6 Fraktales Farnkraut

Ein wichtiges Thema – vor allem im Zusammenhang mit Multimedia – ist die Bildkompression. Neben JPEG bieten sich auch Fraktale zur Komprimierung von Fotos an. Da fast alle natürlichen Gegenstände Fraktale sind, lassen sie sich mit der fraktalen Geometrie genauso gut beschreiben wie künstliche mit der normalen (euklidischen) Geometrie. Michael Barnsley hat dazu die Fraktal-Kompression entwickelt. Diese beruht auf sogenannten »Iterated Function Systems« (IFS). Jedem, der sich mit IFS näher beschäftigen will, sei Barnsleys Buch »Fraktals Everywhere« empfohlen, in dem er die mathematische Theorie der IFS darlegt. In diesem Werk beweist Barnsley auch, daß man für jedes Bild ein IFS finden kann (Collage-Theorem).

Bild 5. Fraktale Gebilde mit »farn.c« muten wie Farnkraut an.

»farn.c« (Listing 8) berechnet nun ein IFS-Farnblatt aus nur 20 Zahlen (Bild 5). Es dauert relativ lange, da für ein vernünftiges Bild mindestens 20 000 Punkte berechnet werden müssen. Das Programm ist, wie man schnell sieht, nicht mehr zu optimieren, da es fast nur normale (Floatingpoint)-Berechnungen durchführt, diese aber sehr oft. Genau hier liegt das Problem. Fließkomma-Berechnungen müssen ohne Coprozessor aufwendig emuliert werden, und auch mit Coprozessor sind sie langsamer als Integer-Berechnungen. So bleiben die Werte, mit denen fast alle Berechnungen angestellt werden, in einem sehr kleinen Bereich. Dies führt dann zur Idee, statt Fließkomma-Zahlen einfach Ganzzahlen einzusetzen. Viele werden jetzt an eine Assembler-Version denken, aber es geht sehr einfach auch in C, ja sogar in Pascal. Multiplizieren Sie einfach die ursprünglichen Werte mit einer Konstante, die das Komma verschiebt (so mit 10 000), und deklarieren Sie sie als »long«.

Mit diesen neuen Werten können Sie jetzt rechnen, wobei nur einiges zu beachten bleibt:

– nach einer Multiplikation müssen Sie das Ergebnis durch die Konstante teilen;

– vor einer Division müssen Sie die Zahl mit der Konstanten multiplizieren;

– Addition und Subtraktion sind normal durchzuführen;

– bevor Sie das Ergebnis verwenden, muß es nochmal durch die Konstante geteilt werden.

Diese Regeln werden schnell klar, wenn Sie ein Beispiel mit dem Taschenrechner durchrechnen.

»farn2.c« (Listing 9) arbeitet nun mit Integer-Werten. Um es noch weiter zu beschleunigen, nutzen Sie als Konstante 2^{13}. Dies hat den Vorteil, daß Sie – anstatt durch 2^{13} zu teilen – einfach die Bits um 13 nach rechts schieben. Dies ist auf einem 286er etwa 2,7 Mal (!) schneller als eine Division. Zudem kann »farn2.c« mit einem einfachen Zoom und anderen Dingen aufwarten (Tabelle 3).

(Marcus Denker/et)

Taste	Bedeutung
+	vergrößern
–	verkleinern
Cursor	verschieben
Enter	weitere 10 000 Punkte berechnen

Tabelle 3. »farn2.c« bietet selbst mit wenigem Quelltext noch einigen Bedienungskomfort.

Listing 8: farn.c

```
/* Funktion: Fraktal wie Farn
   Sprache: Turbo C++ 1.01
   Autor: Marcus Denker
   (c)1993 DMV GmbH & Co.KG */
#include <stdlib.h>
#include <graphics.h>
float a[4] = {.85,.2,-.15},
      b[4] = {.04,-.26,.28},
      c[4] = {-.04,.23,.26},
      d[4] = {.85,.22,.24,.16},e[4],
      f[4] = {1.6,1.6,.44},
      p[4] = {85,92,99,100},x,y,X;
int i,I,l,k,g=DETECT,G;
void main(){
   initgraph(&g,&G,"");
   for (i=1;i<=20000;i++)
   {
     l=rand()%100+1;
     for (I=0;;I++) if(l<=p[I])
     {k=I;break;}
     X=a[k]*x+b[k]*y+e[k];
     y=c[k]*x+d[k]*y+f[k]; x=X;
     if (i>10) putpixel
     (x*40+300,469-y*40,2);
   }
   getch();
   closegraph();}
```

»farn.c« bildet ein fraktales Farnkraut ab.

Listing 9: farn2.c

```
/* Funktion: Fraktal wie Farn
   Sprache: Turbo C++
   Autor: Marcus Denker
   (c)1993 DMV GmbH & Co.KG */
#include <stdlib.h>
#include <graphics.h>
#define C cleardevice();iterat();break;
long  a[4]={6963,1638,-1228},
      b[4]={327,-2129,2293},
      c[4]={-327,1884,2129},
     d[4]={6963,1802,1966,1310},e[4],
      f[4]={13107,13107,3604},
      x,y,X;
int  p[4]={85,92,99,100},
     m=45,v[2]={300,479},
     i,I,l,k,g=DETECT,G;
char ch=13;

void iterat()
  {  for (i=0;i<10000;i++)
     {
      l=rand()%100+1;
      for (I=0;;I++) if(l<=p[I])
      {k=I;break;}
      X = (a[k]*x>>13) +
      (b[k]*y>>13) +e[k];
      y = (c[k]*x>>13) +
      (d[k]*y>>13) +f[k];
      x=X;
      if (i>10) putpixel
      ((x*m>>13)+v[0],v[1]-
      (y*m>>13),2);
     }
  }
void main(){
   initgraph(&g,&G,"");
   do{
    switch(ch){
       case 13: iterat();break;
       case '+': m+=10;C
       case '-': m-=10;C
       case 75 : v[0]-=20;C
       case 77: v[0]+=20;C
       case 72: v[1]-=20;C
       case 80: v[1]+=20;C
    }
    ch=getch();
   } while (ch!=0x1B);
   closegraph();}
```

»farn2.c« steigert Geschwindigkeit und Bedienungskomfort bei der Abbildung des Fraktals.