Computergrafik SS 2010
Oliver Vornberger

noch Kapitel 10:
2D-Grafik im Web

Vorlesung vom 11.05.2010

Webseite

```
<HTML>
<HEAD><TITLE>SVG-Beispiel</TITLE></HEAD>
  <BODY
  <H1>SVG-Datei</H1>
  <object type="image/svg+xml" width="80%" height="80%">
    <param name="src" value="basics.svg"/>
    <param name="type" value="image/svg+xml"/>
    <embed  type="image/svg+xml" src="basics.svg"/>
  </object>
  </BODY>
</HTML>
```

~cg/2010/SVG/uebersicht.html

about:config svg enabled true animate nicht in Firefox !

Adobe Flex

- Vorgestellt von Macromedia in 2004
- Heute Version 3.0 von Adobe
- MXML
- Actionscript
- Flex Builder (als Eclipse Plugin)
- Flex Charting Komponenten

Adobe MXML + Actionscript

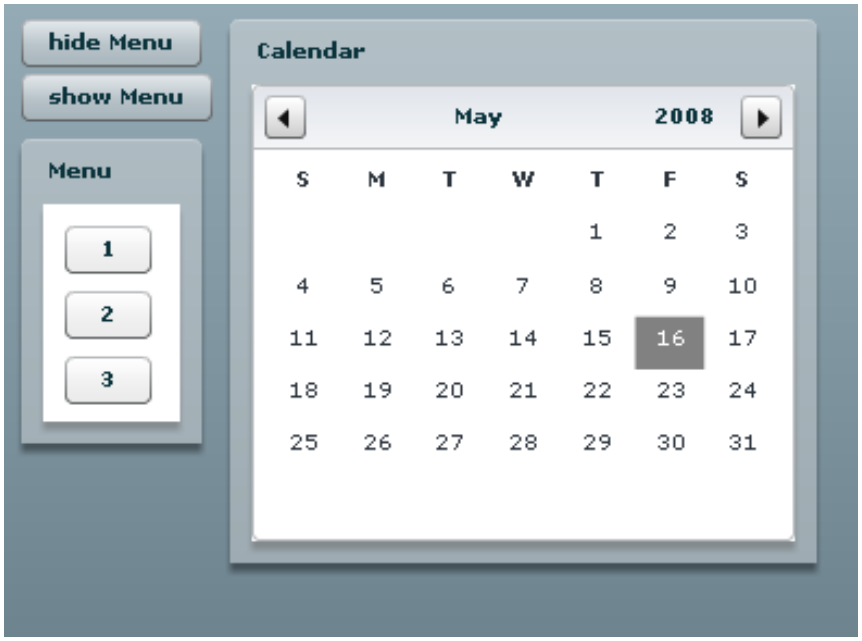
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
   layout="vertical">

  <mx:Script>
    <![CDATA[
    public function copy():void {
      txtIn.text = hello_label.text;
    }
    ]]>
  </mx:Script>

  <mx:Label id="hello_label">
    <mx:text>Hello World!</mx:text>
  </mx:Label>
  <mx:Button id="btn" label="Click me!" click="copy()" />
  <mx:TextInput id="txtIn" text="not clicked" />

</mx:Application>
```

Adobe Flex



http://www-lehre.inf.uos.de/~cg/2010/Flash/flex.html

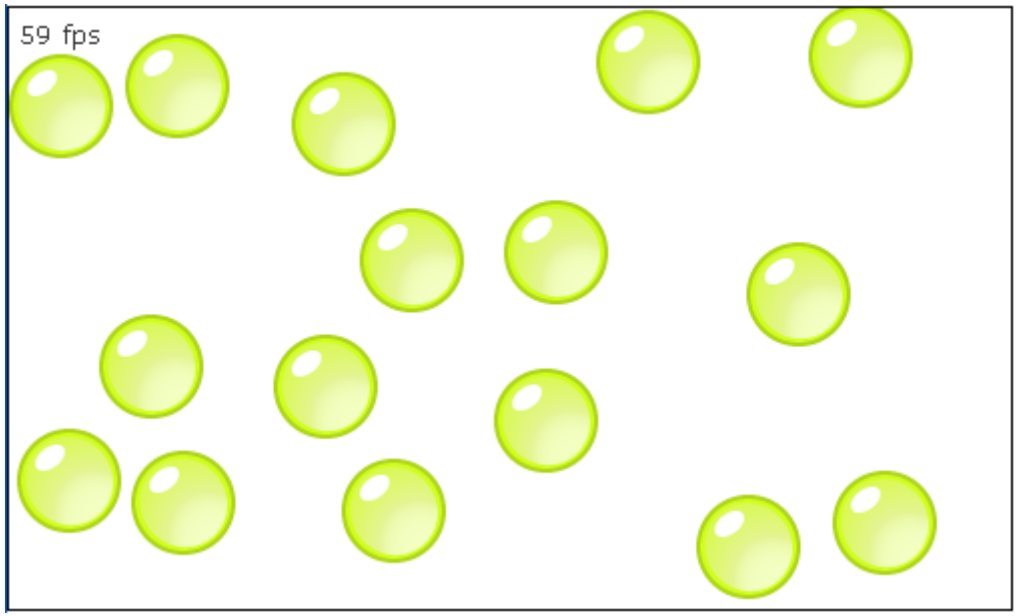
http://demo.quietlyscheming.com/ChartSampler/app.html

http://dbs.informatik.uni-osnabrueck.de:8180/cgprakt08/avid.html

Microsoft Silverlight

- Seit 2008
- Teil von WPF (Windows Presentation Foundation)
- .NET Framework
- C# + VB.NET
- XAML eXtensible Application Markup Language
- http://www.silverlight.net/showcase/

Vergleich



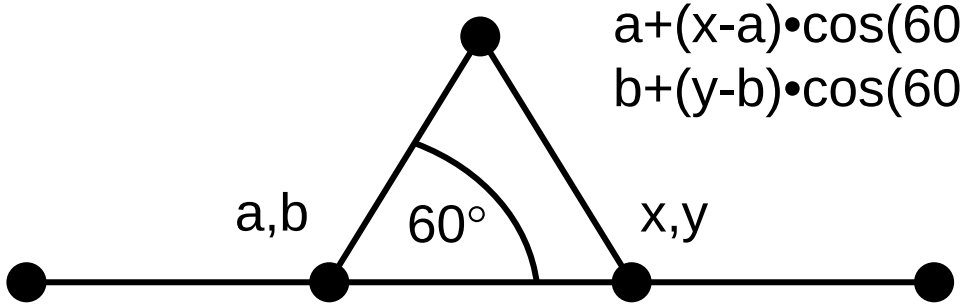
http://bubblemark.com/

Computergrafik SS 2010
Oliver Vornberger

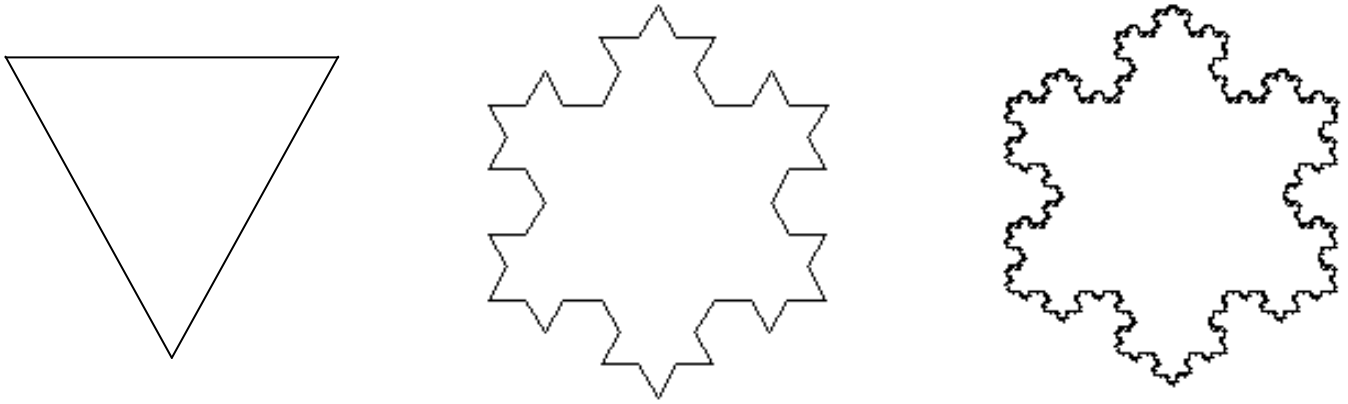
Kapitel 11:
Fraktale

Selbstähnlichkeit

Koch'sche Schneeflocke



$$a+(x-a)\cdot\cos(60^\circ) - (y-b)\cdot\sin(60^\circ)$$
$$b+(y-b)\cdot\cos(60^\circ) + (x-a)\cdot\sin(60^\circ)$$



Fraktale Dimension

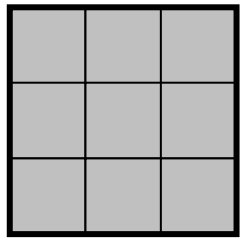
Ein selbstähnliches Objekt hat Dimension D, falls es in N identische Kopien zerfällt, skaliert mit dem Faktor $r = \frac{1}{N^{\frac{1}{D}}}$

Linie hat Dimension 1

N Teilstücke der Länge 1/N

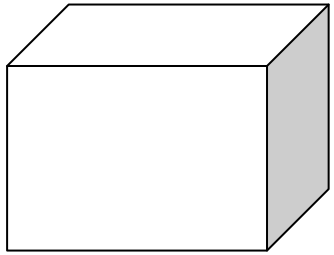
Fläche hat Dimension 2

N Teilfächen mit Kantenlänge $\frac{1}{N^{\frac{1}{2}}}$



Würfel hat Dimension 3

N Teilwürfel mit Kantenlänge $\frac{1}{N^{\frac{1}{3}}}$



Berechnung der Dimension

$$r = \frac{1}{N^{\frac{1}{D}}} \qquad \Rightarrow r \cdot N^{\frac{1}{D}} = 1$$

$$\Rightarrow N^{\frac{1}{D}} = \frac{1}{r} \qquad \Rightarrow \frac{1}{D} \cdot \log(N) = \log(\frac{1}{r})$$

$$\Rightarrow D = \frac{\log(N)}{\log(\frac{1}{r})}$$

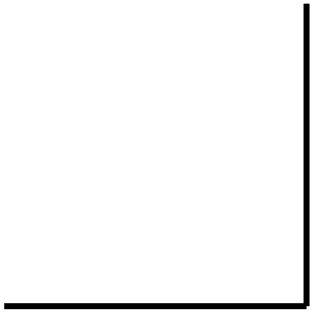
Jede Kante der Koch'schen Schneeflocke zerfällt in N=4 Kopien, skaliert mit r=1/3.

$$D = \frac{\log(4)}{\log(3)} = 1.2618$$

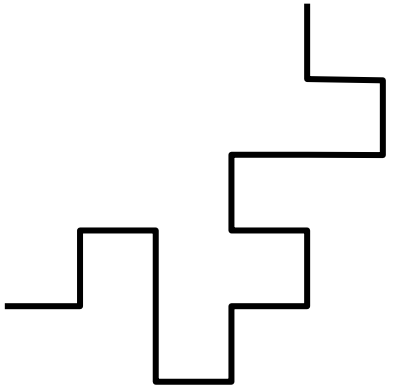
Lindenmayer-Systeme

Alphabet Σ = {**r,u,l,d**}

Regeln f = { **r** $\rightarrow$ **rurddrur**,
u $\rightarrow$ **ulurrulu**,
l $\rightarrow$ **ldluuldl**,
d $\rightarrow$ **drdlldrd**}



ru

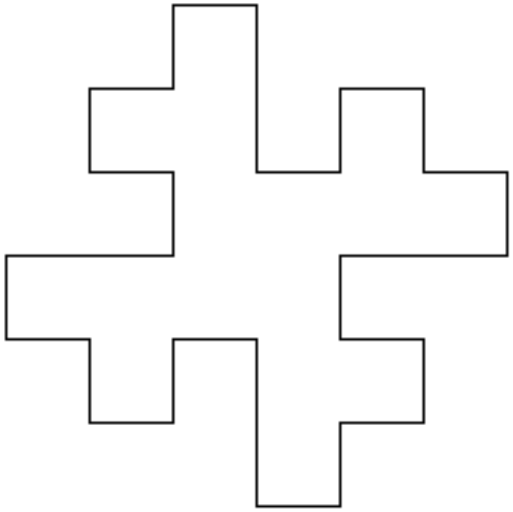


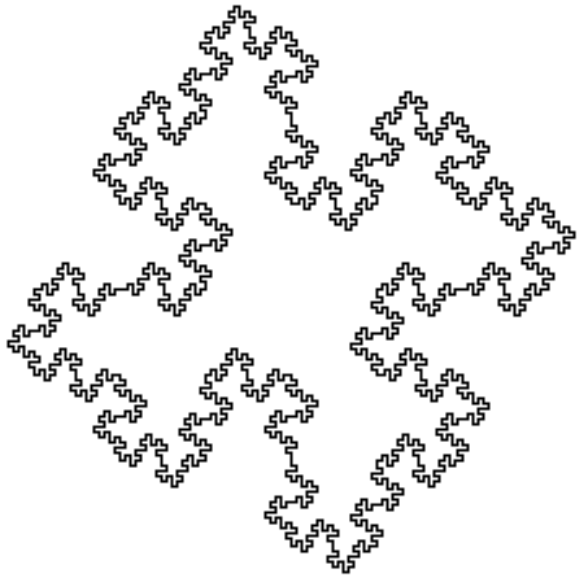
rurddrurulurrulu

Quadratische Koch-Kurve

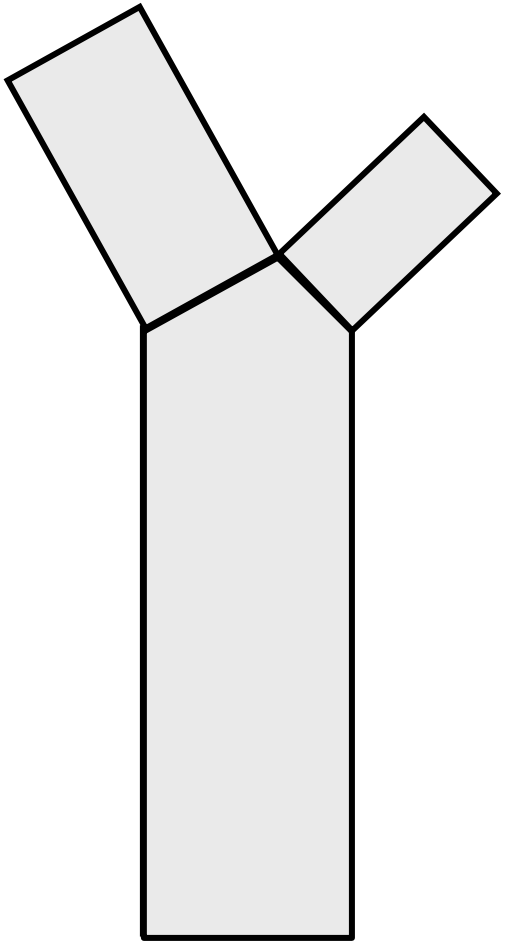
Jeder Kantenzug besteht aus 8 Kopien,
skaliert mit Faktor 1/4

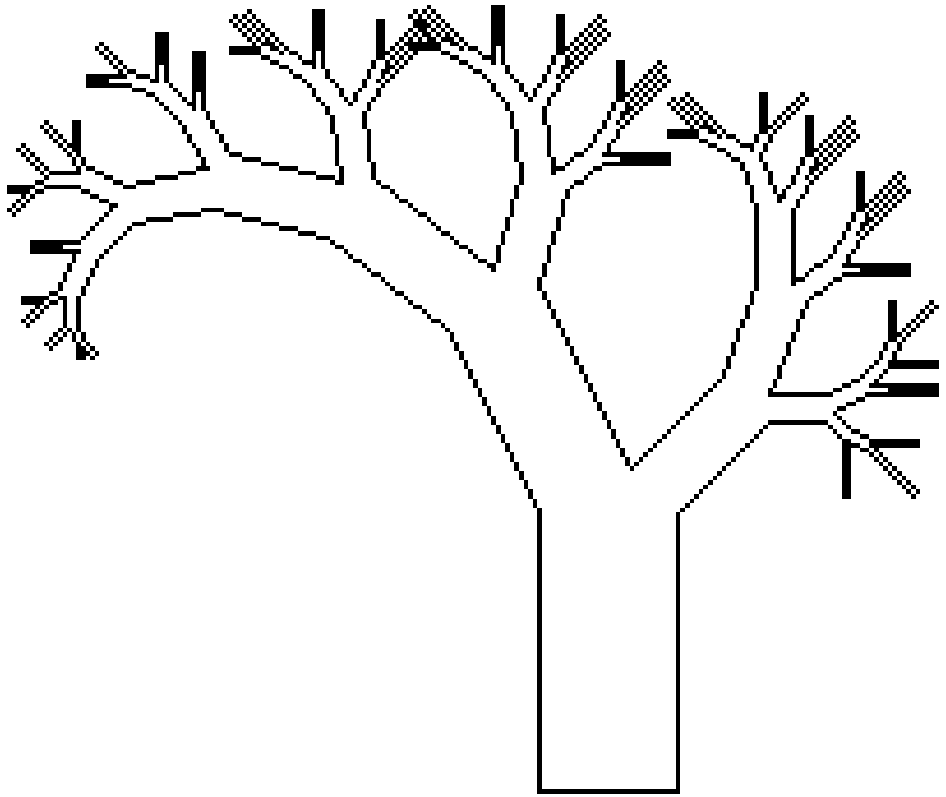
$$D = \frac{\log(8)}{\log(4)} = 1.5$$





Bäume





Implementationen zu Lindenmayer

~cg/2010/skript/Applets/Fraktale2/App.html

~cg/2010/Flash/recursiontree.html

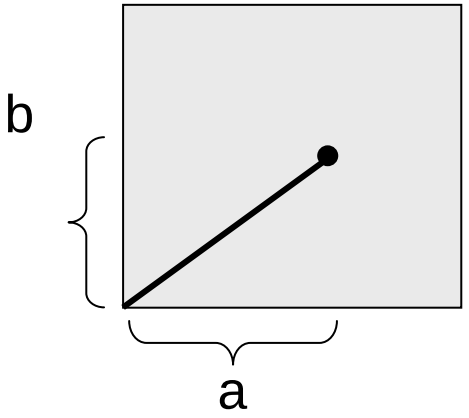
Komplexe Zahlen

Komplexe Zahl $z = a + bi$ $\qquad i^2 = -1$

Realteil a

Imaginärteil b

Betrag von $z = \sqrt{a^2 + b^2}$



Quadrieren einer komplexen Zahl:

$z^2 = (a + bi)\cdot(a + bi) = a^2 + 2abi + b^2i^2$

$\Rightarrow$ Realteil a^2-b^2

$\Rightarrow$ Imaginärteil $2ab$

Iteration

Sei c komplexe Zahl

Sei $f(z) := z^2 + c$

Betrachte 0, $f(0)$, $f^2(0)$, $f^3(0)$, ...

-1.5	0.2	-0.5	0.2
0.000000	0.000000	0.000000	0.000000
-1.500000	0.200000	-0.500000	0.200000
0.710000	-0.400000	-0.290000	0.000000
-1.155900	-0.368000	-0.415900	0.200000
-0.299319	1.050742	-0.367027	0.033640
-2.514467	-0.429014	-0.366422	0.175306
4.638493	2.357487	-0.396466	0.071527
14.457877	22.070379	-0.347930	0.143283
-279.571438	638.381719	-0.399474	0.100294

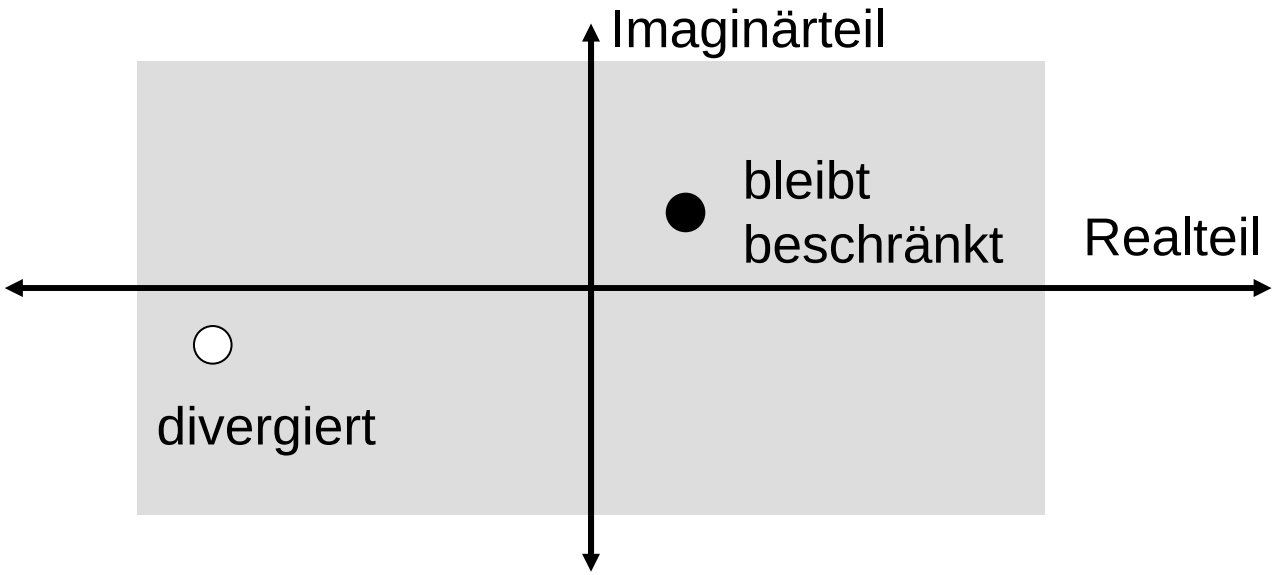
Mandelbrotmenge

Die zu c gehörende Folge kann …

- zu einem festen Wert konvergieren
- einen (beschränkten) Zyklus durchlaufen
- sich (beschränkt) chaotisch verhalten
- gegen unendlich streben

Die Menge der komplexen Zahlen c,
die bei Startwert z=0
zu einer beschränkten Folge führen,
heißt Mandelbrotmenge.

Visualisierung der Mandelbrotmenge



Implementation

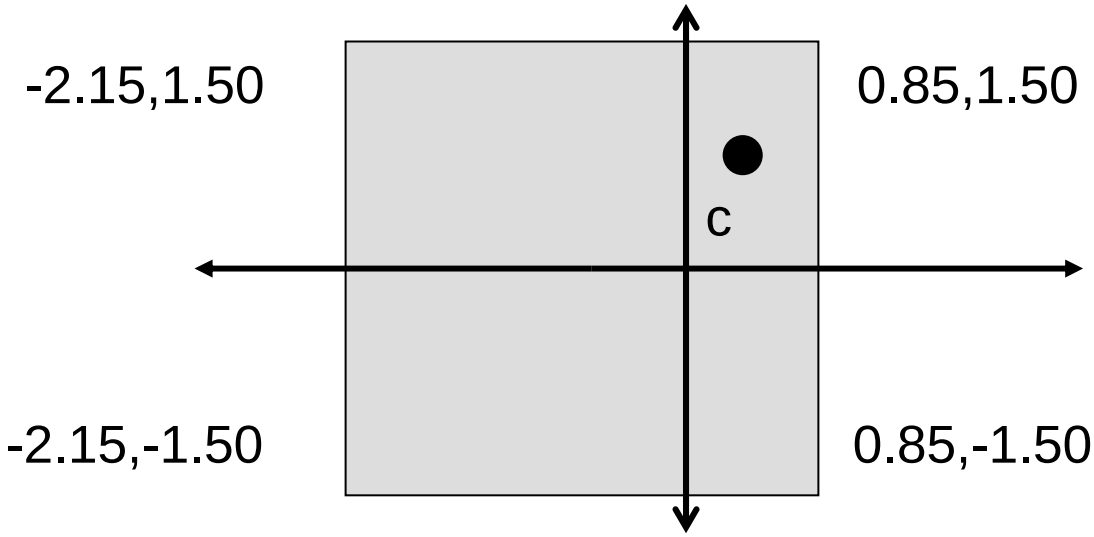
```
public Complex f(Complex z, Complex c){
  double re, im;
  re = z.re*z.re - z.im*z.im + c.re;
  im = 2*z.re*z.im+c.im;
  return new Complex(re,im);
}

int iter = 0;
Complex z = new Complex(0,0);
while (betrag(z) < 2 && iter < 100) {
  z = f(z,c);
  iter++;
}
```

iter<100: divergiert mit Sicherheit

iter>100: bleibt vermutlich beschränkt

Visualisierung



bei 300 Pixeln $\Rightarrow$ Schrittweite = (0.85+2.15)/300 = 0.01

```
public Complex (Point p, Complex start, double schritt){
      this.re = start.re + schritt*p.x;
      this.im = start.im - schritt*p.y;
}
```

Mandelbrot-Programmcode

```
Point p;
double schritt = (ende.re-start.re)/WIDTH;

for (p.x=0, c.re=start.re; p.x < WIDTH;
     p.x++, c.re+=schritt)
for (p.y=0; c.im=start.im; p.y < HEIGHT;
     p.y++, c.im+=schritt){
  Complex z = new Complex(0,0);
  int iter=0;
  while((betrag(z)<2.0) && (iter<100)){
    z = f(z,c);
    iter++;
  }
if (betrag(z) < 2.0) setPixel(p);
```

S/W-Mandelbrotmenge

$$\text{farbe}(c) := \begin{cases} \text{weiß, sobald Betrag} > 2 \\ \text{schwarz, falls Betrag} \\ \text{nach 100 Iterationen} < 2 \end{cases}$$

$$\text{farbe}(c) := \begin{cases} \text{iter\%2, sobald Betrag} > 2 \\ \text{schwarz, falls Betrag} \\ \text{nach 100 Iterationen} < 2 \end{cases}$$

bei mehr Iterationen wird schwarze Fläche weiß

Farbige Mandelbrotmenge

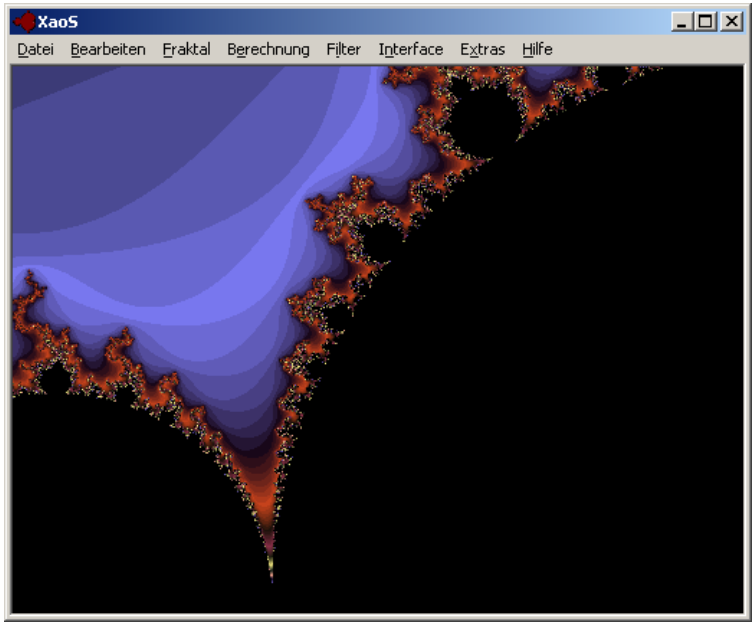
Es seien k Farben verfügbar:
col[0], col[1], …, col[k-1]

$$\text{farbe}(c) := \begin{cases} \text{col}[\text{iter}\,\%\,k], & \text{sobald Betrag} > 2 \\ \text{schwarz, falls Betrag} & \text{nach 100 Iterationen} < 2 \end{cases}$$

bei mehr Iterationen wird schwarze Fläche bunt

Java-Applet zur Mandelbrotmenge

~~cg/2010/skript/Applets/Fraktale2/App.html



XaoS von Jan Hubicka

http://wmi.math.u-szeged.hu/xaos/doku.php

→IFS

Java-Applet zur Julia-Menge

~cg/2010/skript/Applets/Fraktale2/App.html

Iterierte Funktionensysteme

Beschreibe den Bildinhalt durch System von affinen Transformationen

Pro Transformation:

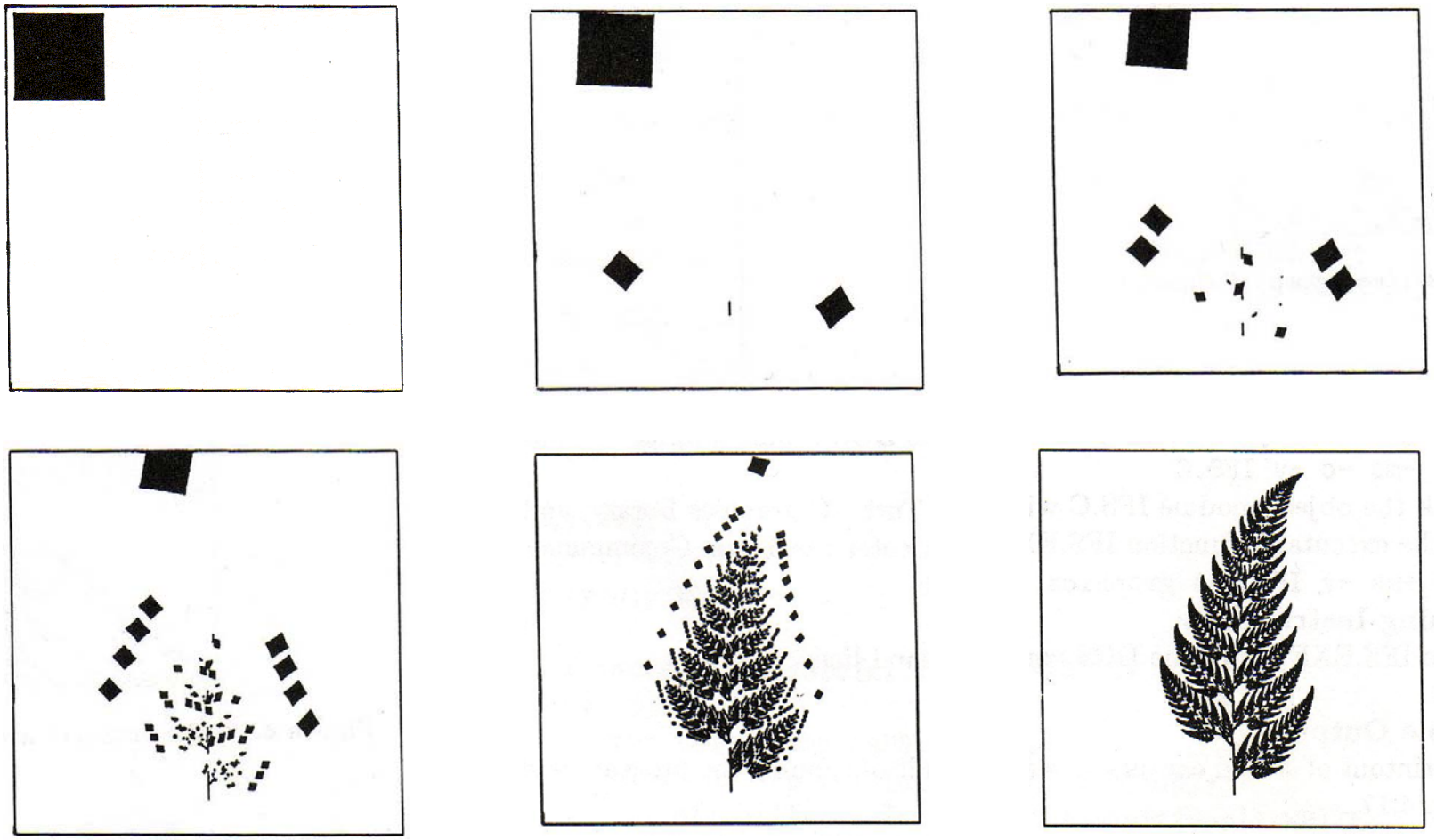
- 2×2 Matrix A
- 2×1 Vektor b
- Anwendungswahrscheinlichkeit w

Bilde $Ax+b$ mit Wahrscheinlichkeit w

Matritzen für Farn

	A	b	w
r_1	$\begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ 1.6 \end{pmatrix}$	85 %
r_2	$\begin{pmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ 1.6 \end{pmatrix}$	7 %
r_3	$\begin{pmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ 0.44 \end{pmatrix}$	7 %
r_4	$\begin{pmatrix} 0.00 & 0.00 \\ 0.00 & 0.16 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}$	1 %

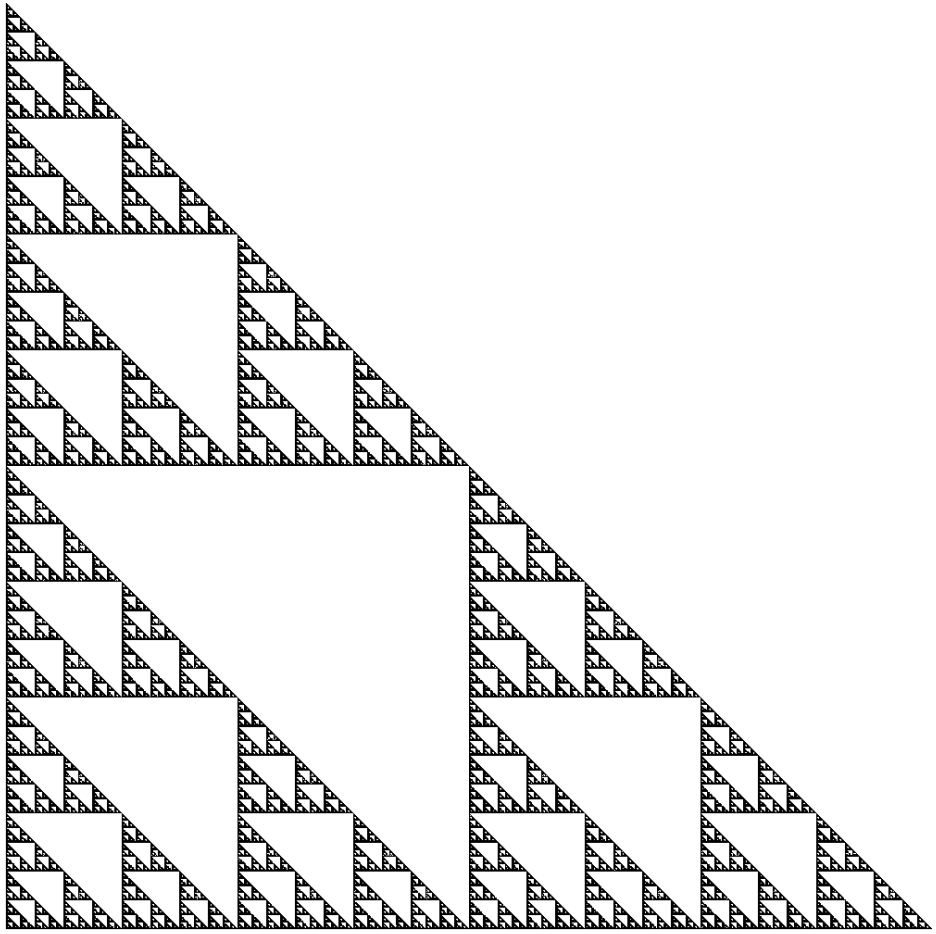
Konvergenzfolge für Farn



Farn-Algorithmus

```
Wähle Startpixel p in Zeichenfläche
while(noch_nicht_zufrieden) {
    wähle Transformation f gemäß w
    p= f(p);
    setPixel(p);
}
```

Sierpinsky-Dreieck

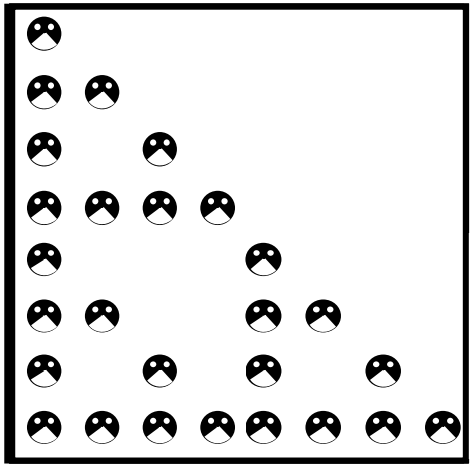


D_1	
D_2	D_3

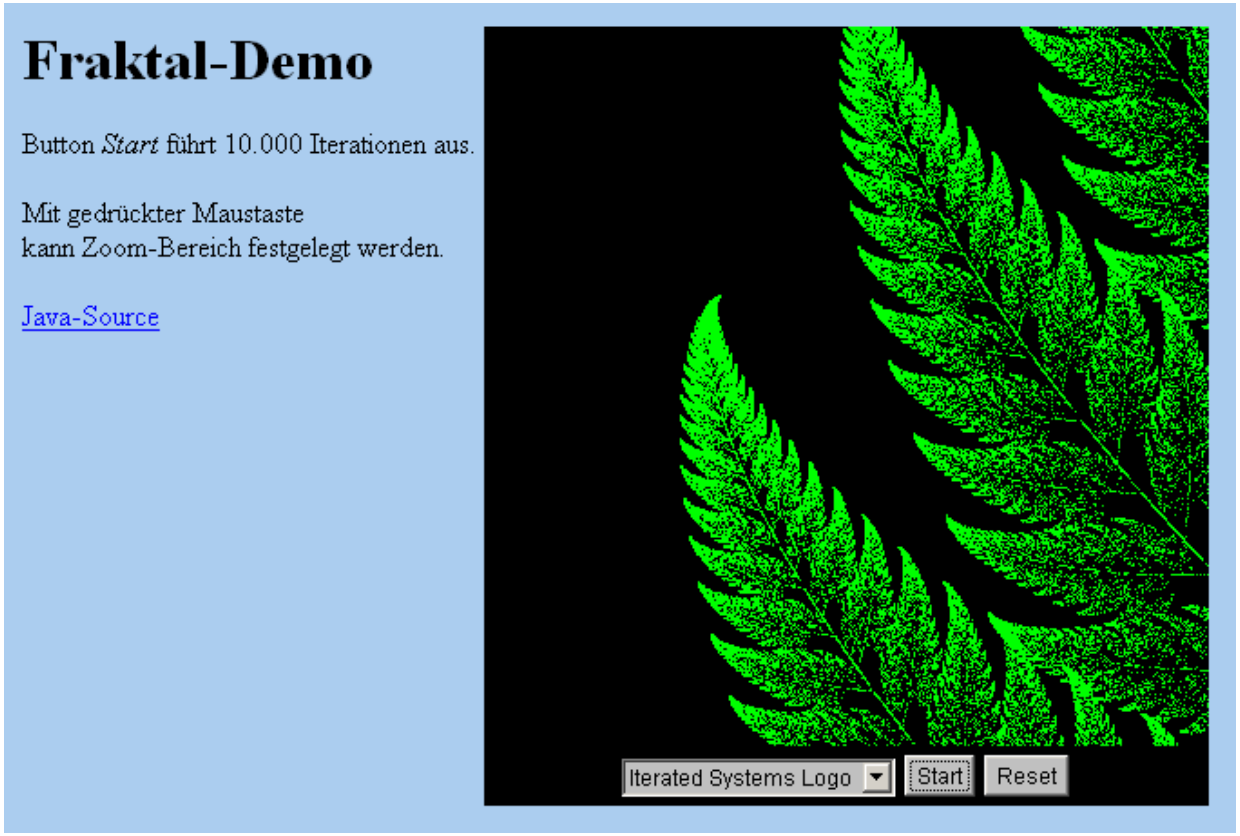
D enthält sich
3mal verkleinert.

Algorithmus für Sierpinsky

D := Rechteck mit zufälligem Inhalt
while (noch_nicht_zufrieden) do {
  D' := mit Faktor ¼ verkleinerte Version von D
  kopiere D' in den 2., 3., 4. Quadranden von D
}



Java-Applet
zu Iterierten Funktionensystemen



~cg/2010/skript/Applets/IFS/fraktal.html

Fraktale Kompression

fixiere 8 Transformationen

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
Identität

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$
90° Drehung

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$
180° Drehung

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$
270° Drehung

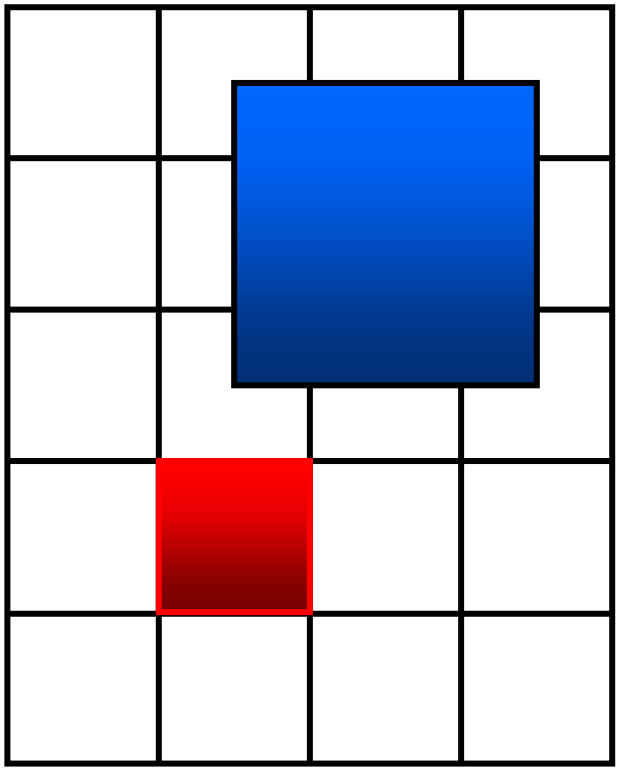
$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
Spiegelung an x

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$
Spiegelung an y

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
Spiegelung an Diagonalen

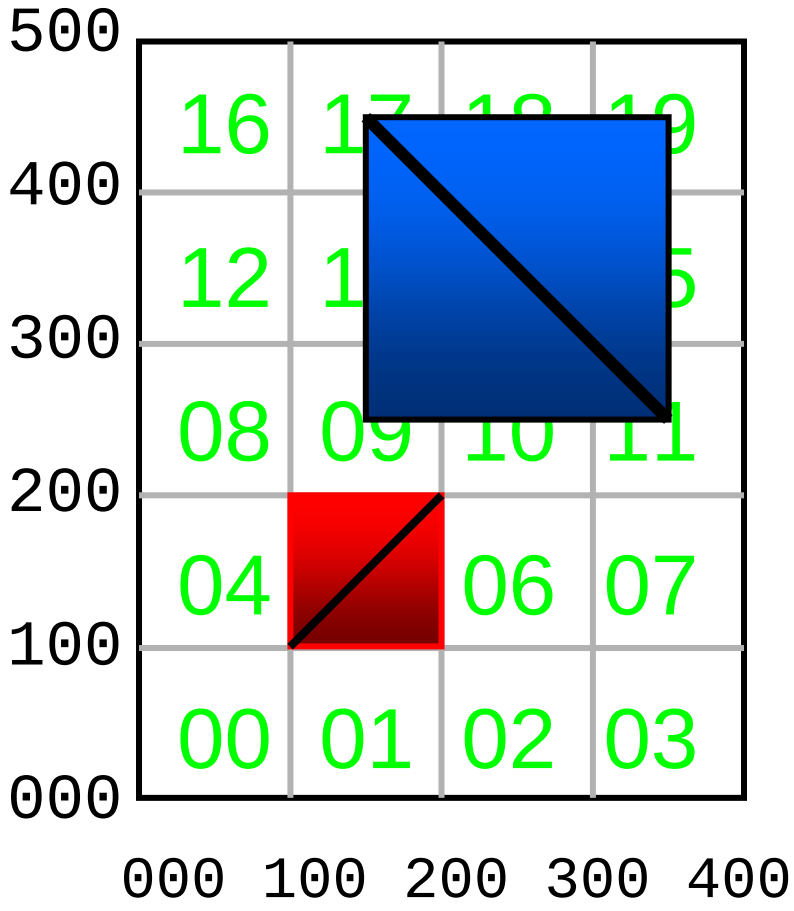
$$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$
Spiegelung an anderer Diagonalen

pro Ziel-Block suche bestes Urbild



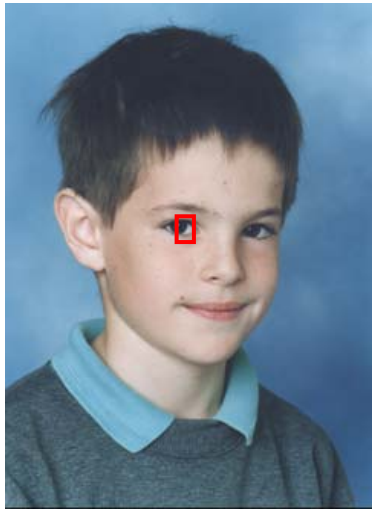
Liste der Abbildungen

Typ	Ursprung	Ziel
1	150,250	5
…		

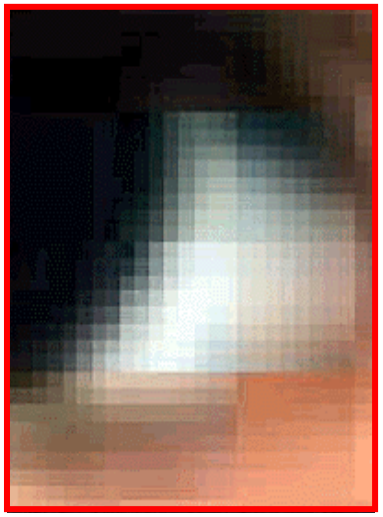


Iterated Systems

(Lizenz nun bei LizardTech)



jan.tif



Ausschnitt aus

jan.jpg

Ausschnitt aus

jan.fif