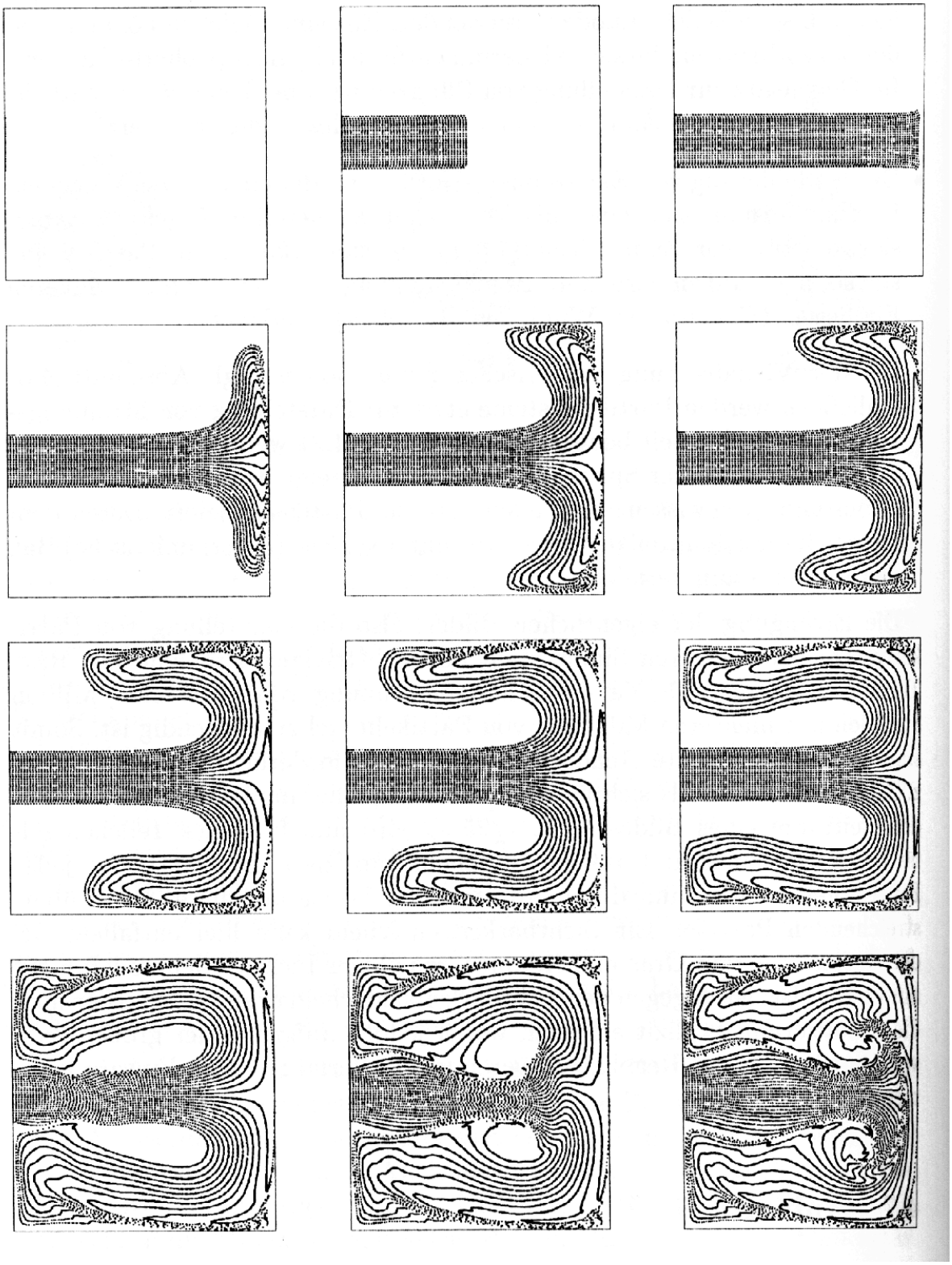


Strukturierte Partikelsysteme nach Reeves und Blau

Verwendung z.B. auch für Modellierung von Flammen, Funkenflug, Nebelschwaden (auch animiert)

Rendering ist bei großer Zahl von Partikeln nichttriviales Problem!

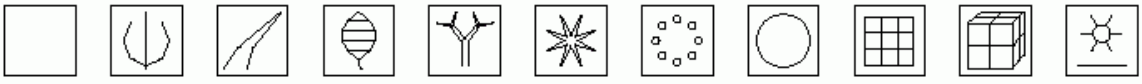
Beispiel: Simulation von Spritzguss mittels Partikelverfolgung (aus Bungartz et al. 1996):



XFROG (Deussen & Lintermann):

- interaktives System zur Pflanzenmodellierung, das sich am Szenengraphen-Konzept orientiert
- botanisch nur schwach fundiert, keine Wachstumsmodellierung
- aber grafisch sehr zufriedenstellende Ergebnisse

Komponenten



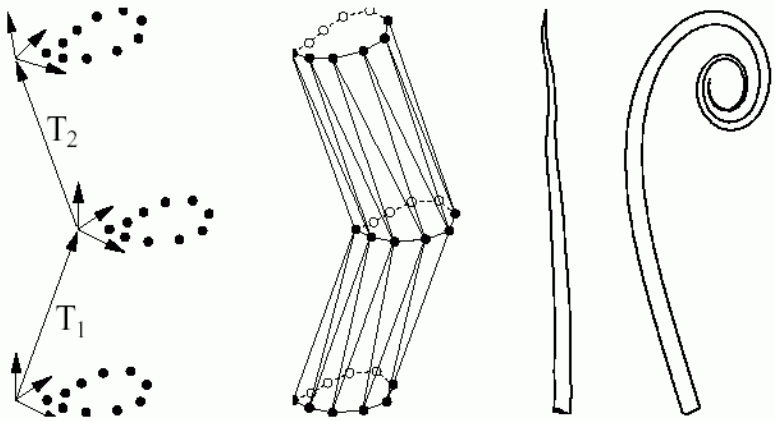
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k)

Geometrieerzeugung: a) Simple, b) Revo, c) Horn, d) Leaf

Multiplikation: e) Tree, f) Hydra, g) Wreath, h) Phiball

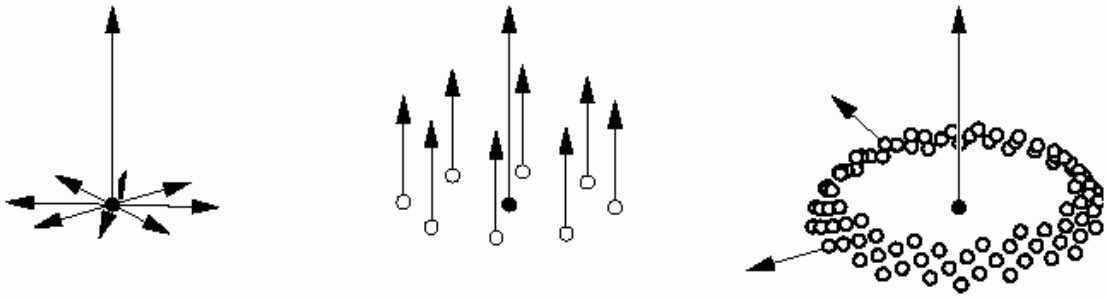
Globale Modellierung: i) FFD, j) Hyperpatch, k) World

Horn/Tree-Geometrie (inspiriert von Todd/Latham):



- hohe Flexibilität

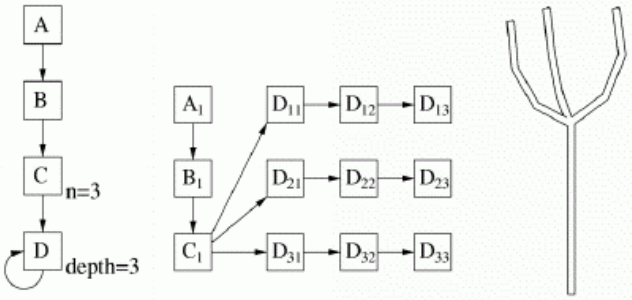
Multiplikation:



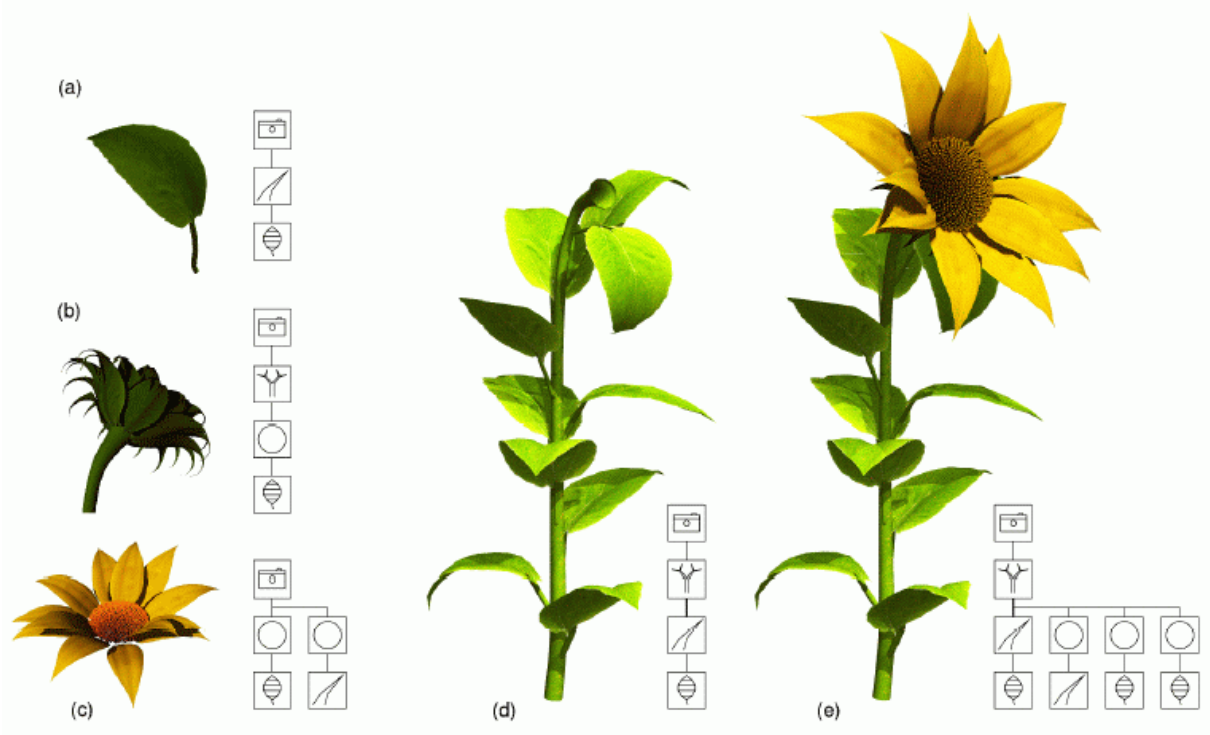
a) Hydra, b) Wreath, c) Phiball

Ein Beispiel

A	Wurzel
B	Geometrieerzeugende Komponente
C	Multiplikator (Anzahl Kopien=3)
D	Geometrieerzeugende Komponente (Rekursionstiefe=3)



Beispiel: Sonnenblume



Beispiel: Rhododendron

Beispiel: Baum