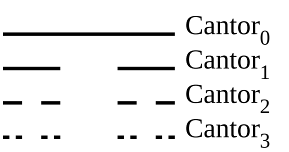
Classical Math Fractals in PostScript

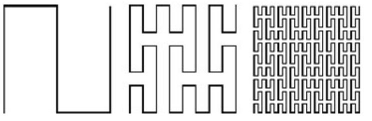
Fractal Geometry I

Abstract
Classical mathematical fractals in BASIC are explained and converted into mean-and-lean EPSF defs, of which the `.eps` pictures are delivered in `.pdf` format and cropped to the prescribed `BoundingBox` when processed by `Acrobat Pro`, to be included easily in pdf(La)TEX, Word, ... documents. The EPSF fractals are transcriptions of the Turtle Graphics BASIC codes or programmed anew, recursively, based on the production rules of oriented objects. The Lindenmayer production rules are enriched by PostScript concepts. Experience gained in converting a TEX script into WYSIWYG Word is communicated.

Keywords
Acrobat Pro, Adobe, art, attractor, backtracking, BASIC, Cantor Dust, C curve, dragon curve, EPSF, FIFO, fractal, fractal dimension, fractal geometry, Game of Life, Hilbert curve, IDE (Integrated development Environment), IFS (Iterated Function System), infinity, kronkel (twist), Lauwerier, Lévy, LIFO, Lindenmayer, minimal encapsulated PostScript, minimal plain TeX, Minkowski, Monte Carlo, Photoshop, production rule, PSlib, self-similarity, Sierpiński (island, carpet), Star fractals, TACP, TEXworks, Turtle Graphics, (adaptable) user space, von Koch (island), Word

Contents

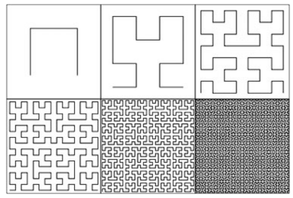



Peano curves: order 1, 2, 3

Introduction

My late professor Hans Lauwerier published nice, inspiring booklets about fractals with programs in BASIC. However, I don't know how to include elegantly the pictures, obtained by running the BASIC codes, in my documents. Moreover, I consider PostScript (PS, for short) more portable in place and time, can include EPSF results in my TEX documents[1] easily, and ... do realize that PS is the de-facto standard industrial printer language.

This note is about conversion of some of Lauwerier's BASIC Turtle Graphics codes for the simplest fractals into EPSF, ànd about the programming of new recursive EPSF `defs` biased by Lindenmayer production rules for oriented objects, enriched with PS concepts.



← Hilbert curves

Sierpiński →
islands 1, 2, 3

Now and then I have explained Lauwerier's algorithms, especially when he associates binary and quaternary number representations with fractals.

Fractals have widened the dimension concept into fractal-valued dimensions. Although the fractal dimension concept is not necessary in order to understand the codes, I have added the appendix Fractal Dimension, because fractal dimensions contribute to characterizing fractals. Moreover, fractal dimension gives meaning to the 19^{th} century 'monstruous' plane-filling curves.

Fractals were invented in the 20^{th} century, and became the geometry of this century due to the development of computers, because computers are the tools for viewing and researching fractals.

The ancestor of fractals is the 1D Cantor Dust. 2D predecessors of fractals are the plane-filling curves named after Peano, Hilbert, Sierpiński, ... , which captivated mathematicians in the late 19^{th} and the early 20^{th} century.

Sierpiński curves have found their niche in the solution of the travelling salesman problem.

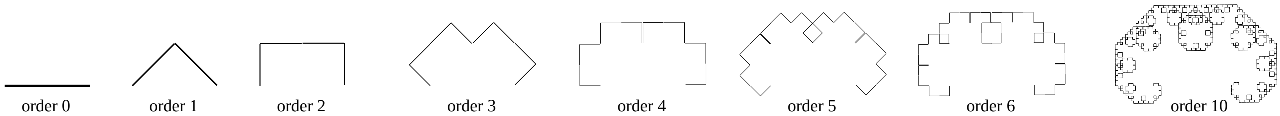
In the sequel Lévy, von Koch, Kronkel (Dutch, means twist), Minkowski, Dragon curve, star fractals, and a variant of the Game of Life are discussed. There are 4 appendices: the first about Fractal Dimension, the second about the historical Cantor Dust, the third about the classical Hilbert curve, and the last about Sierpiński islands.

In the footsteps of Lauwerier, the reader is invited to experiment with the PS programs, of which `defs` are supplied in my `PSlib.eps` library, which I'll send on request. MetaPost aficionados may translate the included Metafont codes into MetaPost, I presume.

Lévy fractal

An approximation of the Lévy fractal is also called a C (broken) line of a certain order. The constructive definition of various orders of C lines starts with a straight line, let us call this line C_0. An isosceles triangle with angles 45°, 90° and 45° is built on this line as hypotenuse. The original line is then replaced by the other two sides of this triangle to obtain C_1. Next, the two new lines each form the base for another right-angled isosceles triangle, and are replaced by the other two sides of their respective triangle, to obtain C_2. After two steps, the broken line has taken the appearance of three sides of a rectangle of twice the length of the original line. At each subsequent stage, each segment in the C figure is replaced by the other two sides of a right-angled isosceles triangle built on it. Such a rewriting relates to a Lindenmayer system. After n stages the C line has length $2^{n/2}\times C_0$: 2^n segments each of size $2^{-n/2}\times C_0$.

Fractals have various infinite lengths. The question arose: Can these blends of ∞ be used to characterize fractals?[2] Below C_0 ... C_6 and C_{10} have been constructed from the definition.

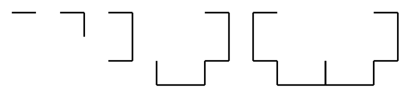


Properties

1a. The above sequence of curves loosely obey

$$C_i = C_{i-1}^{45} \oplus C_{i-1}^{-45}, \quad i = 1, 2, \ldots \quad C_0 = \text{segment}$$
$$\oplus \text{ means spliced} \quad C_{i-1}^{45} \text{ means rotated over } 45^\circ.$$

In fractal terminology such a recursion, or production rule, characterizes what is called the self-similarity of fractals, because C_i consists of 2 spliced copies of C_{i-1}, which are not scalars but 2D, oriented objects. (Positive rotation is counter-clockwise à la PS). Lindenmayer (Dutch theoretical biologist) invented production rules in order to describe plants; production rules are also used in program development.



1b. The C curves at right have a different orientation.
The formula, which reflects the self-similarity in this orientation, reads

$$C_i = C_{i-1} \oplus C_{i-1}^{-90}, \quad i = 1, 2, \ldots$$
$$\oplus \text{ means spliced} \quad C_{i-1}^{-90} \text{ means rotated over } -90^\circ.$$

Self-similarity as construction method can be suitably programmed in MetaPost/-font, with their path data structure, as follows. Create the row of paths $p_0, p_1, p_2, \ldots$

$$p_0 = C_0, \qquad p_i^{45} \oplus p_i^{-45} \to p_{i+1}, \quad \text{for } i = 0, 1, 2, \ldots \quad \text{with } \oplus \text{ the splice operator.}$$

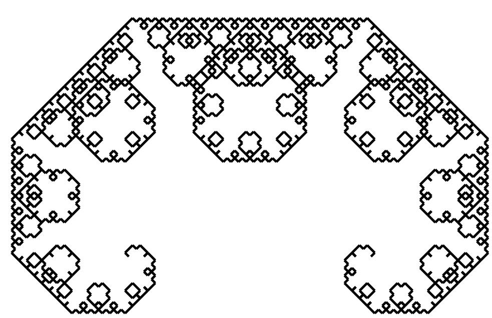
2. In the pen-plotter days the natural question arose: What is the direction of a segment? Lauwerier(1987) gives the intriguing relationship between the angle ϕ_k of a segment and its index k (according to the orientation as given under 1b).

$$\phi_k = (s_k \,\mathbf{mod}\, 4)\frac{\pi}{2} \quad \text{with } s_k = \textstyle\sum_{j=0}^{p-1} b_j \quad \text{sum of bimals of k}$$
$$\text{and } k = \textstyle\sum_{j=0}^{p-1} b_j 2^j \text{ binary representation of k.}$$

3. The Lévy fractal has fractal dimension 2, a local plane-filling curve, Lauwerier(1990). The C curves intersects themselves from order 4 onward.

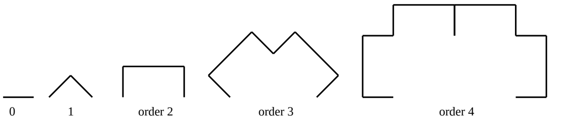
The PostScript program

One might create an efficient recursive backtracking program based on property 1a, as a `levyC def` with the `def` given below. Scaling is commented out; just remove the two % signs if scaling is wanted.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Author: Kees van der Laan
%%Date: april 2011
%%Affiliation: kisa1@xs4all.nl
%%BoundingBox: -1 -1 346 61
%%BeginSetup %crops to BoundingBox
%%EndSetup   %by Acrobat Pro
%%BeginProlog%collection of defs
/levyC{%on stack: the order ==> C line
```

```
      %s = size of line segment (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}%draw line
{1 sub %/s s 1.4142 div def %lower order on stack; s scaled variant
  45 rotate levyC%-45 rotate%combine -45 twice into -90
 -90 rotate levyC  45 rotate
 1 add %/s s 1.4142 div def %adjust order on stack, and s(cale)
}ifelse
}def
%%EndProlog
%
% Program --- the script ---
%
/s 20 def            0 levyC pop
s 2 div   0 translate 1 levyC pop
s         0 translate 2 levyC pop
1.5 s mul 0 translate 3 levyC pop
2.5 s mul 0 translate 4 levyC pop
showpage
%%EOF
```



The above mean-and-lean PS `def` is the result of programming in the spirit of The Art of Computer Programming, TACP for short. I'll come back on a more systematic approach of programming based on production rules, a little further on.

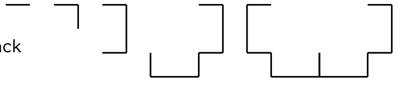
If the `levyC def` is included in the `PSlib.eps` library, then the above `def` can be replaced by

```
(C:\\PSlib\\PSlib.eps) run
```

This feature of the run command is not generally known, so it seems.

Because programming in PostScript is subtle, I have included below the PS `def` based on the production rule as stated under property 1b, which is highly similar to the above backtracking process, but the result differs in orientation. This `levyCvar def` is also included in `PSlib.eps`.

```
/levyCvar{%order on stack ==> C line
          %s = size of segment (global))
dup 0 eq
{0 0 moveto s 0 lineto
   currentpoint stroke translate}%draw line
{1 sub                         %lower order on stack
          levyCvar             %C line
 -90 rotate levyCvar 90 rotate   %rotated C line
  1 add                        %adjust order on stack
}ifelse }def
```



To run the program store the file with extension `.eps` (or `.ps`), right-mouse click the thumbnail of the file and choose the option convert to Adobe PDF in the pop-up menu. That is all when you have installed Acrobat Pro 7. (Other versions of Creative Suite ask for `open in Acrobat`.) I also used Adobe Illustrator and PSView. The latter just by double-clicking the filename upon which the command window opened and a little later PSview.[3]

The Turtle Graphics algorithm is based on property 2. In order to understand the formula mentioned, a table for the direction (with orientation as mentioned under property 1b) of each segment is included. Such a table forms the basis for discovering the regularity.

order \ k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	→																
1	→	↓															
2	→	↓	↓	←													
3	→	↓	↓	←	↓	←	←	↑									
4	→	↓	↓	←	↓	←	←	↑	↓	←	←	↑	←	↑	↑	→	
s_k **mod** 4	0	1	1	2	1	2	2	3	1	2	2	3	2	3	3	0	...

Below I have included Lauwerier's program and my conversion in PS, which is interesting because of the transformation of the user space by ϕ_k, $k = 0, 1, 2, \dots$.

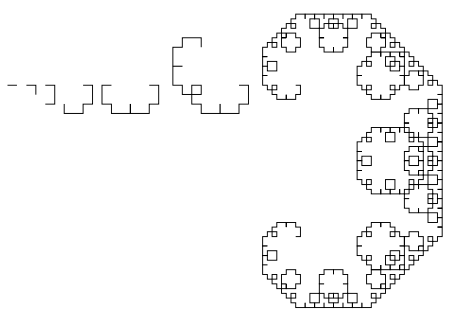
```
10 REM ***Levy fractal***
10 P=12 : REM***order***
20 H=2^(-(P/2)) : A=H*COS(P*PI/4) : B=H*SIN(P*PI/4)
30 LINE (0,0)-(A,-B) : LINE -(A+B,A-B)
40 X=1 : Y=1
50 FOR N=2 TO 2^P-1
60 M=N : S=1
70 IF M MOD 2 = 1 THEN S=S=1
80    M\2
90 IF M>1 THEN GOTO 70
100 IF S MOD 4 = 0 THEN X=X+1
110 IF S MOD 4 = 1 THEN Y=Y+1
120 IF S MOD 4 = 2 THEN X=X-1
130 IF S MOD 4 = 3 THEN Y=Y-1
140 LINE -(A*X+B*Y, A*Y-B*X)
150 NEXT N
160 END
```

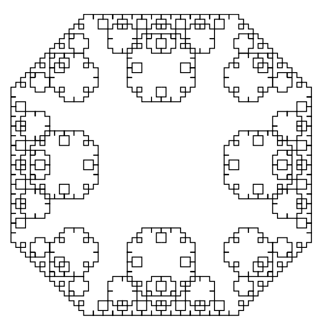
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Levy fractal a la Lauwerier
%%Transcriptor: Kees van der Laan, April 2011, kisa1@xs4all.nl
/LevyLauwerier{/p exch def%order =>C curve (l is global)
0 0 moveto l 0 lineto    %order 0
p 1 eq {0 l neg rlineto} %order 1, segments 0, 1
{0 %previous phi on stack
1 1 2 p exp 1 sub {%for%n, the number of the segments 1,2,...
0 exch %s n :s sum of bimals and n the segment number on stack
   p{dup 2 mod 3 -1 roll add exch 2 idiv}repeat pop%discard n
   4 mod 90 mul dup 3 1 roll sub rotate l 0 rlineto
}for%n
pop%discard phi
}ifelse stroke}def
/l 5 def 10 LevyLauwerier stroke showpage
%%EOF
```

Size of line-piece is wired-in (In PS parametrized)

Subtle stack programming with only 1 rotate for each segment instead of a rotate and a back rotate. The length l of each segment is a global parameter.

Below at left C_0 ...C_5, and C_{10}; at right C_{10} spliced with its `-1 1 scaled` copy, a Lévy carpet.





In my PWT guide of 1995, I did program the above Lévy fractal in TEX (orientation 1b) by the Turtle graphics method, in the footsteps of Knuth. Nowadays, I much prefer the much more powerful and useful PostScript for programming my graphics. Sorry to say so, but Knuth put me on the wrong track by his graphics in the TEXbook.

Lindenmayer enriched by PostScript concepts for the Lévy fractal

What we miss in the 1a property specification is the scaling to smaller size of the segments when the order increases, as well as a more precise meaning of what spliced entails. A more accurate and improved production rule à la 1a, can be obtained when we use PS concepts in the production rule at the expense of simplicity.

$$C_n = [R_{45} S_{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})} C_{n-1}] \oplus T_{\frac{s}{2}\frac{s}{2}} [R_{-45} S_{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})} C_{n-1}]$$

with $C_0 =$ initial line, and
C_n the Lévy C curve of order n,
$\oplus$ splice operator, meaning add properly, i.e. $T_{(\frac{s}{2}, \frac{s}{2})}$,
[means store graphics state on the GS stack and open a new one,
] means remove current graphics state off the GS stack and recall previous,
R_{45} means rotate US 45° in the PS sense,
$S_{a,b}$ means scale US by a and b, in x- and y-direction
$T_{a,b}$ means translate US by a and b, in x- and y-direction.

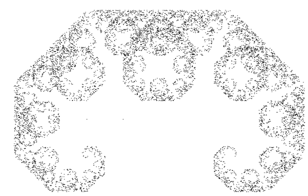
The above PS production rule transcribes systematically into the following PS `def`, which has become more verbose.

```
!PS-Adobe-3.0 EPSF-3.0
%%Author: Kees van der Laan
%%Date: feb 2012
...
/levyC{%on stack: the order => C line
       %s = size of initial segment C_0 (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub
  gsave  45 rotate .7071 dup scale levyC grestore
 .5 s mul dup translate
  gsave -45 rotate .7071 dup scale levyC grestore
 1 add%reset order
}ifelse
}def
```

Systematic programming versus TACP at the expense of verbosity.

Lévy fractal as Iterated Function System

Lauwerier(1994) in one of his exercises created a Lévy fractal by the IFS (Iterated Function Systems) method, which consists of 2 contracted, affine transformations, L and R, (both rotations characterize Lévy) applied with equal chance.



$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{L}{=} \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a-1 \\ b \end{pmatrix} \text{ and}$$
$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{R}{=} \begin{pmatrix} c & -d \\ d & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1-c \\ -d \end{pmatrix}, \ a = .5,\ b = a = c = -d.$$

or after substituting the parameters

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{L}{=} .5 \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + .5 \begin{pmatrix} -1 \\ 1 \end{pmatrix} \text{ and}$$
$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{R}{=} .5 \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + .5 \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

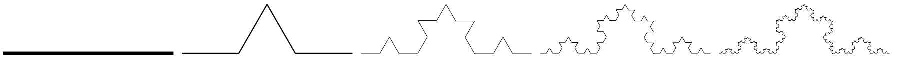
Associated with the Lévy fractal are 2 rotations with rotation centres for L: (-1,0) and for R: (1,0) and contraction $\sqrt{.5} \approx .7$. Amazing, isn't it! Laurier's BASIC program `FRACMC2` and my conversion are given below.

```
REM ***naam: FRACMC2***
KMAX=60000
REM ***Coefficienten***
   A=.5: B=A: C=A : D=-A
   DET1=A*A+B*B : DET2=C*C+D*D
   Q=DET1/(DET1+DET2)
   X=1 : Y=0 : K=0 : KMAX=10000
   DO WHILE K<KMAX AND INKEYS$=""
      R=RND
      IF R<Q THEN
         U=A*X-B*Y-1+A : V=B*X+A*Y+B 'rotatie L
      ELSE
         U=C*X-D*Y-1-C : V=D*X+C*Y-D 'rotatie R
      ENDIF
      X=U : Y=V
      PSET (X,Y)
   LOOP : BEEP
END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: fracmc2
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%Transcriptor: Kees van der Laan, febr 2012
%%BoundingBox: -203 -54 205 206
%%BeginSetup
%%EndSetup
%%BeginProlog
/Courier 7 selectfont
/x 0 def  /y 0 def /halfmaxint 2 30 exp  def
/a .5 def  /b a def /c a def /d a neg def
/det1 a dup mul b dup mul add def /det2 c dup mul d dup mul add def
/q det1 det1 det2 add div def
/printxy{x s  y s moveto (.) show}def
%%EndProlog
/s {100 mul}def 10 srand%10 is seed
10000{rand halfmaxint lt
      {/xnew a x mul b y mul sub 1 sub a add def
       /y b x mul a y mul add b add def           /x xnew def%rot L
      }{/xnew c x mul d y mul sub 1 add c sub def
        /y    d x mul c y mul add       d sub def /x xnew def%rot R
      }ifelse
      printxy}repeat
showpage
%%EOF
```

Helge von Koch

A von Koch broken line and a Lévy C line are related to a Lindenmayer system, also called a rewrite system. For the von Koch broken line the rewrite is: divide a line in 3 pieces and replace the middle piece by an equilateral triangle, with the base omitted. Repeat the process on the 4 line pieces to the required order. It is similar to the defining construction process of the Lévy fractal; the result conveys a different impression, however. Below $K_0 \ldots K_4$, scaled with increasing order (line thickness is scaled as well).



Properties

1. Each von Koch curve contains 4 copies of the von Koch curve of an order lower, meaning self-similarity, which entails the production rule

$$K_i = K_{i-1} \oplus K_{i-1}^{60} \oplus K_{i-1}^{-60} \oplus K_{i-1},$$

$$\text{with } K_0 = \text{initial segment}, \ \oplus \text{ means spliced, } K^{60} \text{ rotated over } 60^\circ.$$

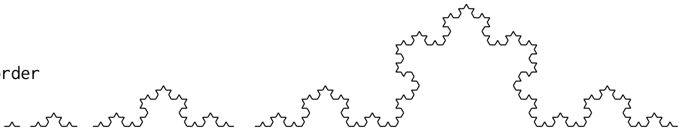
2. The von Koch fractal is a historical example of a curve without a tangent. The curve never intersects itself.
3. The length of the broken line for order n is $(4/3)^n \times K_0$, which with increasing order n goes to ∞. The von Koch curves gave rise to the awareness that the length of the coast of England is infinite. Imagine that the yardstick has length K_0, then all the lines above of the scaled von Koch curves have length 1! So, the length of a fractal depends on the size of your yardstick! Awareness of grades of infinity stirred up the concept of the fractal dimension D, a jolt to the minds of those with an iron cast idea about the 1-2-3-dimensional geometrical world. The fractal dimension is: D = log4 / log3 ≈ 1.26.
4. Lauwerier(1987) mentions the intriguing relationship between the angle ϕ_k of a segment and its index k

$$\phi_k = ((s_k + 1)\,\mathbf{mod}\,3 - p)\frac{\pi}{3} \quad \text{with } s_k = \sum_{j=0}^{p-1} q_j \quad \text{sum of quatermals of k}$$
$$\text{and } k = \sum_{j=0}^{p-1} q_j 4^j \text{ quaternary representation of k.}$$

5. The von Koch island remains within the circumscribed circle of the initial triangle (see later).

The PostScript def is an efficient and concise implementation of the above specified rewrite under property 1, neglecting scaling.

```
/vonKoch{%on stack order >=0; ==> von Koch
        %s = size of the line segment (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub vonKoch %lower the order on the stack and do von Koch
   60 rotate vonKoch
 -120 rotate vonKoch
   60 rotate vonKoch
 1 add        %reset order
}ifelse}def
```



Turtle Graphics algorithm is based on the knowledge of each angle ϕ_k. In order to understand, or get a feeling for, the formula mentioned, I have included a small table for the orders 0 and 1. Order 2 yields a too long table, and has been suppressed.

order	p	k	s_k	$((s_k + 1)\,\mathbf{mod}\,3) - p$	ϕ_k
0	1	0	0	0	0
1	1	0	0	0	0
		1	1	1	$\pi/3$
		2	2	-1	$-\pi/3$
		3	3	0	0

Lauwerier coded a BASIC program based on the knowledge of the direction of each segment via the Turtle Graphics method. Below I have included Lauwerier's tiny program next to my conversion in PS.

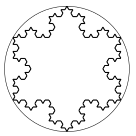
```
10 REM ***Fractal of Helge von Koch***
10 P=4 : DIM T(P) : PI= 3.141593 : REM***order***
20 H=3^(-P) : PSET (0,0)
30 FOR N=0 TO 4^P-1
40 M=N : FOR L=0 TO P-1
50      T(L)=M MOD 4 : M=M\4 : NEXT L
60 S=0 : FOR K=0 TO P-1
70      S=S+(T(K)+1) MOD 3 - 1: NEXT K
80 X=X+H*COS(S*PI/3)
90 Y=Y+H*SIN(S*PI/3)
100 LINE -(X, Y)
110 NEXT N
120 END
```

Length of line-piece wired-in.

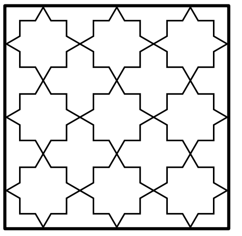
```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: von Koch fractal a la Lauwerier
%%Transcriptor: Kees van der Laan, April 2011, kisa1@xs4all.nl
/vonKochLauwerier{/p exch def %order => von Koch fractal (l is global)
/t p array def /h l 3 p exp div def 0 0 moveto
0 1 4 p exp 1 sub{%for%n
/m exch def
0 1 p 1 sub{t exch m cvi 4 mod put
            /m m cvi 4 idiv def
           }for
/s 0 def
0 1 p 1 sub{t exch get 1 add cvi 3 mod s add /s exch def}for
 /s s p sub def
 60 s mul cos h mul 60 s mul sin h mul  rlineto
}for }def
%
/l 100 def 4 vonKochLauwerier stroke
showpage
%%EOF
```

Note that in PS we have to convert the subscript expression for the index of an array explicitly into integer. Another difference is that the arguments of the trigonometric functions are in degrees in PS and in radians in BASIC.

A von Koch island is a closed splicing of von Koch fractals; at right a von Koch tile (van der Laan(1997)).



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: von Koch triangular island
%%...
/s 100 def
gsave .5 s mul dup neg exch translate 3 vonKochfractal pop
grestore
gsave .5 s mul dup translate
      -120 rotate 3 vonKochfractal pop grestore
gsave 0  -.366 s mul translate
      -240 rotate 3 vonKochfractal pop grestore
.001 setlinewidth 0 21 57.8 0 360 arc stroke
showpage
%%EOF
```



Lindemayer system enriched with PostScript concepts for the von Koch fractal

What we miss in the program is the scaling to smaller size of the segments when the order increases, as well as a more precise meaning of what spliced entails. A more precise production rule enriched with PS concepts reads

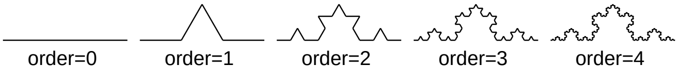
$$K_n = [S_{\frac{1}{3} \frac{1}{3}} K_{n-1}] \oplus T_{\frac{s}{3} 0} [S_{\frac{1}{3} \frac{1}{3}} R_{60} K_{n-1}] \oplus T_{\frac{s}{6} \frac{s\sqrt{3}}{6}}$$

$$[S_{\frac{1}{3} \frac{1}{3}} R_{-60} K_{n-1}] \oplus T_{\frac{s}{6} \frac{-s\sqrt{3}}{6}} [S_{\frac{1}{3} \frac{1}{3}} K_{n-1}]$$

with
K_0 the initial line,
K_n the von Koch curve of order n,
$\oplus$ splice operator, meaning add properly, i.e. translate,
[open a new GS on the GS stack,
] remove current graphics state from the GS stack and recall previous,
R_{60} means rotate US 60° in the PS sense,
$S_{a,b}$ means scale US by a and b, in x- and y-direction
$T_{a,b}$ means translate US by a and b, in x- and y-direction.

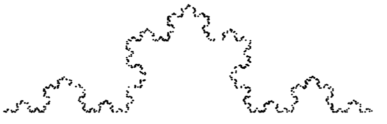
The above PS production rule transcribes systematically into the following PS `def`.

```
!PS-Adobe-3.0 EPSF-3.0
%%Author: Kees van der Laan
%%Date: feb 2012
...
/vonKoch{%on stack: the order => von Koch curve
        %s = size of initial line segment C_0 (global)
dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub %adjust order on the stack
  gsave            .3333 dup scale vonKoch grestore
  .3333 s mul          0 translate
  gsave  60 rotate .3333 dup scale vonKoch grestore
  .1666 s mul .285 s mul translate
  gsave -60 rotate .3333 dup scale vonKoch grestore
  .1666 s mul -.285 s mul translate
  gsave            .3333 dup scale vonKoch grestore
 1 add %reset order on the stack
}ifelse
}def
```



Von Koch-like fractal as Iterated Function System

Lauwerier(1994) in one of his exercises created a von Koch-like fractal by (linear) IFS (Iterated Function Systems), which consists of 2 contracted, affine transformations, L and R, which both for the von Koch fractal do contraction and mirroring. For the picture at right I just took 1000 points in order to expose the dot structure, in contrast with the line structure of the earlier approximations of the fractal.



M.F. Barnsley(1988) Fractals Everywhere, exploited contracted IFS.[4]

Most important property: with each contracted IFS is associated a limit figure, the fractal attractor.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{L}{=} \begin{pmatrix} a & b \\ b & -a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a-1 \\ b \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{R}{=} \begin{pmatrix} c & d \\ d & -c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1-c \\ -d \end{pmatrix}, \ a = .5,\ b = .289,\ c = a,\ d = -b.$$

or, after substitution of the parameters

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{L}{=} \begin{pmatrix} .5 & .289 \\ .289 & -.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -.5 \\ .289 \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \stackrel{R}{=} \begin{pmatrix} .5 & -.289 \\ -.289 & -.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} .5 \\ .289 \end{pmatrix}$$

Associated with the von Koch fractal are 2 rotations with mirroring with centres for L: (-1,0) and for R: (1,0) and contraction $\sqrt{.5^2 + .289^2} \approx .58$. Amazing, isn't it! Laurier's BASIC program `FRACMC4` and my transcription are given below. (MC is abbreviation for Monte Carlo, meaning alternate L and R by gambling.)

```
REM ***iteratief systeem, 2 spiegelingen, FRACMC4***
REM ***coefficienten***
A=.5 : B=..289 : C=A : D=-B
DET1=A*A+B*B : DET2=C*C+D*D : Q=DET1/(DET1+DET2)
X=1 : Y=0 : K=0 : KMAX=1000
DO WHILE K<KMAX AND INKEY$=" "
   R=RND
   IF R<Q THEN
      X1=A*X+B*Y-1+A : Y1=B*X-A*Y+B 'spiegeling L
   ELSE
      X1=C*X+D*Y+1-C : Y1=D*X-C*Y-D 'spiegeling R
   END IF
   X=X1 : Y=Y1
   PSET (X,Y),10
   K=K+1
LOOP : BEEP
END
```

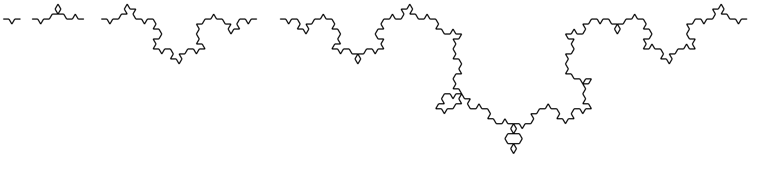
```
Other values of the parameters
a=.5 b=.5   c=.6667 d=0     %bebladerde tak
a=.5 b=.289 c=.5    d=-.289 %von Koch
a=.5 b=.5   c=.5    d=0     %kale tak
a=.5 b=.5   c=.6    d=-.2
a=0  b=.64  c=0     d=-.64  %tegelpatroon
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: fracmc4
%%Author: H.A. Lauwerier(1994): Spelen met Graphics en Fractals
%%Transcriptor: Kees van der Laan, febr 2012
%%BoundingBox: -100 -1 103 60
%%BeginSetup
%%EndSetup
%%BeginProlog
/Courier 7 selectfont
/x 1 def /y 0 def /halfmaxint 2 30 exp def/maxint 2 31 exp 1 sub
def
/a .5 def  /b .289 def /c a def /d b neg def
/det1 a dup mul b dup mul add def /det2 c dup mul d dup mul add
def
/q det1 det1 det2 add div def
/printxy {x s y s moveto (.) show}def
%%EndProlog
10 srand%10 is seed
/s {100 mul }def%scaling
1000{rand maxint div q lt
     {/xnew a x mul b y mul add 1 sub a add def
       /y b x mul a y mul sub b add def /x xnew def%mirror L
     }
     {/xnew c x mul d y mul add 1 add c sub def
      /y d x mul c y mul sub d sub def /x xnew def%mirror R
     }ifelse
     printxy
 }repeat
showpage
%%EOF
```

Deterministic von Koch and randomness

Peitgen c.s.(2004) mentions the deterministic von Koch fractal combined with randomness, and states that a better model for coastlines is obtained.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: von Koch Random fractal, 2012
%%Author: Kees van der Laan, kisa1@xs4all.nl
%%BoundingBox: -5 -125 650 15
%%BeginSetup
%%EndSetup
%%BeginProlog
/vonKoch{dup 0 eq
{0 0 moveto s 0 lineto currentpoint stroke translate}
{1 sub       vonKoch
 pm{  60 rotate vonKoch
    -120 rotate vonKoch
      60 rotate vonKoch}
   { -60 rotate vonKoch
     120 rotate vonKoch
     -60 rotate vonKoch}ifelse
1 add
}ifelse
}def
22121943 srand
/pm{rand 1073741823 gt{true}{false}ifelse}def
%%EndProlog
%
%Program ---the script---
/s 5 def        % s = initial size of line piece
                    1 vonKoch  pop % order 1
2 s mul 0 translate 2 vonKoch  pop
3 s mul 0 translate 3 vonKoch  pop
showpage
```



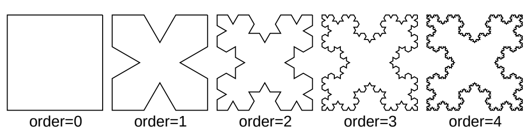
KRONKEL is Lauwerier's universal program to construct fractal islands based on similarity transformations.

```
10 REM ***KRONKEL: Fractal polygonal Island and
model***
10 DIM x(4096), Y(4096)
20 U=4 : DIM A(U), B(U) : REM ***Number of sides of Island***
30 V=4 : DIM C(V), D(V) : REM ***Number of pieces of model***
40 DATA 1,1,-1,1,-1,1,-1,-1,1,-1,1,1 : REM ***Corners island***
50 DATA .3333,0,.5,.2887,.667,0      : REM ***Data model***
60 INPUT P : REM ***Choice order***
70 FOR I=0 TO U   : READ A(I), B(I) : NEXT I
80 FOR I=1 TO V-1 : READ C(I), D(I) : NEXT I
90 REM ***Calculation coordinates Kronkel line***
100 C(0)=0 : D(0)=0 : x(0)=0 : Y(0)=0 : X(v^P)=1 : Y(V^P)=0
110 FOR I=0 TO P-1
120    FOR J=0 TO V^P-1 STEP V^(P-I)
130       M1=J+V^(P-I) : DX=x(M1)-X(J) : DY=Y(M1)-Y(J)
140       FOR K=1 TO V-1
150           M2=J+K*V^(P-I-1)
160           x(M2)=DX*C(K)-DY*D(K)+X(J)
170           Y(M2)=DY*C(K)+DX*D(K)+Y(J)
180       NEXT K
190    NEXT J
200 NEXT I
210 REM ***DRAW ISLAND***
220 PSET(A(0),B(0))
230 FOR M=0 TO U-1
240     DA=A(M+1)-A(M)
250     DB=B(M+1)-B(M)
260     FOR N=0 TO V^P
270         LINE -(DA*X(N)-DB*Y(N)+A(M), DB*X(N)+DA*Y(N)+B(M))
280     NEXT N
290 NEXT M
300 END
```

Length of size of line-piece is wired-in.

In PS
s scaling
u a b size and corners of island
v c d size and corners of model line
have been used as globals, and could have been initialized within the dictionary
No fixed bounds on x and y

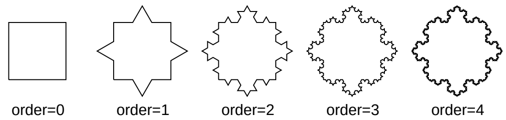
```
%!PS-Adobe-3.0 EPSF-3.0
%%Titel: Kronkel ---Koch Island--- H.A. Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl, April 2011
/kronkel%order p ==> fractal island (globals s, u, a, b, v , c, d)
{kronkeldict begin%push kronkeldict on the d-stack
 /p exch def
/x u v p exp mul cvi array def /y u v p exp mul cvi array
def%auxiliaries
%calculate coordinates corners `kronkel'
 x 0 0  put             y 0 0 put
 x v p exp cvi 1   put y v p exp cvi 0 put
 0 1 p 1 sub{/i exch def %for i
  0 v p i sub exp  v p exp 1 sub{/j exch def %for j
   /m1 j v p i sub exp add def
   /dx x m1 cvi get x j cvi get sub def
   /dy y m1 cvi get y j cvi get sub def
   1 1 v 1 sub{/k exch def %for k
    /m2 j k v p i sub 1 sub exp mul add def
x m2 cvi dx c k get mul dy d k get mul sub x j cvi get add puty m2
cvi dy c k get mul dx d k
get mul add y j cvi get add put }for%k
  }for%j
 }for%i
%create path for each side m of base line
 a 0 get b 0 get moveto
 0 1 u 1 sub{/m exch def%for m
  /da a m 1 add cvi get a m cvi get sub def
  /db b m 1 add cvi get b m cvi get sub def
  0 1 v p exp{/n exch def%for n
   da x n cvi get mul db y n cvi get mul sub a m cvi get add
db x n cvi get mul da y n cvi get mul add b m cvi get add lineto
  }for%n
 }for%m
end}def %end pops kronkeldict off the d-stack
/kronkeldict 8 dict def
%
/s {50 mul} def %scale
/u 4 def %number of corners of the island
/a [ 1 s -1 s -1 s 1 s 1 s ] def %x coordinates of corner points
/b [ 1 s 1 s -1 s -1 s 1 s ] def %y coordinates of corner points
/v 4 def %number of line pieces of the model line
  /c [ 0 .3333  .5    .6667  ] def%x coordinates of corners
  /d [ 0 0      .2887 0      ] def%y coordinates of corners
2 kronkel stroke showpage
%%EOF
```



inward islands

His islands are based on similarity transformations, not on the calculation of the direction of the next line piece as in the line fractals. The `Kronkel` program can also be used for degenerated islands, i.e. for line fractals, such as Lévy, von Koch, Minkowski, …

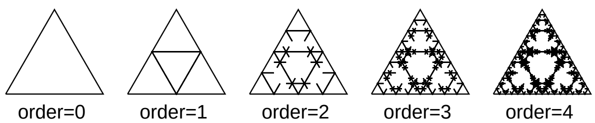
The order of specifying the corners of the island determines whether the fractal is drawn inside (anti-clockwise specification) or outside (clockwise specification)



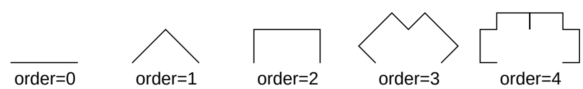
outward islands

```
...
(C:\\PSlib\\PSlib.eps) run
%globals s, u, a, b, v, c, d
/s {30 mul} def %scale
/u 4 def
/a [-1 s -1 s 1 s 1 s -1 s] def %abcissae corners island (clockwise=>inside, scaled)
/b [-1 s 1 s 1 s -1 s -1 s] def %ordinates corners island (clockwise=>inside, scaled)
/v 4 def
/c [0 .3333 .5    .6667 ] def %abcissae corners broken model line
/d [0 0     .2887 0 ]     def %ordinates corners broken model line
0 kronkel stroke gsave
110 0 translate 1 kronkel stroke grestore
showpage
```

A triangular island (0, 0), (1, 0) (.5, .866) (0, 0) can equally-well be specified, with the broken model line (0, 0), (.5, 0), (.375, .2165), (.5, 0), (.625, .2165), (.5, 0).



The degenerate Lévy island can be specified by the line (-1, 0), (1, 0), with the (broken) model line (0, 0), (.5, 0)




```
%...
(C:\\PSlib\\PSlib.eps) run
%globals s, u, a, b, v, c, d
/s {30 mul} def
/u 1 def /a [ -1 s 1 s ] def%abcissae corners line (scaled)
         /b [  0   0   ] def%ordinates corners line (scaled)
/v 2 def /c [ 0  .5] def    %abcissae corners broken model line
         /d [ 0  0 ] def    %ordinates corners broken model line
               0 kronkel stroke
110 0 translate 1 kronkel stroke
%...
showpage
%%EOF
```

An interesting program to experiment with. The PS transcription is also included in my `PSlib`. Lauwerier provides moreover variants: `KRONKEL`**T**, biased by the number system (Dutch **t**alstelsel) approach and `KRONKELB`, where **b**acktracking has been used.

Minkowski fractal

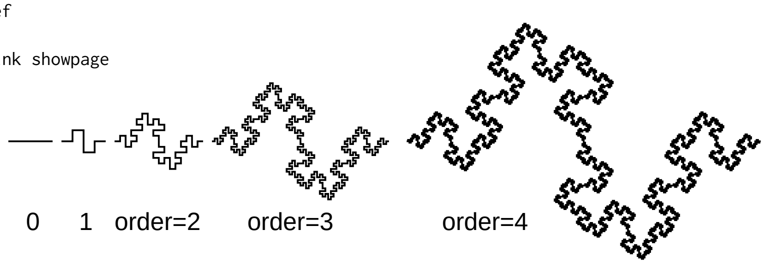
Much similar to the von Koch fractal is the Minkowski fractal, called sausage by Mandelbrot. The replacement scheme can be distilled from the illustration below, especially $M_0 \to M_1$.

```
10 REM ***Sausage of Minkowski***
10 DIM A(7) : A(0)=0 : A(1)=1 : A(2)=0 : A(3)=3
20           A(4)=3 : A(5)=0 : A(6)=1 : A(7)=0
30 P=3 : DIM T(P) : REM***order***
20 H=4^(-P) : X=0 : y=0: PSET (0,0)
30 FOR N=0 TO 8^P-1
40 M=N : FOR L=0 TO P-1
50       T(L)=M MOD 8 : M=M\8 : NEXT L
```

```
60 S=0 : FOR K=0 TO P-1
70       S=(S+A(T(K))) MOD 4: NEXT K
80 IF S=0 THEN X=X+H
81 IF S=1 THEN Y=Y+H
82 IF S=2 THEN X=X-H
83 IF S=3 THEN y=y-H
90 LINE -(X, Y)
91 NEXT N
92 END
```

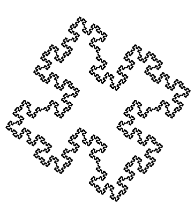
Length of line-piece is wired-in.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Minkowski sausage by H.A Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl,m April 2011
/mink{/p exch def%order, global l length of initial line
/a [ 0 1 0 3 3 0 1 0] def%model specification by direction numbers
/t p array def 0 0 moveto
/h l 4 p exp div def
0 1 8 p exp 1 sub{/m exch def
 0 1 p 1 sub{t exch m cvi 8 mod put /m m cvi 8 idiv def}for
 /s 0 def
 0 1 p 1 sub{/s a t 4 -1 roll get get s add cvi 4 mod def}for
 s 0 eq { h     0     rlineto}if
 s 1 eq { 0     h     rlineto}if
 s 2 eq { h neg 0     rlineto}if
 s 3 eq { 0     h neg rlineto}if
}for stroke} def
%
/l 100 def 3 mink showpage
%%EOF
```



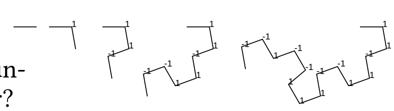
The fractal dimension of the Minkowski fractal $D = \frac{\log 8}{\log(1/4^{-1})} = 1.5$. The array a contains the direction numbers: 0, 1, 3, meaning direction 0°, 90°, -90°, respectively.

Minkowski island The essentials of the island program in PS are given below.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Minkowski island
%%...
/l 200 def                          4 mink
gsave l 0     translate  -90 rotate 4 mink grestore
gsave l l neg translate -180 rotate 4 mink grestore
      0 l neg translate -270 rotate 4 mink
showpage
%%EOF
```

Dragon figures



Folding a strip of paper repeatedly and after unfolding one may ask: How to draw the meander?

The curve with rounded 90° corners is named Dragon curve by Heighway. The curve does not intersect itself. A nice example for developing the mathematical problem solving attitude in discovering the intriguing pattern. (Be aware of folding consistently in the right direction.)

Let us set up a table, where for each line piece the continuation angle is given: r means rotate $-90°$, and l means rotate $90°$, and unearth the regularity in the directions $d(n)$, for $n = 1, 2, 3, \ldots$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
r	r	l	r	r	l	l	r	r	r	l	l	r	l	l	r	r	r	l	r	r	l	l	l	r	r	l	l	r	l	l	r

$d(n) = r(ight)$ for $n = 1, 5, 9, \ldots$ $\quad d(n) = l(eft)$ for $n = 3, 7, 11, \ldots$ $\quad d(n) = d(n/2)$ for n is even.

Express n in the form $k \times 2m$ where k is an odd number. The direction of the n^{th} turn:

if $k \,\mathbf{mod}\, 4 = 1$ then the n^{th} turn is r;

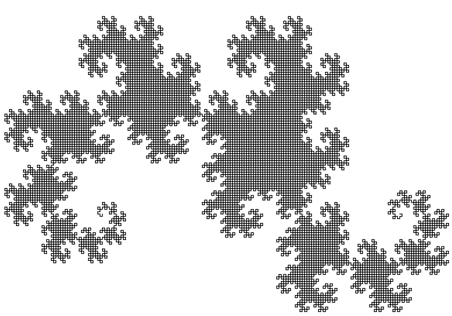
if $k \,\mathbf{mod}\, 4 = 3$ then the n^{th} turn is l.

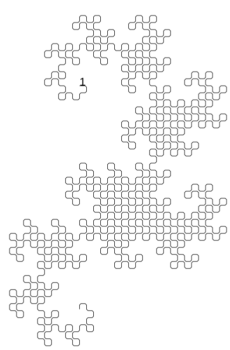
The direction of turn 76376: $76376 = 9547 \times 8$ & $9547 \,\mathbf{mod}\, 4 = 3 \rightarrow d(76376) = l$.

```
10 REM ***Draak***
10 P=3 : REM***order***
20 H=2^(-P/2) : A=1.7453 : REM ***Corner***
30 B=PI-A : X=H : Y=0 : LINE (0,0)-(H,0) : S=0
40 FOR N=1 TO 2^P-1 : M=N
50 IF M MOD 2 = 0 THEN M=M/2: GOTO 50
60 IF M MOD 4 =1  THEN D=1 ELSE D=-1
70 S=S+D
80 X=X+H*COS(S*B)
90 Y=Y+H*SIN(S*B) : LINE -(X,Y)
91 NEXT N
92 END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Draak by H.A Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl, April 2011
/draak{%draak curve with angle A and order p, global scaling s
       %angle p ==> dragon curve
{/p exch def /b 180 3 -1 roll sub def%order p and angle b
/h 2 p -2 div exp s def %size piece scaled
0 0 moveto h 0 lineto
1 1 2 p exp 1 sub{%for n
 /m exch def
 {m cvi 2 mod 0 eq {/m m 2 div def}{exit}ifelse}loop
  m cvi 4 mod 1 eq {/d b neg def}{/d b def}ifelse
  d rotate h 0 rlineto%no currentpoint translate necessary
}for %n
}bind def %in library for local variables with draakdict
%
/s {30 mul} def 100 3 draak stroke showpage
%%EOF
```

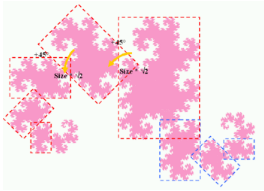
For the order p =14 and angle 90° I reproduced Lauwerier's result in PS, see below at left.



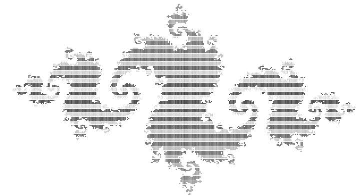


The number of line pieces is 2^p. The curve of order 10 with rounded corners is at right. The curves don't intersect themselves, which is seen in the figure with rounded corners. (In Lauwerier's program the direction `D` is not in agreement with the folded paper and the dragon figure. This is adapted in the PS code.)

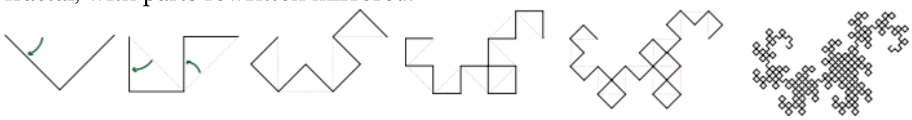
Knuth in the TEXbook Appendix D p390 also mentions the dragon curve in relation to Turtle Graphics, and draws dragon figures in TEX. When I tried the order 12 in TEX, in 1995, TEX gave the error message 'TEX capacity exceeded.'



Dimensions The bounding box obeys the proportion 3 : 2. The fractal dimension of the curve equals 2, a local plane filling curve (Courtesy http://en.wikipedia.org/wiki/Dragon_curve, which mentions more properties of the Dragon curve, such as its self-similarity and the spiral shape.)
At right a filled dragon-like Julia set.(More on Julia fractals: JULIA fractals in PostScript, submitted for MAPS.)

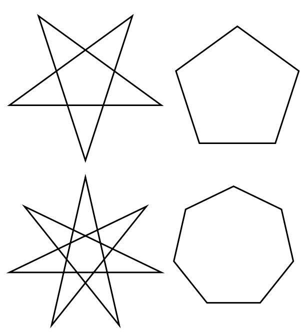


The Dragon curve can be generated similarly to the rewriting scheme of the Lévy fractal, with parts rewritten mirrored.



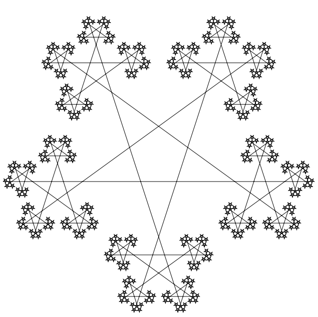
Star fractals

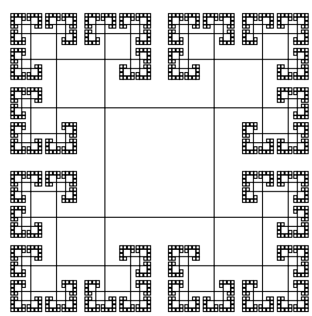
As introduction a generalization of the program star of the Blue Book p51. The program is more general because it allows to draw the pentagram or the 5-star depending on the value of the angle parameter. Moreover, the number of vertices can be varied, to obtain for example a heptagon casu quo 7-star (heptagram).



```
/gonstar%p (order) v ==> star
{gonstardict begin /v exch def /angle exch def
0 0 moveto
v{angle rotate l 0 rlineto}repeat closepath
end} bind def
/gonstardict 2 dict def
%%EndProlog
%
%Program ---the script---
%
/l 100 def   144  5 gonstar stroke
gsave 75 -25 translate
/l 50 def 1.415 setmiterlimit
72 5 gonstar stroke grestore
gsave 0 -110 translate
/l 100 def   1080 7 div  7 gonstar stroke
grestore
gsave 65 -130 translate
/l 35 def 1.415 setmiterlimit
360 7 div 7 gonstar stroke
showpage
```

Lauwerier's ingenious, concisely programmed star fractal illustrations, left and right below, consist also of 1 (broken) line.





```
10 REM ***Star***
10 P=5 : REM***order***
20 V=4: A=.8*3.141593 : R=.35
30 PSET (0,0) : X=0 : Y=0
40 FOR N=0 TO (V+1)+V^(P-1)-1
50    M=N : B= N*A : F=0
60    IF M MOD V <> 0 OR F>=P-1 THEN GOTO 80
70      F=F+1 : M=M\V : GOTO 60
80    X=X+R^(P-F-1)*COS(B)
90    Y=Y+R^(P-F-1)*SIN(B)
100   LINE -(X,Y)
110 NEXT N
120 END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Starfractal by H.A Lauwerier
%%Transcriptor: Kees van der Laan, kisa1@xs4all.nl, May 2011
/starfractal%reduction angle p (order) v ==> starfractal
{starfractaldict begin
 /v exch cvi def /p exch def /a exch def /r exch def
 0 0 moveto
 0 1 v 1 add v p 1 sub exp mul 1 sub
   {/n exch cvi def
    /m n def /f 0 def
    {m v mod 0 ne   f p 1 sub ge   or
     {exit}
     {/f f 1 add def /m m v idiv def}
     ifelse
    }loop
    r p f sub 1 sub exp s 0 rlineto
    a rotate
   }for %n
 end} bind def
/starfractaldict 8 dict def
%
starfractaldict begin /s {300 mul} def end % scaling
.3 144 3 4 starfractal fill % r=.3 a=144 order=3 vertiges-1=4
showpage
%%EOF
```

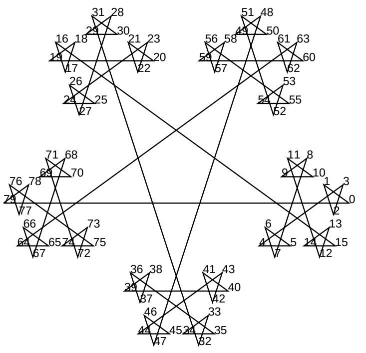
Remarks

scaling factor has been added in PS
user space is rotated by the angle after each line
parameter driven

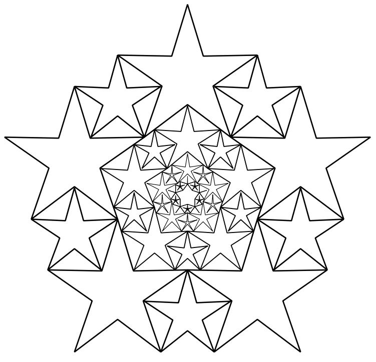
The algorithm is based on that consecutive line pieces make a constant angle, only the line size varies. For the order 5 we have 5 different lengths of the line pieces

1	n=0, 256, 512, 768, 1024, ...
r	n=64, 128, 192, **320**, 384, 448, ...
r^2	n=16, 32, 48, **80**, 96, 112, ...
r^3	n=4, 8, 12, **20**, 24, 28, **36**, 40, 44, **52**, ...
r^4	n=1, 2, 3, **5**, 6, 7, **9**, 10, 11, **13**, ...

In order to follow the way the drawing has been made sequential numbers have been added in the accompanying illustration.



Another 5-star composition has been published in my Tiling in PS and Metafont in MAPS 97.2. I copied the program from the article, adapted it to EPSF, et voilà. In the middle a composition borrowed from Helmstedt created by a Lindenmayer production rule in Mathematica. At right a nice illustration from Lauwerier(1990), which reminds me of Escher's limit cycles.

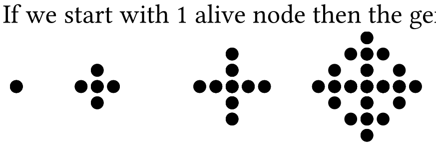


Game of Life

Lauwerier(1990) mentions a fractal which he obtained from the Pickover variant of the Game of Life, made popular by Martin Gardner in a Scientific American in 1970. The game is played on a grid. Each node can be alive or dead. Once alive it stays alive. If dead it comes to life if only one neighbour, N, E, S or W is alive. On each heartbeat the whole grid is inspected in parallel. Lauwerier's BASIC program is given below.

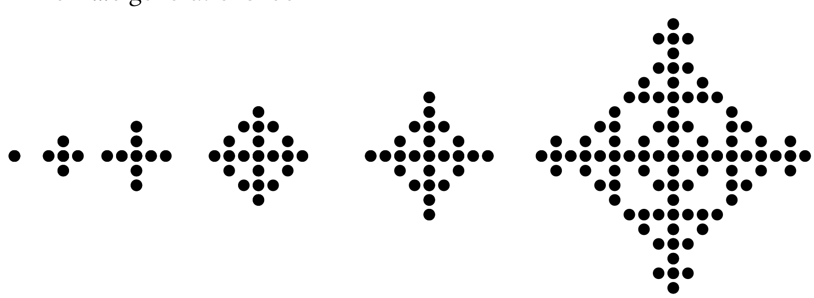
```
***naam: PICK1***
40 DEFINT I, J, K, N, T, X, Y
70 IF SCR=9 THEN XM=320 : YM=175
80 IF SCR=12 THEN XM=320 : YM=240
100 INPUT "NUMBER OF ROWS=", N
120 DIM X(N,N), Y(N,N)
130 X(0,0)=1
140 FOR K=1 TO N-1
150 FOR I=0 TO K : FOR J=0 TO K-I
160 IF X(I,J)=0 THEN GOSUB 220 ELSE GOSUB 270
170 NEXT J : NEXT I
180 FOR I=0 TO K : FOR J=0 TO K-I
190 X(I,J)=Y(I,J)
200 NEXT J : NEXT I : NEXT K
210 A$=INPUT$(1) :END
220 IF I>=1 AND J>=1  THEN T=X(I+1,J)+X(I-1,J)+
                             X(I,J+1)+X(I,J-1)
230 IF I=0 AND J>=1  THEN T=2*X(1,J)+
                             X(0,J+1)+X(0,J-1)
240 IF I>=1 AND J=0  THEN T=2*X(I,1)+
                             X(X+1,0)+X(I-1,0)
250 IF T=1 THEN Y(I,J)=1
260 RETURN
270 PSET (XM+2*I, YTM-2*J),14 : PSET (XM-2*I, YM-2*J),14
280 PSET (XM+2*I, YTM+2*J),14 : PSET (XM-2*I, YM+-2*J),14
290 RETURN : END
```

If we start with 1 alive node then the generations 1, 2, 3, 4 look as follows

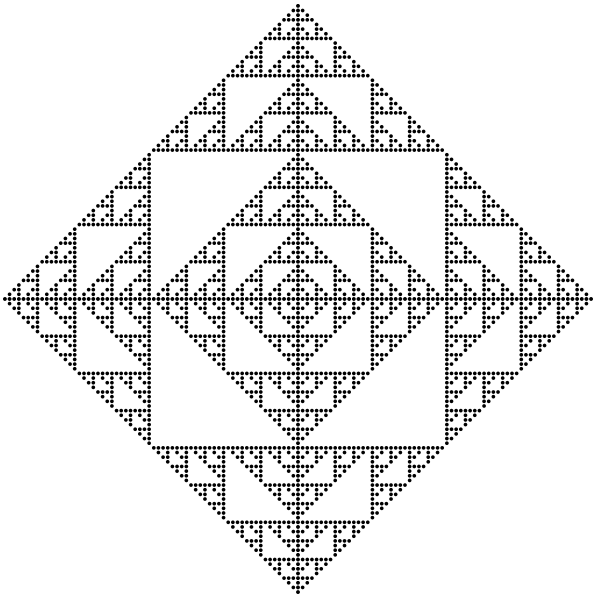


Lauwerier's program is computational intensive.
My translated version could not reproduce Lauwerier's result in reasonable time.
I simplified the program by inspecting on each heartbeat only the new contra diagonals ($i + j =$constant) in the first quadrant.

The 1 ...6 generations look



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Growth Cell model a la Pickover, simplified
%%Author:  Kees van der Laan
%%Date: March 2012
%%BoundingBox: -195 -195 195 195
%%BeginSetup
%%EndSetup
%%BeginPrologue
%%DocumentFonts: Times-Roman
/Times-Roman 20 selectfont
/printdot{3 i mul -3 j mul moveto (.) show
          3 i mul +3 j mul moveto (.) show
         -3 i mul -3 j mul moveto (.) show
         -3 i mul +3 j mul moveto (.) show
         }def
/alive?{%check whether cell has become alive
  i 1 ge j 1 ge and
    {ax i 1 sub n mul j       add get
     ax i       n mul j 1 sub add get add}if
  i 0 eq j 1 ge and
    {2 ax  n  j   add get mul
       ax     j 1 sub get add}if
  i 1 ge j 0 eq and{ax i 1 sub n mul get}if
  1 eq{ax i n mul j add 1 put printdot}if
}def%alive?
/pickover{% stack integer>=0 the order==> cell pattern
/n exch def
/ax n 1 add dup mul array def                    %array
1 1 n n mul{/k exch def ax k 0 put}for ax 0 1 put%initialize
/i 0 def /j 0 def printdot
1 1 n 1 sub {/k exch def
  0 1 k{/i exch def /j k i sub def%contradiagonal
        alive?
       }for%i
  }for%k
}def
%%EndProlog
%
% Program
%
64 pickover showpage
%%EOF
```



The point Lauwerier wanted to make — the game yields fractal patterns — is also obtained by this simplified game.

Annotated References

- An introductory survey: `http://en.wikipedia.org/wiki/Dragon_curve`.
- Adobe Red, Green and Blue Books. The musts for PS programmers.
- Biography of H.A. Lauwerier: `http://bwnw.cwi-incubator.nl/cgi-bin/uncgi/alf`.
- Gleisk, J(1987): CHAOS — making a new science. Penguin.
  (An introduction to and survey of the world of non-linearity, strange attractors and fractals.)
- Goossens, M(2007, sec ed) et. al.: LaTeX Graphics Companion. ISBN 978 0 321 50892 8.
- Helmstedt, J(2011): A New Method of Constructing Fractals and Other Graphics. The Mathematica Journal. (Nice examples of Lindenmayer systems, for which Lauwerier's KRONKEL can be used.)

- Jackowski, B, P. Strelczyk, P. Pianowski(1995-2008): PSView5.12. WWW. `bop@bop.com.pl`. (Extremely fast previewer for `.eps` among others, which allows `PSlib`(rary) inclusion via the `run` command).
- Knuth, D.E, T. Larrabee, P.M. Roberts(1989): Mathematical Writing. MAA notes 14. The Mathematical Association of America.
- Knuth, D.E(1990, 10th printing): The TEXbook. Addison-Wesley. ISBN 0-201-13447-0. (A must for plain TEXies.)
- Lauwerier, H.A(1987): FRACTALS — meetkundige figuren in eindeloze herhaling. Aramith. (Contains programs in BASIC. Lauwerier, H.A (1991): Fractals: Endlessly Repeated Geometrical Figures, Translated by Sophia Gill-Hoffstadt, Princeton University Press, Princeton NJ1. ISBN 0-691-08551-X, cloth. ISBN 0-691-02445-6 paperback. "This book has been written for a wide audience ... " Includes sample BASIC programs in an appendix. Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1988): The Pythagoras Tree as Julia Set. CWI-Newsletter.
- Lauwerier, H.A(1989): Oneindigheid — een onbereikbaar ideaal. Aramith. ISBN 90 6834 055 7. (Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1990): Een wereld van FRACTALS. Aramith. ISBN 90 6834 076 X. (Sequel and updated version of Lauwerier(1987). Audience: Instructors, (high-school) students, and the educated layman.)
- Lauwerier, H.A(1994): Spelen met Graphics and Fractals. Academic Service. ISBN 90 395 0092 4. (An inspiring book with Math at the high school level for a wide audience; the BASIC programs I consider outdated for direct use. Audience: Instructors, (high-school) students, and the educated layman.)
- Peitgen, H.O, H.Jürgens, D. Saupe(2004 sec. ed.): Chaos and Fractals. New frontiers of Science. (Images of the fourteen chapters of this book cover the central ideas and concepts of chaos and fractals as well as many related topics including: the Mandelbrot set, Julia sets, cellular automata, L-systems, percolation and strange attractors. This new edition has been thoroughly revised throughout. The appendices of the original edition were taken out since more recent publications cover this material in more depth. Instead of the focused computer programs in BASIC, the authors provide 10 interactive JAVA-applets for this second edition via `http://www.cevis.uni-bremen.de/fractals`. An encyclopedic work. Audience: Accessible without mathematical sophistication and portrays the new fields: Chaos and fractals, in an authentic manner.)
- Swanson, E(1986, revised ed): Mathematics into Type. American Mathematical Society.
- Szabó, P(2009): PDF output size of TEX documents. Proceedings EuroTEX2009/ConTEXt, p57–74. (Various tools have been compared for the purpose.)
- Van der Laan, C.G(1992): LIFO and FIFO sing the Blues. MAPS 92.2.
- Van der Laan, C.G(1995): Publishing with TEX. Public Domain. (See TEX archives. `BLUe.tex` comes with `pic.dat` the database of my pictures in TEX-alone.)
- Van der Laan, C.G(1997): Tiling in PostScript and MetaFont — Escher's wink. MAPS 97.2.
- Van der Laan, C.G(unpublished, BachoTEX workshop): TEXing Paradigms. (A plea is made for standardized macro writing in TEX to enhance readability and correctness.)
- Van der Laan, C.G(submitted EuroTEX2012): Julia fractals in PostScript.
- Veith, U(2009): Experiences typesetting mathematical physics. Proceedings EuroTEX2009/ConTEXt, p31–43. (Practical examples where we need to adjust TEX's automatic typesetting.)

Conclusions

It was pleasure, educative and inspiring to read Lauwerier's booklets. Some of his algorithms have found a wider audience by converting his BASIC codes into PostScript, hopefully.

I don't know how to include the results of the BASIC programs elegantly in publications. The results of the PS programs can be easily included in pdf(La)TeX, Word, ... documents.

PS' variable user space and recursion alleviated programming, with concise `defs` and programs as readable as literature, but ... be aware of its subtleness. PostScript's variable user space was the key to my adaptation of production rules. Because of PS' subtleness not many people program in PS, I presume, or ... do they consider it of too low-level?

In programming self-similarity the awareness of orientation is paramount. I did not find classical Math fractals in PS on the WWW, only one Sierpiński curve in Java.

Lauwerier's analysis — associating binary, ternary, ... tree structures with binary, ternary, ... numbers, is an eye-opener. In his, and my, programs all the self-similar sub-curves are draw anew. In Metafont/-Post we could just build the paths and splice them suitably into paths of higher order, as I did in the past with the Pythagoras Tree in Metafont.

'Het Wiskunde boek' states that fractals have renewed and raised interest in Mathematics.

Before publishing consult the Wikipedia on aspects of the subject as well as Wolfram's knowledge base `http://www.wolframalpha.com`.

TEX mark up For the symbols of the number systems I, N, Q, R, C, which curiously are not provided for in plain TEX, I use the the AMS (blackboard) font `msbm10`.

The $\stackrel{\rm L}{=}$ and $\stackrel{\rm R}{=}$ composed relational operators are marked up by `\mathrel {\mathop =^{\rm L}}` and not by `$\buildrel\rm L \over=$`, TEXbook p437; the latter is OK for the stacked composed symbol as such.

For typesetting tables `\halign` and the tabbing mechanism have been used (TEXbook ch22). The 11-element of one of the tables needs an oblique line. I provided for this in PS, which is simpler and not restricted by obliqueness. In 1995, in my PWT guide, I used the GKP macros for this, which suffer from the same inconvenience as LATEX's picture environment: restricted obliqueness.

A blank line before display math yields too much white space! This blank line is important, though, in order to avoid widows.

Locally I have used for parallel listings of program texts `vbox-s` next to each other, which inhibits proper page breaks. I don't know how to provide macros for local elegantly marked up multi-column texts, which allow page breaks. (I also tried `\valign`, alas in vain.) My inserted pictures suffer from the same inconvenience as in Word: changing the text might disturb the layout, such that the pictures will become ill-placed.

As known, I could not use footnotes from within a `vbox`; kludged around.

In TEXworks I used the Terminal font in the edit window with the pleasing effect that comments remain vertically aligned in the `.pdf` window.

Conversion of my TEX script into Word made me (hands-on) aware of the differences between TEX and Word. If you are after utmost accurate, user-controlled typeset Mathematics then TEX is to be preferred. For bread and butter Mathematics Word can do, especially with Cambria, I presume. I did not find in Word (MS equation 3.0) the possibility to discern between displayed Math and in-line Math. Tables in Word are tricky, the WYSIWYG approach does not always yield the table layout you are after. As in TEX I can't make an appropriate 11-element. I could not handle the inclusion of a PS made 11-element in Word. Program texts, as columns in a table, don't suffer from difficulties in allowing page breaks. A pre-index, as is usual with hyper-geometric functions, I could not nicely typeset with MS equation 3.0.

Inclusion of the `.jpg` figures and `.pdf` objects went smoothly. I had to convert `.png` objects. The `inclusion of EPSF object` option did not work on my PC, though the option is available. It invoked Adobe Illustrator CS2 12.0.0 and fell silent. The same EPSF invoked by AI directly worked. Maybe incompatible versions? Neglecting superfluous spaces, which TEX does automatically, has been lost in the conversion. A local change in Word might change the document more than local, beyond user control. I don't know how to switch off, or change, pre-settings, such as: don't underline automatically WWW addresses, maybe by de-activating the option `WWW addresses as hyperlinks`?

Conversion also entailed splitting up the original (concept) paper and rewriting the parts into 2 new papers. Converting back into TEX, after changes were made, was more difficult than converting into Word.

After I had finished I became aware of Acrobat Pro X which also converts `.pdf` into a Word document.

Acknowledgements

Thank you Adobe for your maintained, adapted to LanguageLevel 3 since 1997, good old, industrial standard PS and `Acrobat Pro` (actually DISTILLER) to view it, Don Knuth for your stable plain TEX, Jonathan Kew for the TEXworks IDE, Hàn Thế Thành for pdf(La)TEX, Hans Lauwerier for your nice booklets with so many inspiring examples of fractals.

Thank you Jos Willink for proofing, Wim Wilhelm for drawing my attention to cellulaire automata, and the LATEX graphics environment `asymptote` which he integrated in TEXnicCenter; I don't have experience with them, but they may be of interest for LATEX users. MAPS editors for improving my use of English and Taco Hoekwater for procrusting my plain TEX note into MAPS format.

Thank you Zinadia Nikolaevna Gulka for inviting me to submit a paper or two for the 'Informatsionnie Texnologii i Matematicheskoe Modelirovanie' journal', and her co-worker for stimulating me to convert the TEX marked up ASCII source of an early version of this note into the required Word. The invitation stimulated me to adapt and revise the material. Thank you GUST for publishing a previous version of this note in the BachoTEX2012 proceedings. Thank you Svetlana Morozova for prompting me in the use of Word.

IDE My PC runs 32 bits Vista, with Intel Quad CPU Q8300 2.5GHz assisted by 8GB RAM. I visualize PS with Acrobat Pro 7. My PS editor is just Windows 'kladblok (notepad).' I use the EPSF-feature to crop pictures to their BoundingBox, ready for inclusion in documents. For document production I use TEXworks IDE with the plain TEX engine, pdfTEX, with as few as possible structuring macros taken from my `BLUe.tex` — adhering minimal TEX markup. I use the Terminal font in the edit window with the pleasing effect that comments remain vertically aligned in the `.pdf` window.

For checking the spelling I use the public domain `en_GB` dictionary and hyphenation patterns `en_GB.aff` in TEXworks.

Prior to sending my `PDF`'s by email the files are optimized towards size by Acrobat Pro.

The bad news with respect to `.eps` into `.pdf` conversion is that the newest Acrobat 10 Pro X does not allow for the `run` command for library inclusion.

Notes

1. Alas, the `\psfig` has been lost in pdfTEX. Happily, ConTEXt and LuaTEX allow direct EPSF inclusion.
2. Lauwerier(1989) narrates what mathematicians thought about the ∞-concept through the ages from the ancient Greeks onward.
3. Acrobat Pro X does not allow for the library inclusion via `run`, alas :-(. BASIC is interactive, PS is batch-oriented.
4. Barnsley is famous for his fern fractal. In his later works he is more ambitious and constructs fractals given a picture, on demand.
5. The 'fixed-point' of the production rule is the fractal. At right my old TEX code is displayed with its result.
6. The 'fixed-point' of the production rule is the real fractal.

My case rests, have fun and all the best.

Kees van der Laan
Hunzeweg 57, 9893PB Garnwerd, Gr, NL
kisa1@xs4all.nl

Appendix: Fractal Dimension

The need arose to associate various fractal curves with numbers, to characterize them. Mathematicians came up with definitions which were generalizations and compatible extensions of the classical, topological dimension notion. The fractal dimension à la Kolmogorov(1958) is based upon covering the fractal with a grid and counting the cells of the grid which contain points of the fractal, the so-called box-counting dimension. The definition reads

$$D = \lim_{s\to 0} \mathbf{log} N/\mathbf{log}(1/s),$$

with N the number of cells which contain points of the fractal and s the size of the side of a cell. Let us calculate the (fractal) dimension of a square in order to verify the compatibility of the definition with the classical fact. Cover the (unit)square with a grid which has s as side of a cell, then we have $1/s^2$ cells which cover the unit square, and therefore $D = \mathbf{log}(1/s^2)/\mathbf{log}(1/s) = 2$, QED.

All plane-filling fractal curves have fractal dimension $D = 2$.

For the calculation of the fractal dimension of fractals defined by contracted mappings, with contraction factors f_1, f_2, f_3, ... Lauwerier(1990) mentions the Hausdorff formula

$$\sum_i f_i^D = 1.$$

For the Lévy fractal, which can be defined by 2 contractions each with contraction factor $1/\sqrt{2}$, we arrive at

$$2 * (1/\sqrt{2})^D = 1 \rightarrow\ D = 2.$$

For the von Koch fractal, which can be defined by 4 contractions each with contraction factor 1/3, we arrive at

$$4 * (1/3)^D = 1 \rightarrow\ D = \mathbf{log}4/\mathbf{log}3 \approx 1.26.$$

For the Sierpiski sieve, which can be defined by 3 contractions with contraction factors 1/2, we arrive at

$$3 * (1/2)^D = 1 \rightarrow\ D = \mathbf{log}3/\mathbf{log}2 \approx 1.58.$$

For the Menger sponge, which can be defined by 8 contractions each with contraction factor 1/3, we arrive at

$$8 * (1/3)^D = 1 \rightarrow\ D = \mathbf{log}8/\mathbf{log}3 \approx 1.89.$$

The details of the definition of the fractal dimension D by Hausdorff (1919) are cumbersome, and serve a theoretical need. The calculation of other fractal dimensions: self-similarity dimension, and compass dimension (for coast lines) goes beyond the scope of this paper, see Peitgen c.s.(2004). My purpose is that one knows about the concept as a grey box with nodding knowledge, that one knows some fractal dimensions for simple cases, and ... that one does not become frightened or confused on encounter.

For a survey of various fractals and their dimensions see Peitgen c.s.(2004) and/or `http://en.wikipedia.org/wiki/List_of_fractals_by_Haussdorf_dimension`.

Finally, do you know that the path of Brownian movement is a fractal with fractal dimension 2, and ... do you know that the Cantor Dust has Hausdorf dimension $D_H =$ log2/log3 $\approx .63$, which differs from the fractal dimension $D \approx .69$?

Appendix: Cantor Dusts

The Cantor Dust has been shown at the beginning of this note. The idea is that each line is replaced by its left and right third parts. This pattern can be repeated yielding what is called a fractal nowadays. (It is included in `pic.dat`, the library of (TEX-alone) pictures which come with `BLUe.tex`.)

In Publishing with TEX(1995) the Cantor Dust example by TEX alone was included.

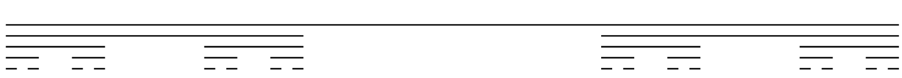
The PS code for Cantor Dust of order n has a similar production rule as for the von Koch fractal:

$$CD_n = [S_{\frac{1}{3}\frac{1}{3}} CD_{n-1}] \oplus [S_{\frac{1}{3}\frac{1}{3}} T_{\frac{2s}{3}0} CD_{n-1}] \quad \text{with}$$

CD_0 the initial line,
CD_n the Cantor Dust of order n,
$\oplus$ splice operator, meaning add properly the second piece to the set,
[open a new GS on the GS stack,
] remove current graphics state from the GS stack and recall previous,
$S_{a,b}$ means scale US by a and b, in x- and y-direction
$T_{a,b}$ means translate US by a and b, in x- and y-direction.
The above PS production rule transcribes systematically into the following PS `def`.[5]

```
%!PS-Adobe-3.0 EPSF-3.0
/cd%integer n>=0 ==> Cantor Dust of order n
{1 sub dup -1 eq
 {0 0 moveto s 0 lineto stroke}
 {gsave .3333 1 scale                      dup cd grestore
  gsave .3333 1 scale  2 s mul 0 translate dup cd grestore
 }ifelse 1 add
}def
%%EndProlog
%
% Program
%
%...
/s 300 def 4 cd pop
   305 -2  moveto TR10 setfont (Cantor) show
     0 -3 rmoveto TR7  setfont (4) show 0 -10 translate
%...
showpage
%%EOF
```

```
%Borrowed from BLUe's pic.dat
\newcount\x\newcount\y\newcount\width
\newdimen\unitlength
\def\E#1{\hbox to 0pt{\kern\x\unitlength
   \vbox to 0pt{\vss\hrule width#1\unitlength
                \kern\y\unitlength
               }\hss
   }\advance\x#1 }
\def\cf{\ifnum0=\width \fc\fi
  {\E{\the\width}}\divide\width3
   \advance\y-3
   {\cf}\advance\x\width
        \advance\x\width
   {\cf}\relax}%
\def\fc#1\relax{\fi}%
$$\width243 \unitlength1pt
  \x-\width \divide\x2
  \cf$$
```



The ‘fixed point’ of the production rule is the Cantor Dust fractal. For just showing a few approximations of the Cantor Dust the TEX code will do. Lauwerier(1987) provides a tiny BASIC program KAM. He also associates the Cantor dust with the trinary number system because the interval is divided in 3 pieces, repeatedly. The Cantor dust of [0, 1] consists of the trinary fractions where only the digits 0 and 2 occur, Lauwerier(1987, p26). An eye-opener!

Cantor Dust as IFS Lauwerier(1989, ch8) constructs the Cantor Dust of high order by the IFS

$$x_{n+1} \stackrel{L}{=} x_n/3 \qquad \text{and} \qquad x_{n+1} \stackrel{R}{=} x_n/3+2/3 \qquad n=0,1,2,\ldots \quad x_0=1/3.$$

Again an eye-opener! His tiny program and my conversion read

```
X=.3         'start
FOR K=1 TO 100
   IF RND<.5 THEN X=X/3 ELSE X=(X+2)/3
   PSET(X, 0)'scale X if wanted
NEXT K
END
```

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Cantor Dust via IFS
%%Author: H.A. Lauwerier(1989): Oneindigheid
%%Transcriptor: C.G. van der Laan, may 2012
%%BoundingBox: -1 39 1010 70
%%BeginSetup
%%EndSetup
%
% Program
%
/Courier 10 selectfont
22121943 srand /x .3 def%start
100{rand 1073741823 gt{/x x 2 add 3 div def}
                      {/x x 3 div def}ifelse
    x 1000 mul 50 moveto (.) show
   }repeat
showpage
%%EOF
```

If you, kind reader, shrug shoulders about paying so much attention to such a tiny problem, I can only say that if you don't analyse tiny problems deeply, your solutions of bigger problems will lack ingenuity.

Generalization to 2D yield the so-called Sierpiński carpets.

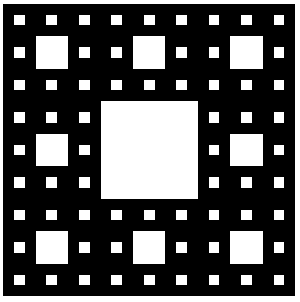
The algorithm reads: divide a square in 3^2 equal sub-squares and delete the middle, or the middle cross, and do this repeatedly for the remaining 8 squares.

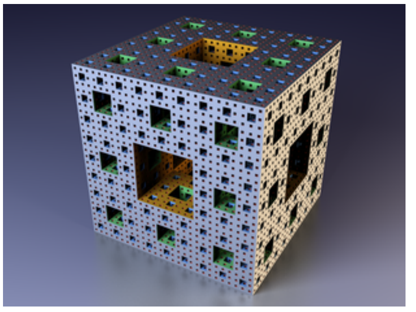
At the EuroTEX95 Bogusłav Jackowski showed his variant, probably in connection with his `mftoeps` program, which transforms Metafont code into PS. (I did not use `mftoeps`, because it was PC-biased, and I had a Mac Powerbook 150.) This was at the time that MetaPost, the preprocessor for PS with Metafont-biased user-language, was not yet released in the public domain.

In my opinion he was on the right track: PS is mandatory for graphics to be included in documents. It is just a pity he did not pursue that PS alone, or better EPSF, is suitable for constructing graphics to be included in documents. His collaborators, Pjotr and Pjotr, pursued PS in PSview.

I programmed the Sierpiński carpet, it is not the gasket, in TEX and in Metafont after the conference. For historical reasons I have included the programs below. The black-and-white figure is created by TEX-alone on-the-fly. (I just copied it from the revised 1996 FIFO-article, and see,... it still works. The MF program also still works on my museum Mac Powerbook 150 of 1995!)

The picture at right has been borrowed from the WWW. It illustrates a generalization of the good old Cantor Dust into 2.5D, also called Menger sponge. Both pictures are more interesting and more beautiful than the original 1D Cantor Dust.





```
tracingstats:=1;proofing:=1;screenstrokes; %
pickup pencircle scaled 1;
def sierpinskisquare (expr s, p)=
if s>5:unfill unitsquare scaled .333s
      shifted (p+.333s*(1,1));
   sierpinskisquare(.333s, p);
   sierpinskisquare(.333s, p+(.333s,0));
   sierpinskisquare(.333s, p+(.667s,0));
   sierpinskisquare(.333s, p+(0,.333s));
   sierpinskisquare(.333s, p+(0,.6673s));
   sierpinskisquare(.333s, p+(.667s,.333s));
   sierpinskisquare(.333s, p+(.667s,.667s));
   sierpinskisquare(.333s, p+(.333s,.667s));
fi enddef;
%
s=100; fill unitsquare scaled s;
sierpinskisquare(s,origin);
showit;
end
```

```
\newdimen\x\newdimen\y\newdimen\size\newdimen\lsize
\def\sier{\ifdim\size<35pt \reis\fi
   \divide\size3
  {\sier}{\advance\x\size\sier}{\advance\x2\size\sier}%
  {\advance\y\size{\sier}{\advance\x2\size\sier}}%
  \advance\y2\size{\sier}{\advance\x\size\sier}%
                      \advance\x2\size\sier}
\def\reis#1\sier{\fi\putatxy\draw}
%with auxiliaries
\def\putatxy#1{\vbox to0pt{\vss
   \hbox to0pt{\kern\x#1\hss}\kern\y}}
\def\draw{{\lsize\size\divide\lsize3
   \rlap{\vrule height\size width\lsize
   \vbox to\size{\hrule width\lsize height\lsize\vss
                  \hrule width\lsize height\lsize}%
      \vrule height\size width\lsize}}}
```

```
$$\vbox to120pt{\vss
  \offinterlineskip\size120pt\x=-.5\size\y0pt
  \sier}$$
```

A PS program for a 2.5D Cantor Dust is cumbersome, because it has to deal with projection and has to handle hidden lines.

Appendix: Hilbert Curve

Hilbert curves H_1 ... H_6 have been shown in the introduction. Wirth(1975) I consider a starting point for programming a Hilbert curve. Wirth could not know about the rotation of US facility, nor did PS exist. The self-similarity property of the H-curve, i.e. a curve is composed of (rotated) curves of one order lower, he programmed by four rotated instances in (recursive) procedures A, B, C and D, which entailed a more complex recursion scheme. For those who don't own the 1975 book I have included the program and the procedure A. (Procedures B, C and D are similar.)

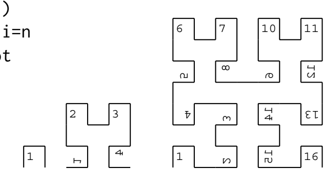
The left PS-program below is based on the production rule[6]

$$H_i = {}_mH^{90}_{i-1} \oplus \uparrow \oplus H_{i-1} \oplus \rightarrow \ \oplus H_{i-1} \oplus \downarrow \ \oplus {}_mH^{-90}_{i-1}, \quad \text{for } 1 = 1, 2, \ldots$$

where $\oplus$ means spliced, the superscript denotes the rotation angle, the arrows $\uparrow \rightarrow \downarrow$ mean draw a segment in the direction North, East, and South. In order to splice correctly, the rotated copies have also to be mirrored, which is indicated by the pre-index m.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Hibert curve, Feb 2012
%%Author: Kees van der Laan, kisa1@xs4all.nl
%%BoundingBox: -1 -1 321 151
%%BeginSetup
%%EndSetup
%%BeginProlog
/-s  s neg def
/lineN{0 0 moveto 0 s  lineto currentpoint stroke translate}def
/lineS{0 0 moveto 0 -s lineto currentpoint stroke translate}def
/lineE{0 0 moveto s 0  lineto currentpoint stroke translate}def
%
/Hilbert{%on stack order >=1 ==> Hilbert curve
         %s size of line segment (global),
1 sub dup 0 gt
{90 rotate  1 -1 scale Hilbert  1 -1 scale -90 rotate
  lineN                Hilbert
  lineE                Hilbert
  lineS
 -90 rotate 1 -1 scale Hilbert 1 -1 scale  90 rotate
 }if 1 add%reset order
} def
%%EndProlog
%
%Program ---the script---
%
/s 10 def %size of segment
                    1 Hilbert pop
   s    0 translate 2 Hilbert pop
2 s mul 0 translate 3 Hilbert pop
3 s mul 0 translate 4 Hilbert pop
showpage
%%EOF
```

```
program Hilbert(pf,output);
{plot Hilbert curves of orders 1 to n. Wirth(1975).}
const n= 4; h0=512;
var i,h,x,y,x0,y0: integer;
   pf: file of integer; {plotfile}
procedure A(i: integer);
begin if i>0 then
  begin D(i-1); x:=x-h; plot;
        A(i-1); y:=y-h; plot;
        A(i-1); x:=x+h; plot;
        B(i-1);
  end
end;
{B, C and D similar for rotated instances}
begin startplot;
   i:=0; h:= h0; x0:=hdiv2; y0:=x0;
   repeat {plot Hilbert curce of order i}
      i:=i+1; h:= h div 2;
      x:=x0 + h div 2;
      y:=y0 + h div 2;
      setplot;
      A(i)
   until i=n
   endplot
end.
```



The US facility of PS facilitates programming of these kinds of curves. Moreover, we don't need a plotfile in PS. After executing the PASCAL code we still have to create a .pdf file suitable for inclusion in documents.

Lauwerier(1990) provides a BASIC program, Peano1, for the Hilbert curves, restricted to orders $\leqslant 5$, via the Turtle Graphics method, i.e. drawing forward, meaning the broken line is obtained by drawing the segments in increasing order 1, 2, 3, ... The above program draws also forward by backtracking, a short of LIFO, Last-In-First-Out. In Peano1 the direction numbers are stored in an array of size 3411. Large enough for practical purposes. (The program above does not need the explicit direction numbers.)

The Peano curve, as shown at the beginning of this note, can be programmed along the same lines as for the Hilbert broken line above or via the Turtle Graphics casu quo Lindenmayer approach à la Lauwerier, which I leave as an exercise to the reader.

Application of plane-filling curves occurs in discretization of images where instead of a Cartesian grid the plane filling curve is used.

Hilbert curve in Metafont and Joseph Romanovski's (1995) PS code

```
%cgl, 1995
tracingstats:=1; proofing:=1;screenstrokes;autorounding:=0;
pickup pencircle scaled 0.2pt;
def openit = openwindow currentwindow
    from origin to (screen_rows,screen_cols) at (-40s,15s)enddef;
path p; s=10;
sz=0;p:=origin;%H_0 size and path
n=5; %Order of H-curve
for k=1 upto n:%H_1,...H_n consecutively
p:= p transformed (identity rotated 90
    reflectedabout (origin,up))--
p shifted ((-sz-1)*s,0)--
p shifted ((-sz-1)*s,(-sz-1)*s)--
p transformed (identity rotated -90
   reflectedabout (origin,up) shifted (-sz*s,(-2sz-1)*s));
sz:=2sz+1;
clearit;draw p; showit;
endfor
end
```

```
...
/S{0 R rlineto currentpoint stroke moveto}def
/T{90 rotate}def
/TM{T 1 -1 scale}def
/H{TM dup    0 gt
   {1 sub
          H S
       TM H S
          H T S
-1 1 scale H 180 rotate
    1 add
   }if    TM
}def
/R 8 def 100 100 moveto 6 H pop
showpage
```

Joseph's code is a bit cryptic (but is similar to mine). He uses short names, which does not make sense, makes the code difficult to read, which is considered a bad practice. But, ... he was on the right track, also with the use of colours via PS.

Conclusion. Such a simple(?) problem yielded already a few variants, of which most suffer from bad readability, because they are not biased by a production rule, IMHO.

Appendix: Sierpiński islands

Sierpiński islands S_1... S_3 have been shown in the introduction, in overlay.

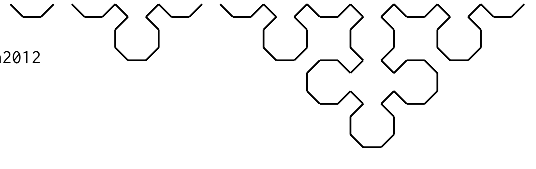
Wirth(1975), I consider a starting point for programming a Sierpiński island. Wirth did not have the rotation of US facility, nor did PS exist.

Algorithm: The curve is closed and is composed of 4 (90° rotated) broken lines connected by lines in the direction SE, SW, NW and NE. The program consists of 2 parts: first the generation of a side and second the appropriate splicing of rotated copies.

The PS program is based on the following production rule for a side

$$S_i = S_{i-1} \oplus SE \oplus S_{i-1}^{-90} \oplus E \oplus S_{i-1}^{90} \oplus NE \oplus S_{i-1}, \quad \text{for } i = 1, 2, \ldots$$

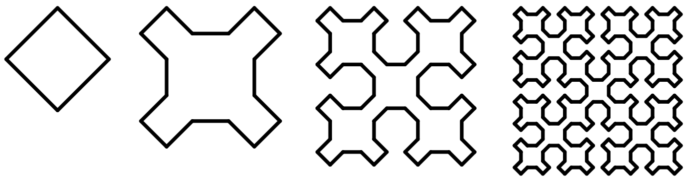
where $\oplus$ means spliced, the superscript denotes the rotation angle. and SE, E, NE mean draw a line in the direction South-East, East, and North-East.



```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski island Side, March2012
%%Author: Kees van der Laan
%% Affiliation: kisa1@xs4all.nl
%%BoundingBox:-1 -82 287 2
%%BeginSetup
%%EndSetup
%%BeginProlog
/s' s 1.4142 div def
/NE{0 0 moveto s' dup     lineto currentpoint stroke translate}def
/SE{0 0 moveto s' dup neg lineto currentpoint stroke translate}def
/E {0 0 moveto s 0        lineto currentpoint stroke translate}def
/SideS{%on stack order >=1  ==> Sierpinski side
      %s size of line segment (global)
1 sub dup 0 ge
{ dup           SideS
  SE
  dup -90 rotate SideS  90 rotate
  E
```

```
  dup  90 rotate SideS -90 rotate
  NE
  dup            SideS
}if 1 add} def
0 setlinejoin 1.415 setmiterlimit 1 setlinecap
%%EndProlog
%
%Program ---the script---
%
/s 10 def % size of segment
              1 SideS pop
s 0 translate 2 SideS pop
s 0 translate 3 SideS pop
showpage
```

Wirth programmed the four rotated instances in (recursive) procedures A, B, C and D, which entailed a more complex recursion scheme.



Sierpiński islands 0 ... 3

For those who don't own the 1975 book I have included Wirth's program, translated into Metafont in 1995.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Sierpinski island, March 2012
%%Author: Kees van der Laan
%%Affiliation: kisa1@xs4all.nl
%%BoundingBox:-10 -185 307 2
%%BeginSetup
%%EndSetup
%%BeginProlog
(C:\\PSlib\\PSlib.eps) run %loads /SideS and auxiliaries from PSlib
%%EndProlog
%
%Program ---the script---
%
/s 20 def % size of segment
4{                          SE -90 rotate}repeat%S_0
1.5  s mul 0 translate /s s 2 div def
4{             1 SideS pop SE -90 rotate}repeat%S_1
4.5  s mul 0 translate /s s 2 div def
4{             2 SideS pop SE -90 rotate}repeat%S_2
10.5 s mul 0 translate /s s 2 div def
4{             3 SideS pop SE -90 rotate}repeat%S_3
showpage
%%EOF
```

The $S_0 ... S_3$ pictures obtained by the given program are shown above.

```
%Translation of Wirth's code into Metafont
tracingstats:=1;proofing:=1;screenstrokes;
pickup pencircle scaled 0.01pt;
s=10; path p;
def openit = openwindow currentwindow from origin
    to (screen_rows,screen_cols) at (-5s,5s)enddef;
def A expr k=if k>0:
   A(k-1);draw z--hide(x:=x+h;y:=y-h)z;
   B(k-1);draw z--hide(x:=x+2h)z;
   D(k-1);draw z--hide(x:=x+h;y:=y+h)z;
   A(k-1); fi enddef;
def B expr k=if k>0:
   B(k-1);draw z--hide(x:=x-h;y:=y-h)z;
   C(k-1);draw z--hide(y:=y-2h)z;
   A(k-1);draw z--hide(x:=x+h;y:=y-h)z;
   B(k-1); fi enddef;
def C expr k=if k>0:
   C(k-1);draw z--hide(x:=x-h;y:=y+h)z;
   D(k-1);draw z--hide(x:=x-2h)z;
   B(k-1);draw z--hide(x:=x-h;y:=y-h)z;
   C(k-1); fi enddef;
def D expr k=if k>0:
   D(k-1);draw z--hide(x:=x+h;y:=y+h)z;
   A(k-1);draw z--hide(y:=y+2h)z;
   C(k-1);draw z--hide(x:=x-h;y:=y+h)z;
   D(k-1); fi enddef;
def sierpinski expr k=%k is order of curve
  if k>0: A(k);draw z--hide(x:=x+h;y:=y-h)z;
          B(k);draw z--hide(x:=x-h;y:=y-h)z;
          C(k);draw z--hide(x:=x-h;y:=y+h)z;
          D(k);draw z--hide(x:=x+h;y:=y+h)z;
  fi enddef;
z=origin; h=16;
sierpinski2; showit; end
```