Fractals

I.Kolingerová

1. Self-similarity and fractal dimension
2. Fractal types and their computation
3. Geometric fractals
4. Stochastic fractals

Literatura

- Francis S.Hill Jr.: Computer Graphics, Macmillan Publishing Company, New York, 1990
- J.Vince: 3-D Computer Animation, Addison-Wesley Publishing Company, 1992
- S.C. Hoggar: Mathematics for Computer Graphics, Cambridge University Press, 1992
- H.A. Lauwerier, J.A. Kaandrop: Fractals (Mathematics, Programming and Applications), TR CS-R8762, Centre for Mathematics and Computer Science, Amsterdam, The Nietherlands, 1980
- N. Wirth: Algoritmy a štruktúry údajov (Algorithms + Data Structures = Programs), Alfa, Bratislava 1988

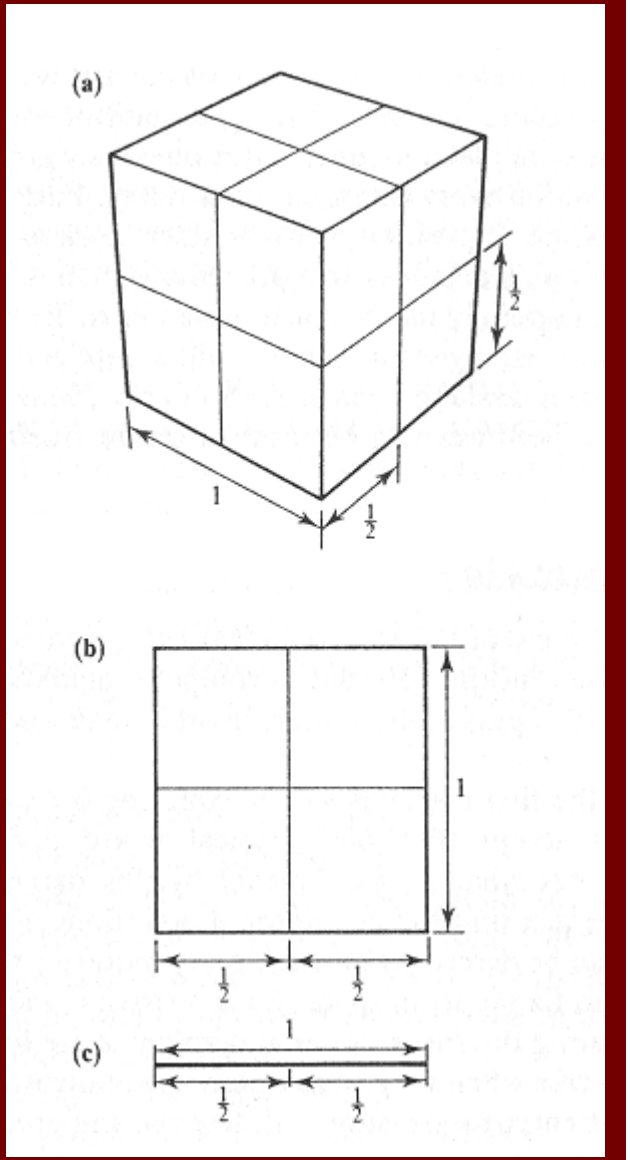
1. Self-similarity and fractal dimension

- A self-similar shape: general level of its detail the same regardless the distance from which we look at the shape
- For example, “rougness" of Koch curve Kn for

  n->∞ always the same (similarly Hilbert curve

  or triomino)
- ∞ in computer graphics is only approximated but the effect is OK
- Landscape examples of self-similarity: costline, rocks, clouds, leaves, blood system ...
- Research: B. Mandelbrot – self-similar curves – fractals (<= fractal dimension)

- Common – topological dimension:
  - a point 0D
  - a line 1D
  - a plane 2D ...
- Curve of ∞ length embedded in a finite part of a plane – - something between 1 and 2
- => fractal dimension



D-dimension
N-number of self-similar parts
S-scaling factor

a) D=3,N=8,S=1/2
b) D=2,N=4,S=1/2
c) D=1,N=2,S=1/2

- Relation among D,N,S:

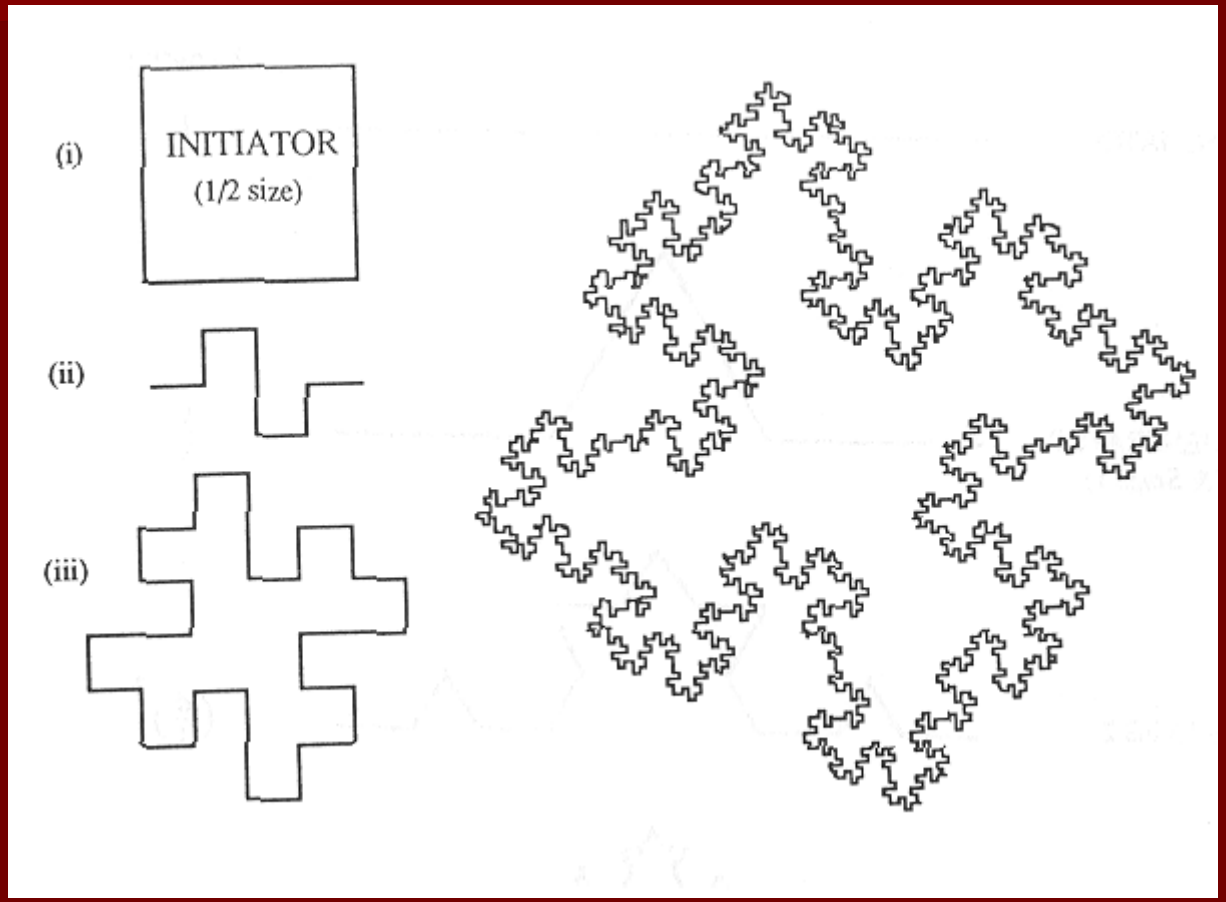
  $N=1/S^D \Rightarrow D=\log(N)/\log(1/S)$
- Koch curve: S=1/3,N=4 => D=log(4)/log(3)=1.2619..
- This dimension has not the original meaning => fractal dimension

2. Fractal types and their computation

Fractal types

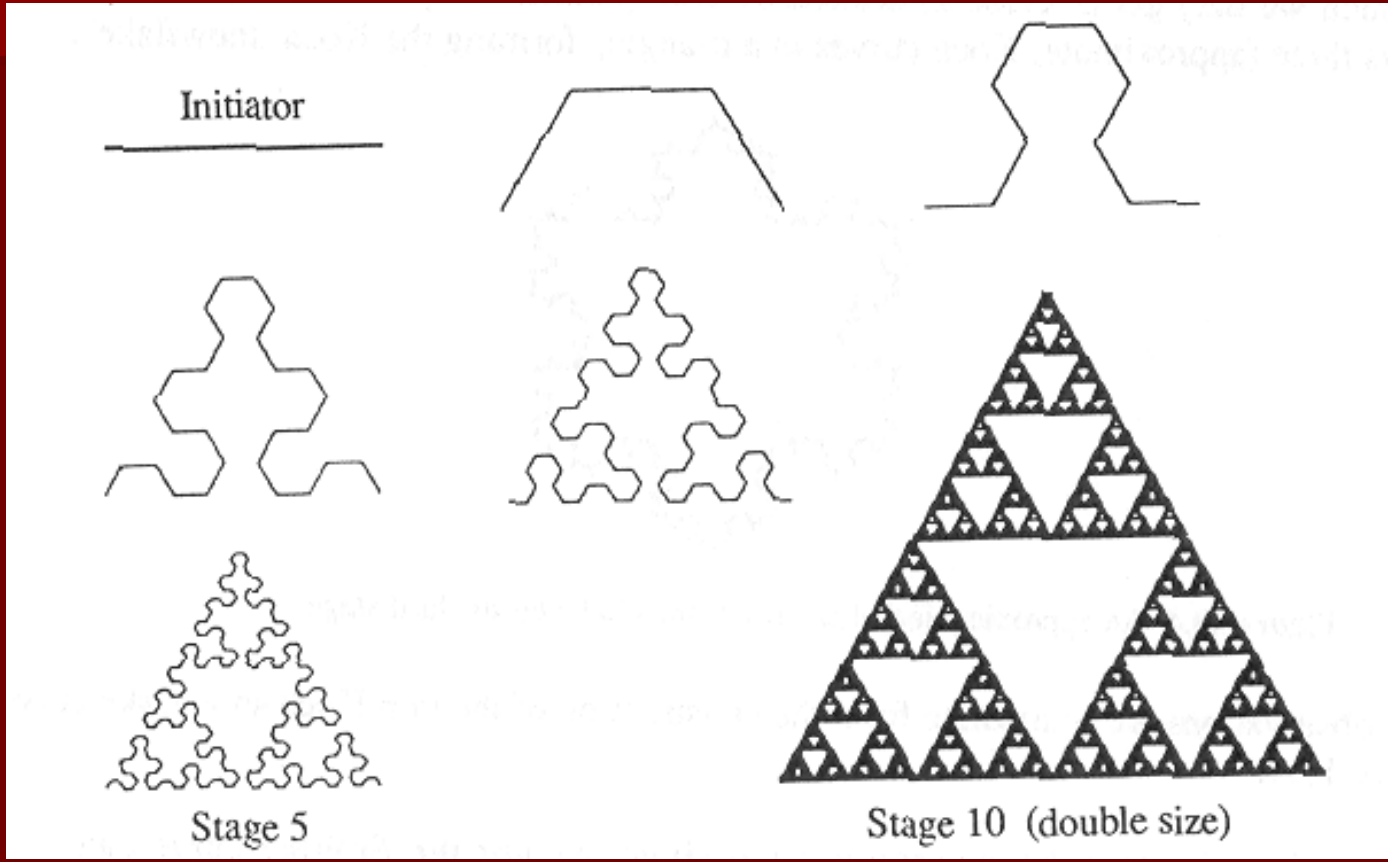
- geometric – generated by geometric patterns – an initiator, a generator; line segments of the initiator replaced by the generator
- stochastic – generated by a random process – costline, terrains
- algebraic – generated by iterations of algebraic transformation functions, e.g. $z <= z^2 + c$

Example of geometric fractal: Koch island



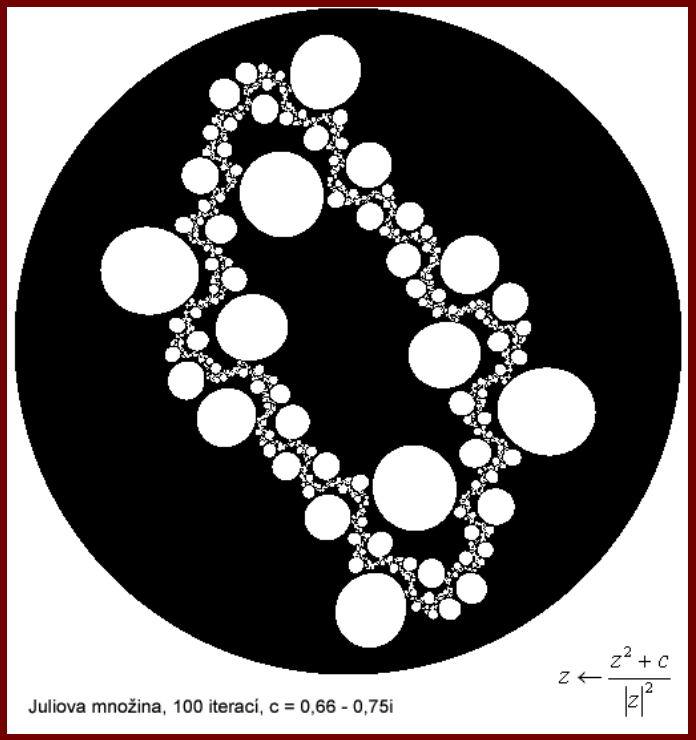
Koch island
(8th generation),
(i) initiator
(ii) generator
(iii) 1st generation
- (i) to (iii) in half size

Example of geometric fractal: Sierpinski curve

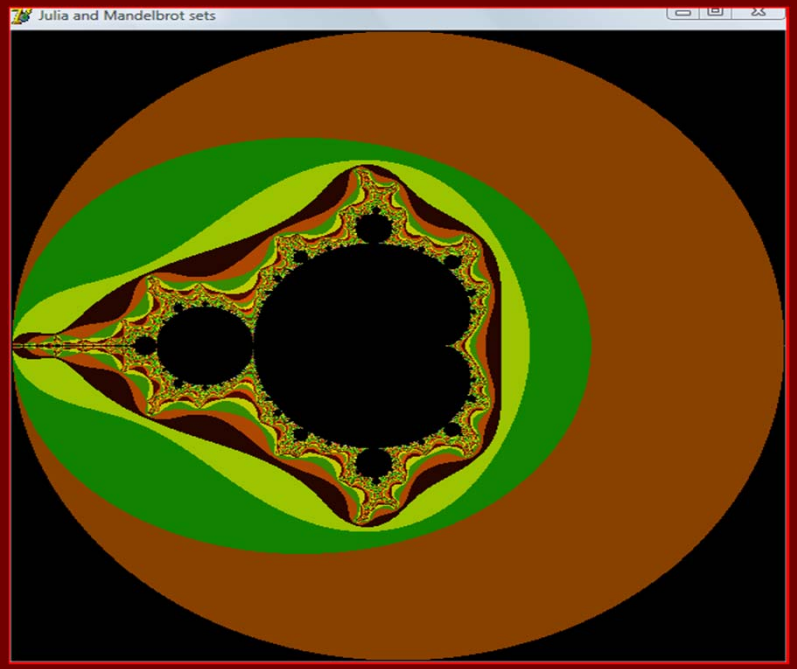


Other geom. fractals: see recursively defined curves

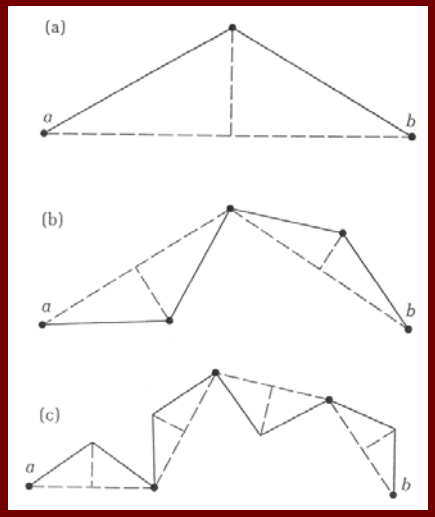
Example of algebraic fractal: Mandelbrot set, Julia set



Juliova množina, 100 iterací, c = 0,66 - 0,75i



Example of stochastic fractal: a recursive computation



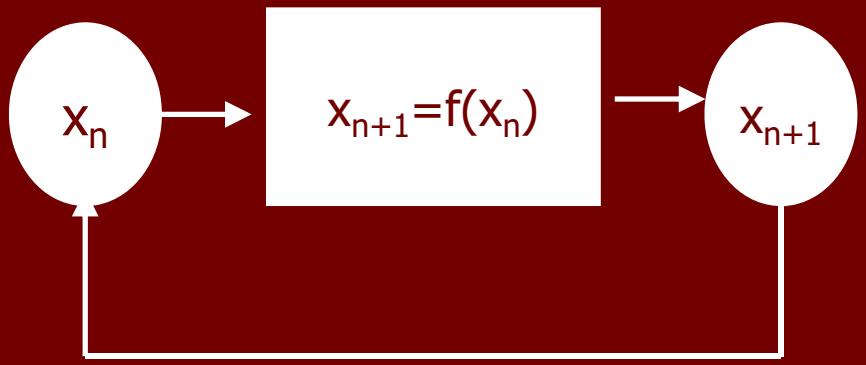
Fractals computation:

- Recursive (geometric construction)
- Iterated function system (IFS)
- Formal languages (L-systems)
- Non-linear complex maps
- Chaos game (similar to IFS)

- Big differences between methods, but some fractal objects can be obtained in more than one way (e.g., Sierpinski triangle)

Fractal computation:

- Principle of feedback – the output of one iteration is the input for the next iteration



- Relations linear or non-linear
- The output not necessarily has fractal dimension – sometimes a "normal" object, points in ∞ or a point of attraction

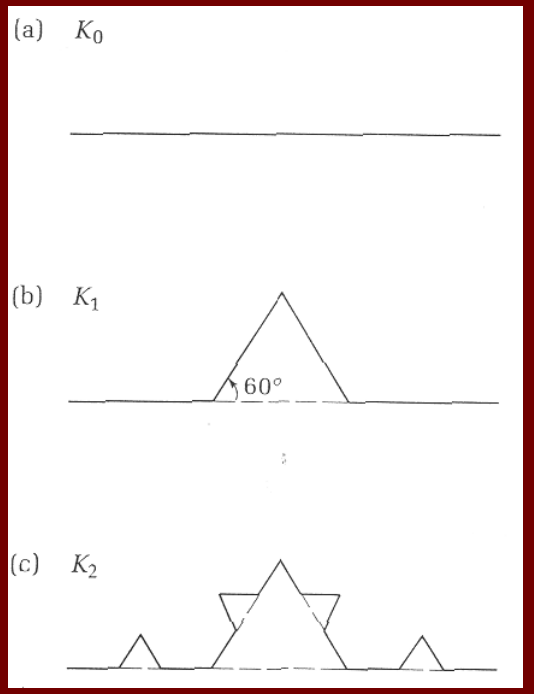
3. Geometric fractals

- recursively generated curves

  a) Koch curve
  b) C-curves and dragon curves
  c) Curves filling the space
  d) Reptiles

a) Koch curve

- 1904, Helge von Koch
- Infinitely long line on a finite plane
- Progressive generation



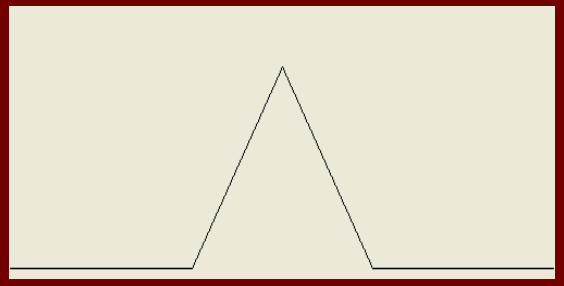
- K2 is 4/3-times longer than K1
- Ki+1 by replacement of each of 4 segmens by a “peak"
- The total length $(4/3)^i$, for i ->∞ the length ->∞ , still the curve stays embedded in a finite area

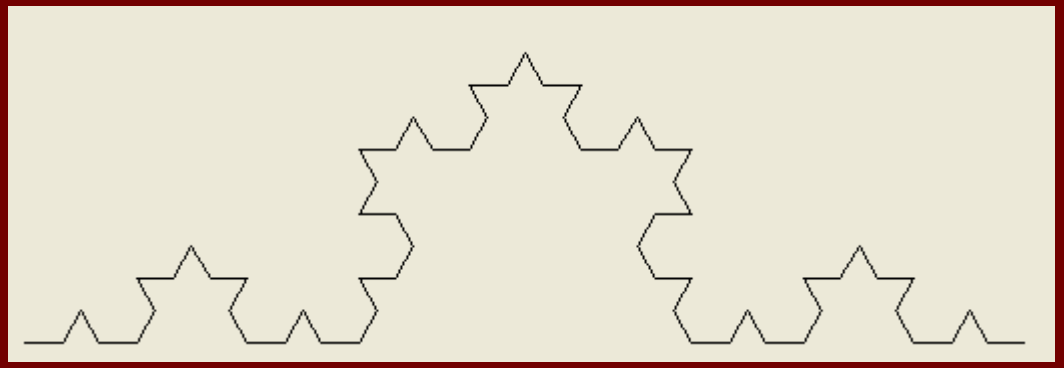
Algorithm of Koch curve computation

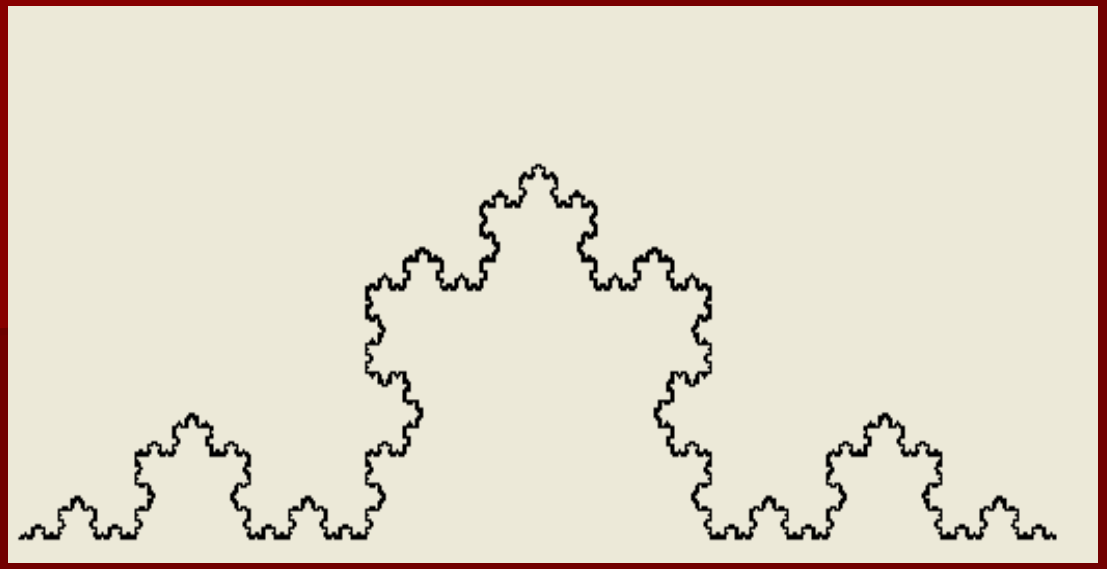
- Recursive definition, the input is: curve order n, starting segment length l, starting point, direction dir
- In this way a replacement of any line segment,

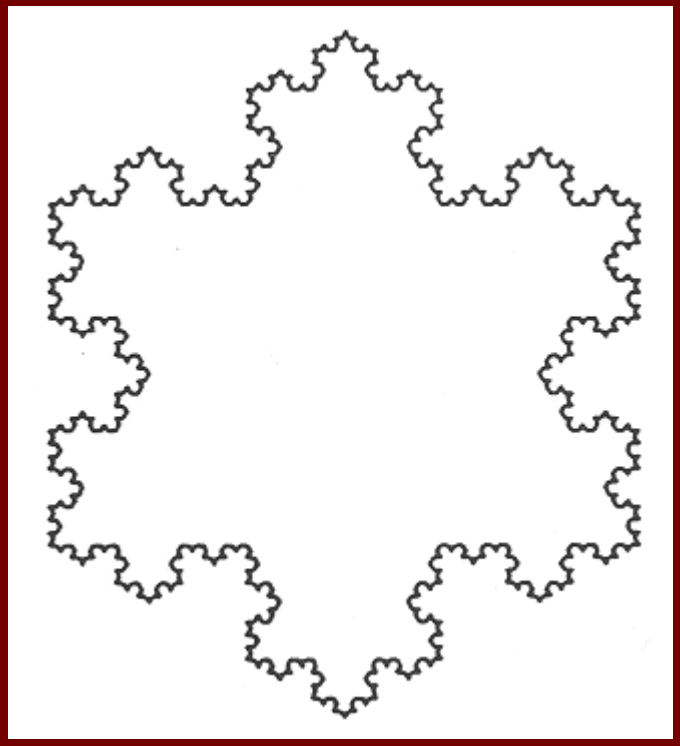
  ->Koch landscape,

  equilateral triangle -> Koch snowflake







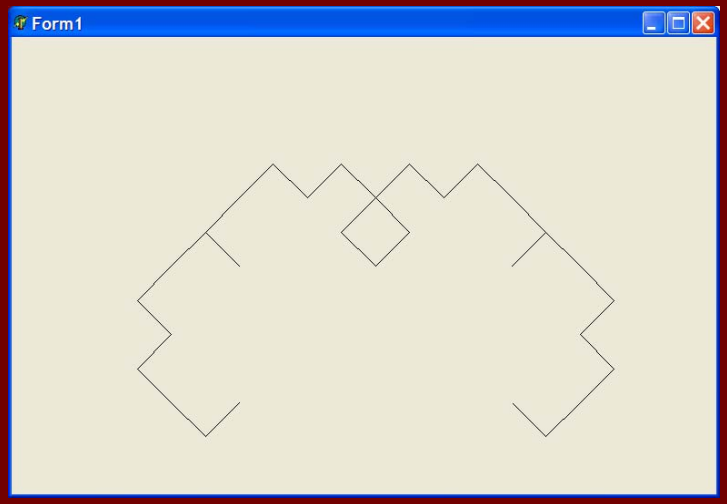


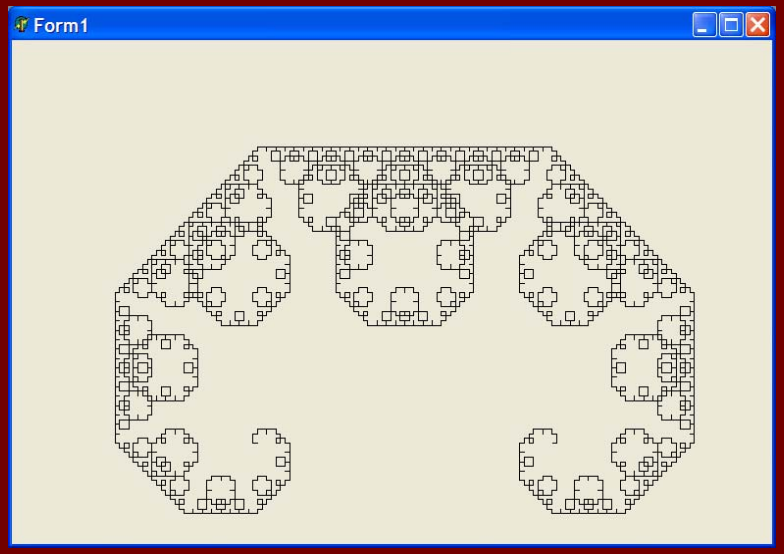
```
procedure Koch (dir,len:real;n:integer);
{ the line sg. length len in the direction dir, order n, start in (xp,yp) }
const rads=0.017453293;
begin
  if n > 0 then
    begin
      Koch (dir,len/3,n-1); dir := dir+60;
      Koch (dir,len/3,n-1); dir := dir-120;
      Koch (dir,len/3,n-1); dir := dir+60;
      Koch (dir,len/3,n-1);
    end
 else
    begin
     LineTo (xp+len*cos(rads*dir),yp+len*sin(rads*dir));
     xp := xp + len*cos(rads*dir); yp := yp + len*sin(rads*dir);
    end;
end;
```

b) C-curves, dragon curves

- Replacement of the parent “C”
- Length is 1/√2-times the length of the parent

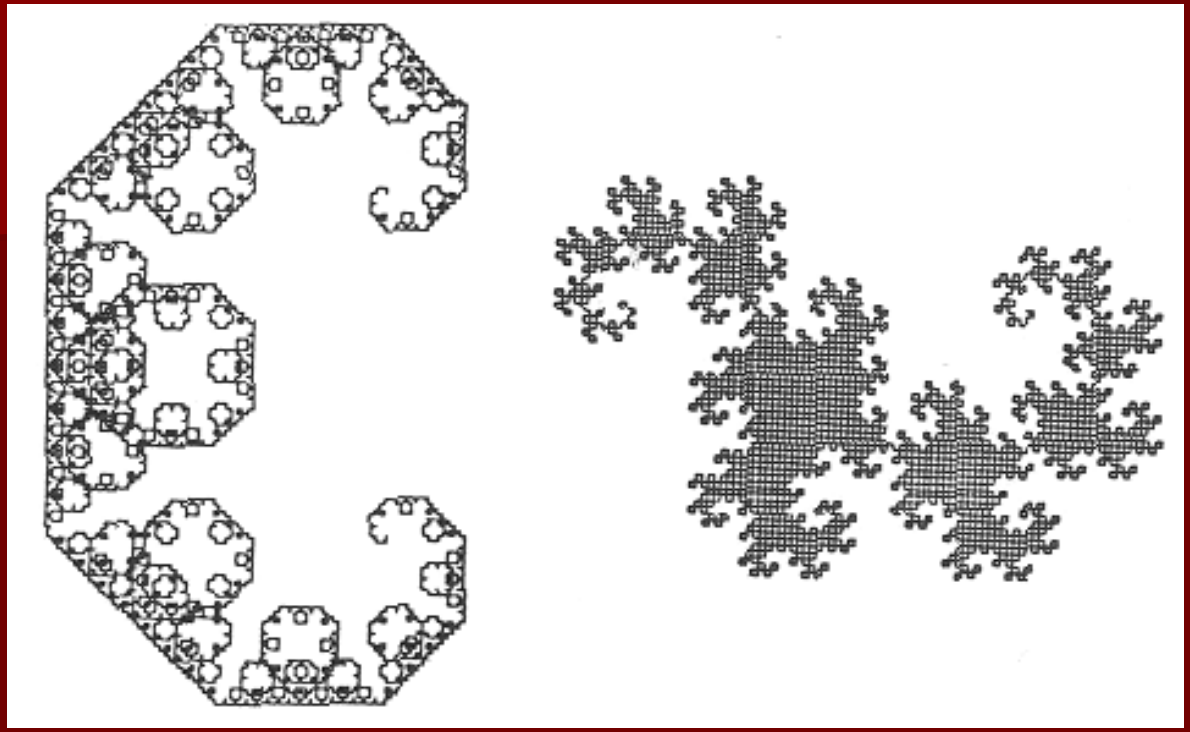
n=5





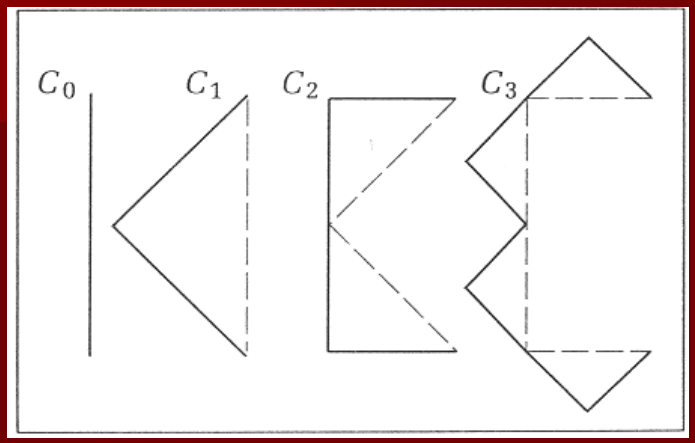
n=12

- Dragon curve – similar but the direction of connection alternates (1 segment replaced by left turn, 1 by right turn)

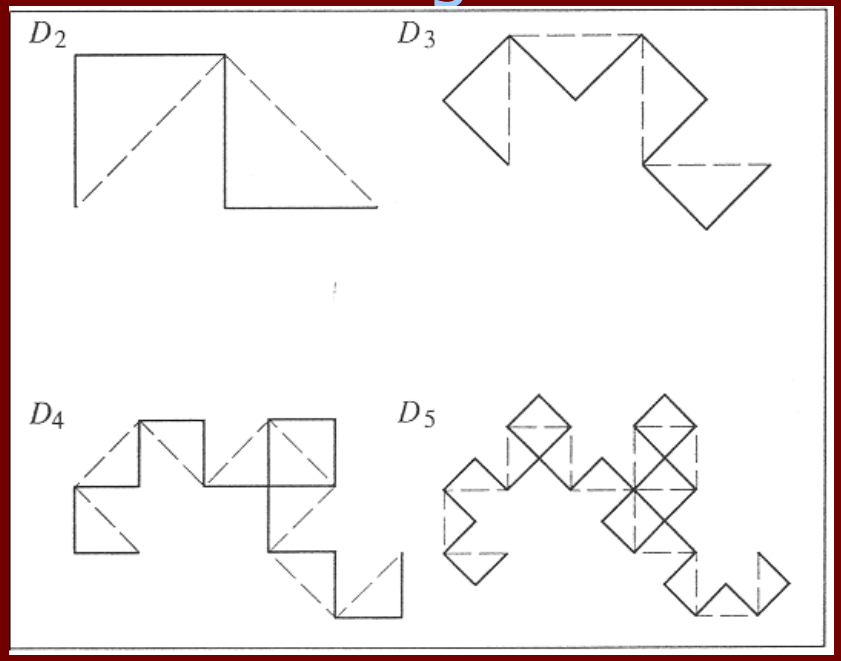


C-curve and dragon curve of order 12

Several generations of C-curves:



Several generations of dragons:



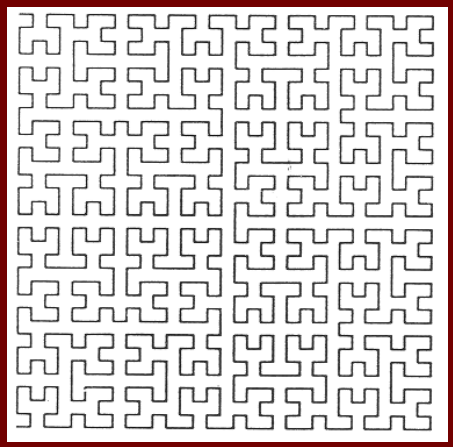
```
procedure DrawC (dir,len:real;n:integer);
{the line sg. length len in the direction dir, order n, start in (xp,yp) }

const fct=0.7071067; {1/sqrt(2) } rads=0.017453293;
begin
  if n > 0 then
    begin
      dir  := dir + 45; DrawC (dir,len*fct,n-1);
      dir  := dir - 90;  DrawC (dir,len*fct,n-1);
    end
 else
    begin
     LineTo (xp+len*cos(rads*dir),yp+len*sin(rads*dir));
     xp := xp + len*cos(rads*dir); yp := yp + len*sin(rads*dir);
    end;
end;
```

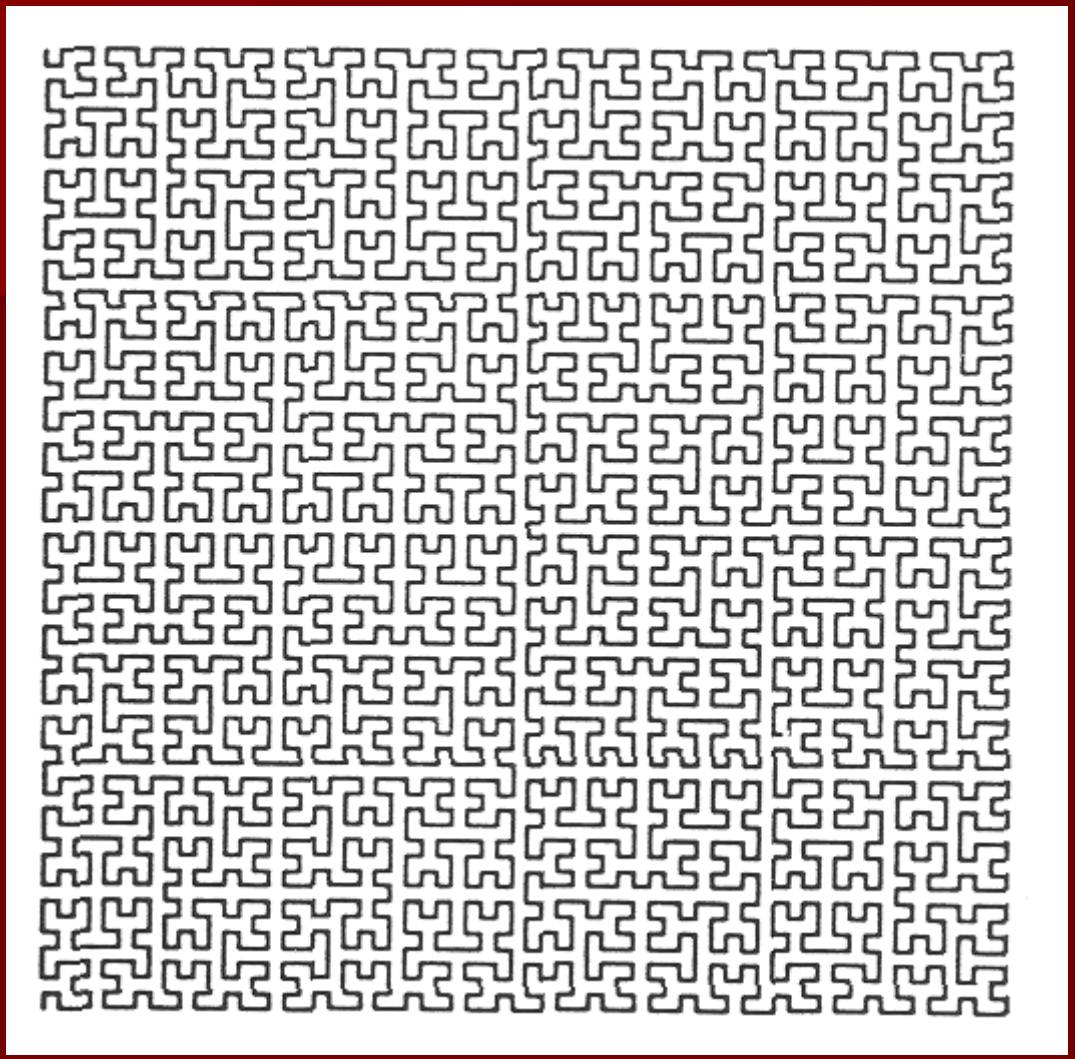
c) Curves filling space

Hilbert curves

- Named after the mathematician D.Hilbert (1862-1943)
- Each H. curve has its order and one of 4 orientations A, B, C, D
- They are joined by auxiliary connections to the curves of the second order
- A2 ~ D left A down A right B
- B2 ~ C up B right B down A
- The same for higher orders,
  - E.g., B9 ~ C8 up B8 right B8 down A8



n=5



Hilbert curve of order 6

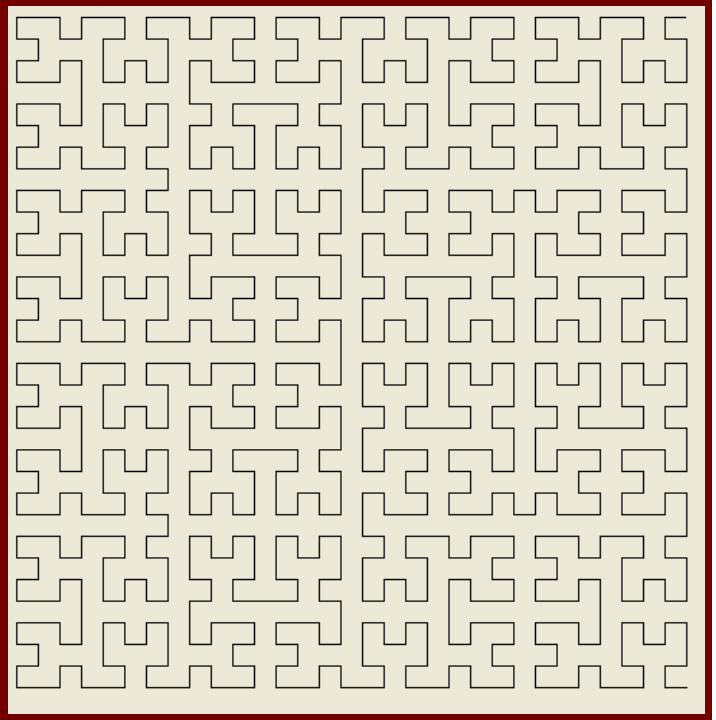
- Definition table of Hilbert curve:

  A: DAAB A: left down right

  B: CBBA B: up right down

  C: BCCD C: right up left

  D: ADDC D: down left up

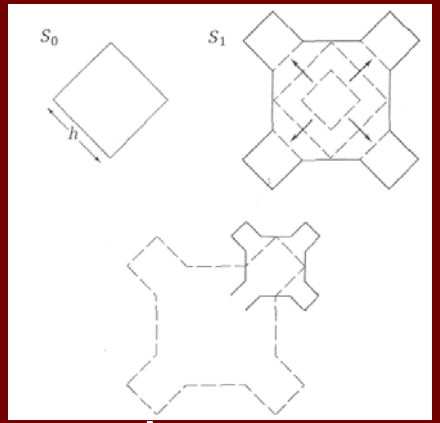


- Ranks among the curves filing space - the higher the order, the more the space filled

```
procedure TForm1.A (i : integer);
begin
 if i > 0 then
    begin
     D(i-1); x := x - h; MyLineTo (x,y);
     A(i-1); y := y - h; MyLineTo (x,y);
     A(i-1); x := x + h; MyLineTo (x,y);
     B(i-1)
    end
end;
procedure TForm1.B (i : integer);
begin
 if i > 0 then
    begin
     C(i-1); y := y + h; MyLineTo (x,y);
     B(i-1); x := x + h; MyLineTo (x,y);
     B(i-1); y := y - h; MyLineTo (x,y);
     A(i-1)
    end
end;
```

```
procedure TForm1.C (i : integer);
begin
 if i > 0 then
    begin
     B(i-1); x := x + h; MyLineTo (x,y);
     C(i-1); y := y + h; MyLineTo (x,y);
     C(i-1); x := x - h; MyLineTo (x,y);
     D(i-1)
    end
end;
procedure TForm1.D (i : integer);
begin
 if i > 0 then
    begin
     A(i-1); y := y - h; MyLineTo (x,y);
     D(i-1); x := x - h; MyLineTo (x,y);
     D(i-1); y := y + h; MyLineTo (x,y);
     C(i-1)
    end
end;
```

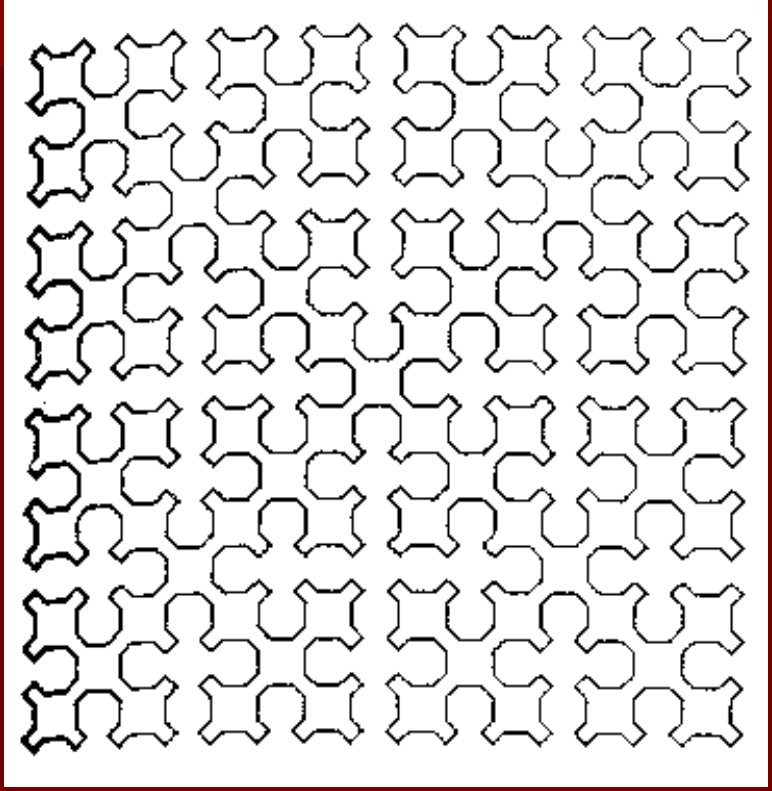
Sierpinski curves

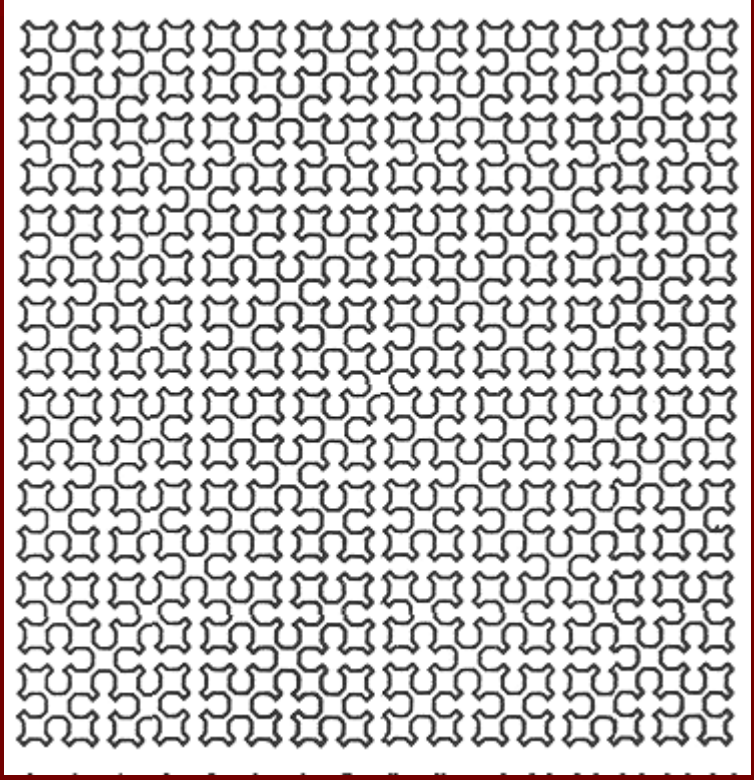


- Other space-filling curves
- Def. as sequence of operation Reply and Connect

- S0 = square sized h sitting on its corner, S1 is obtained by replacement of corners by 4 half-sized copies, shifted from the centre by h, connected by horizontal and vertical lines and by erasure of 4 inner sides

- Next generation: replacement of squares by 4 half-sized copies, shifted from the centre by a proper distance, drawing of diagonals, erasure of inner sides





Sierpinski curve S4 (left) and S5 (right)

- This approach easy, but erasure necessary
- Better:

S: A ↘ B ↙ C ↖ D ↗

- Open patterns are connected by lines which do not belong to the recursive pattern – we add them to S0

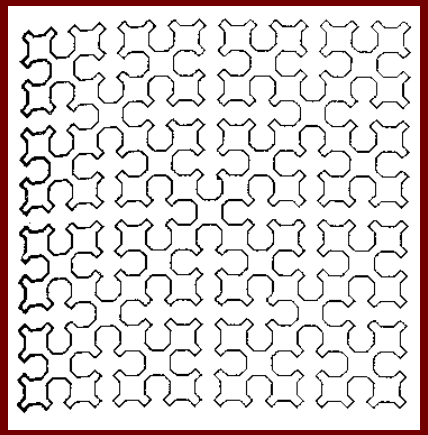
- Recursive formulae:

  A: A↘B➡D↗A

  B: B↙C⬇A↘B

  C: C↖D⬅B↙C

  D: D↗A⬆C↖D

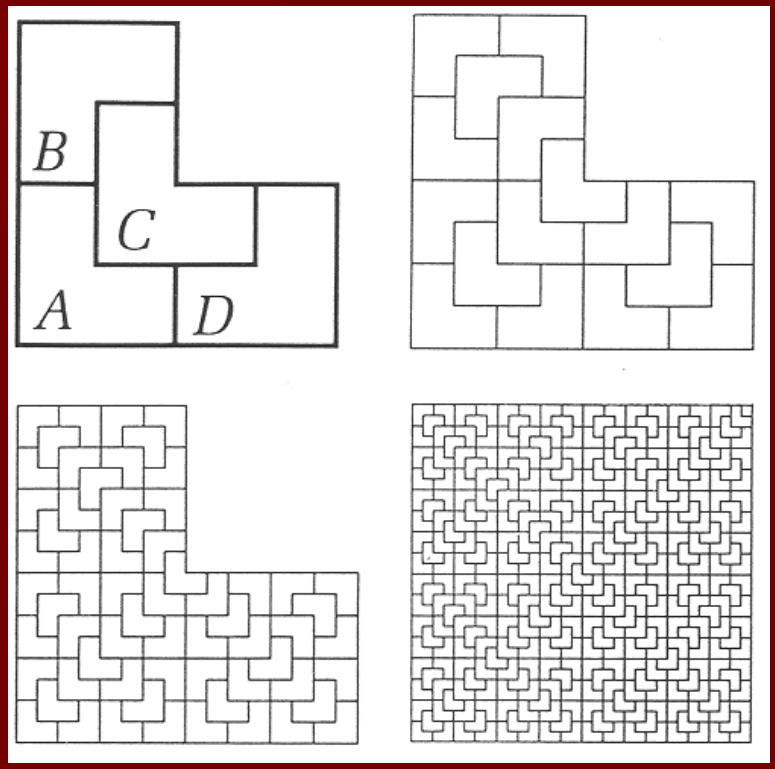


Program: try yourselves

d) Reptiles

- Non-periodic tilings, their easiest description is recursive
- Various replications of reptiles form together a big reptile of the same shape
- The reptile defined recursively

Ex.: triomino



- 4 orientations
  of the same shape
- Each triomino can be replaced
  by this quartet, it can be again replaced…
- Number of refinements ~ curve order
- Various orientations => non-periodical

4. Stochastic fractals

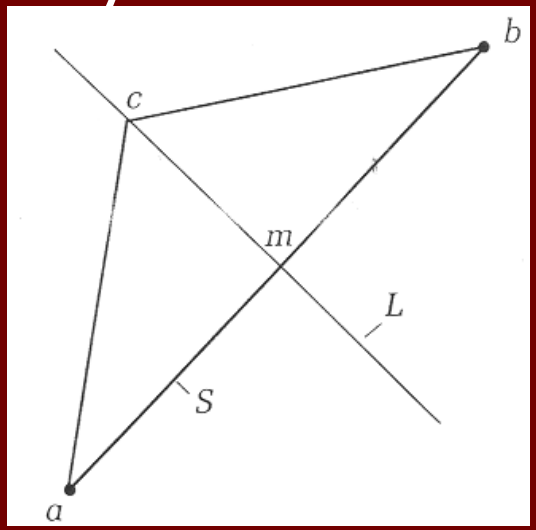
- The simplest fractal – a fractalization of line segment – replace a line segment by a random peak in each step
- Ex.: S replaced by 2 line s.,

  c is a randomly chosen point

  on L, the rectangular bisector

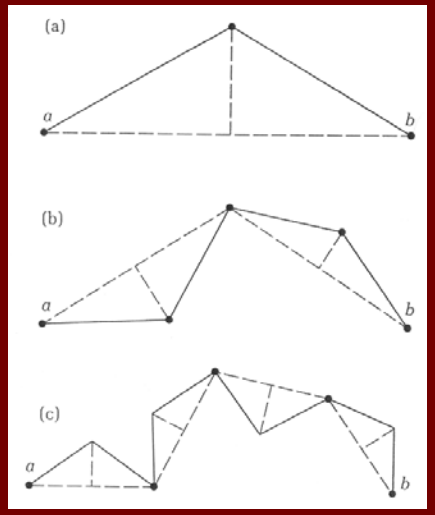
  to S,

  m=(mx,my) – line sg. middle point



- The distance between c, m ~ t, length S; t is positive or negative

- t usually modeled as Gaussian random value
- Recursive generation until some criterion is fulfilled



Control of fractal curve persistence

- Control of scale – how “raggid" the curve is
- $f = 2^{(1/2-H)}$, StDev := StDev*f, t:=Gauss*StDev
- H between 0 and 1, above 1/2 – smoother, more persistent curves, below 1/2 – more raggid
- Ex.:

H	f
0.0	1.4
0.3	1.1
0.5	1.0
0.7	0.9
1.0	0.7

f=1 for H=1/2 - standard deviation the same on all levels – a model of Brownian motion

f<1 for H>1/2 – standard deviation decreases with level growth

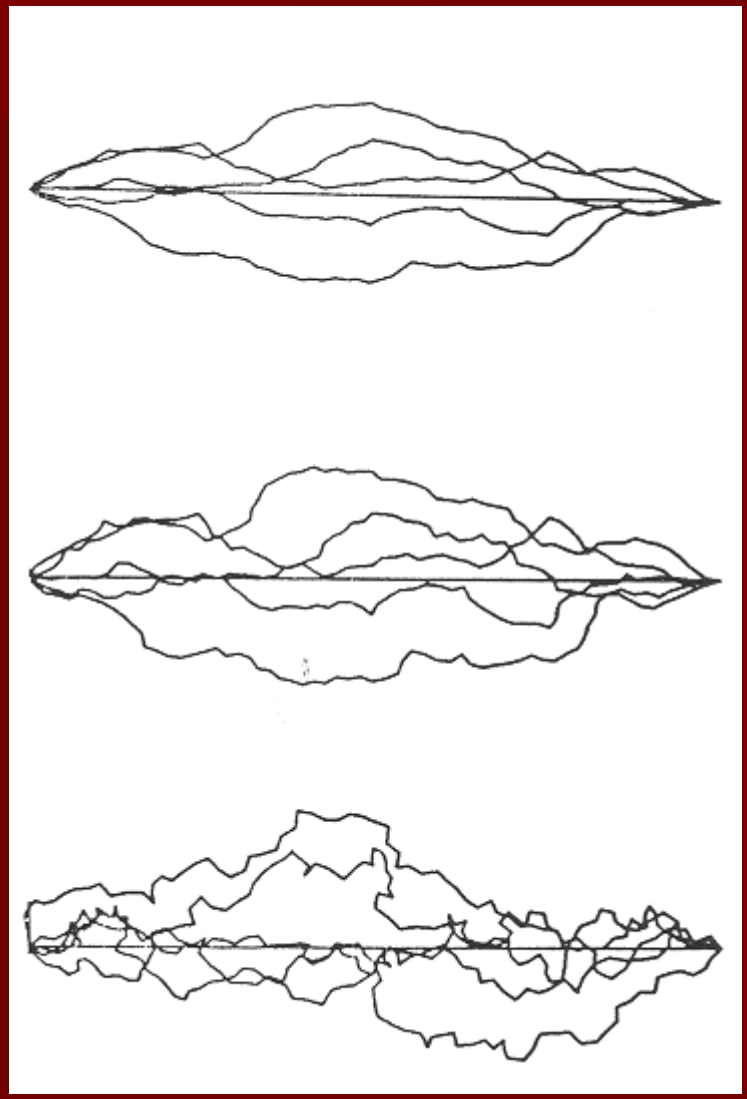
f > 1 for H < 1/2 – standard deviation increases with level growth – "antipersistent curve"

Examples:

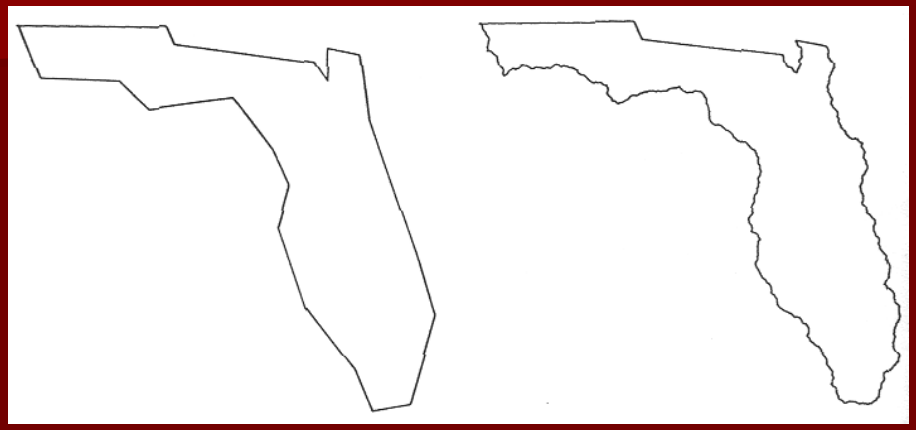
H=0.7

H=0.5

H=0.3



- Any curve can be fractalized

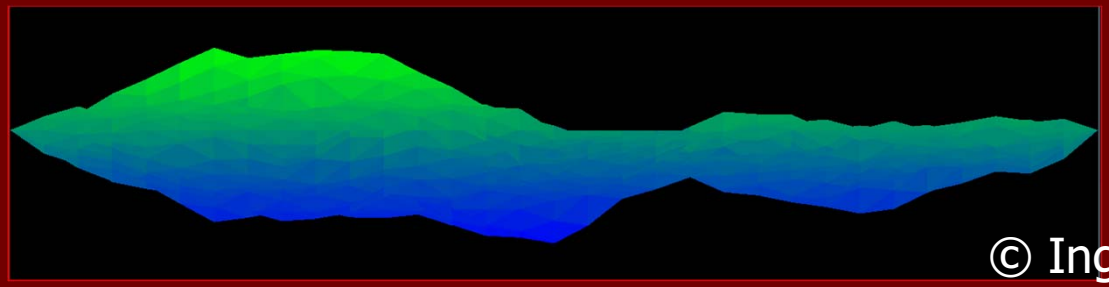


realistic costline model

- Sequences are reproducible if RandSeed is the same
- It is enough to store
  - vertices of original line segments
  - minimal length of a line or maximal level of recursive subdivision and standard deviation
  - randseed

Fractal surfaces

- For realistic look of mountain terrains
- Start – a planar triangle, middle points of edges are shifted vertically by a random distance
- All is repeated for 4 triangle faces until the faces are “small enough"
- Holes may appear between triangles (creases)



Fractal surfaces

Technique of repeated triangle subdivision with random point shifts

Fractal subdivision in middle points,
part of heights the same –
- simulation of sea level