IFS and Chaos Game

I.Kolingerová

1. IFS
2. Modification: Chaos Game
3. Possible Modifications

References

- M.F.Barnsley: Fractals Everywhere, Springer-Verlag, New York, 1988
- H.-O.Peitgen, D.Saupe [Eds]: The Science of Fractal Images, Springer-Verlag, New York, 1988
- H.-O.Peitgen, H. Jurgens, D. Saupe: Fractals for the Classroom, Springer-Verlag, New York, 1988
- R.L. Bowman: Fractal Metamorphosis: A Brief Student Tutorial, Computers & Graphics, Vol.19, No.1, pp.157-164, 1995
- H.J.Jeffrey: Chaos Game Visualization of Sequences, Computers&Graphics, Vol.16, No.1, pp.25-33, 1992

1. Iterated Function System (IFS)

- M.F.Barnsley, Fractals Everywhere, Springer-Verlag, New York, 1988
- We need the term of affine transformation:

$$w\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) : \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0, a, b, c, d, e, f \in R$$

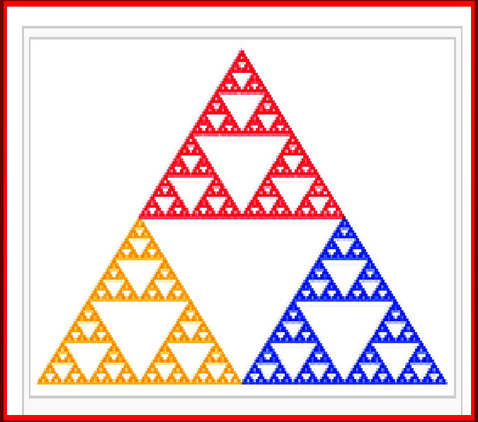
=> An inversion transformation exists

- IFS=[{w1,w2,...,wn},{p1,p2,...,pn}], Σpi=1
- wi – a set of affine transformations ("contraction mapping")
- pi – their probabilities

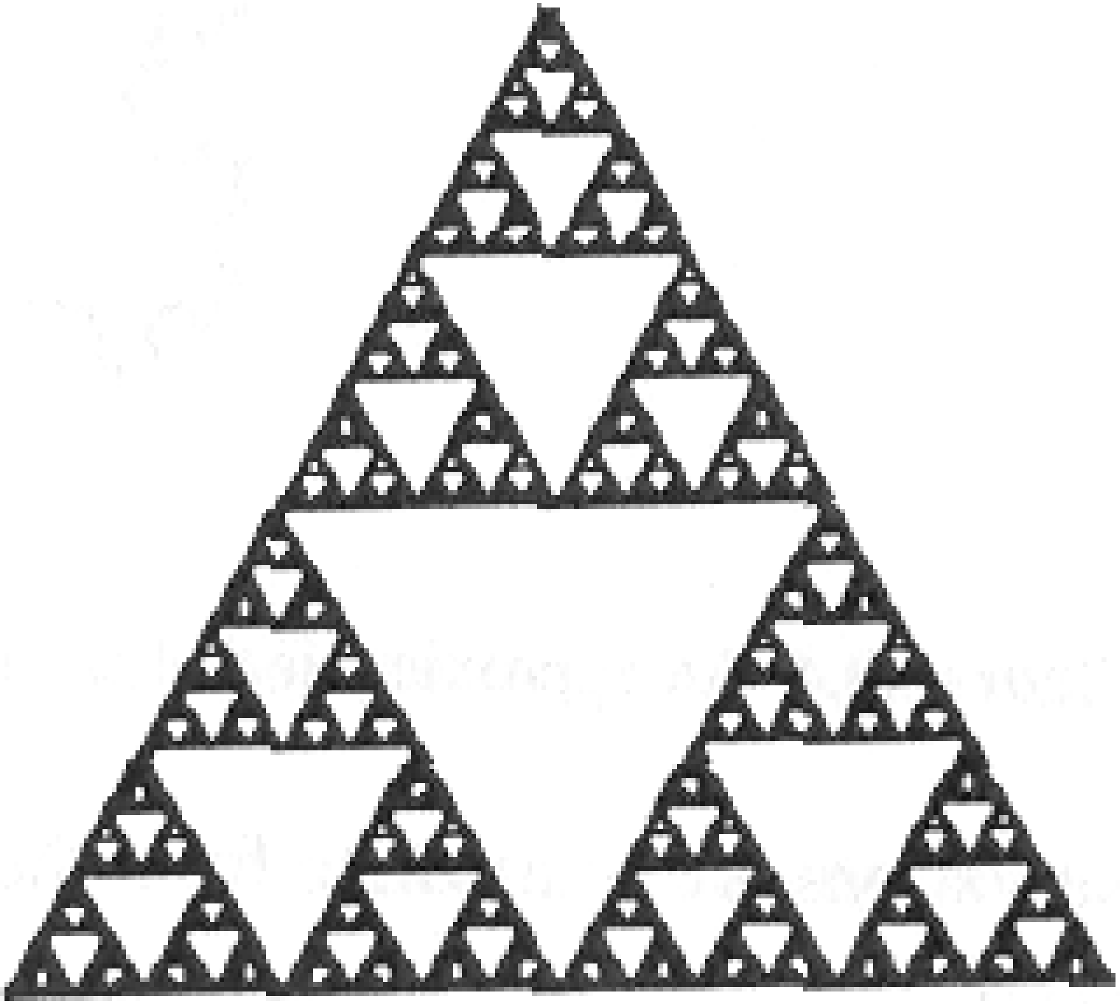
- The transformations have to be average contractive, i.e., they have to contract a point-to-point distances "in average"
- All so transformed points are gradually "drawn" into the area of one set – the so-called IFS attractor
- Coefficients a,b,c,d – rotation, shear, scale, e,f - translation

- One iteration – a new point from an old one; on the beginning, several points are not drawn, then the points converge to the attractor
- Ex. The Sierpinski triangle – 3 functions

w	a	b	c	d	e	f	p
1	0.5	0	0	0.5	0	0	1/3
2	0.5	0	0	0.5	0.5	0	1/3
3	0.5	0	0	0.5	0.5	0.5	1/3



- Higher dimensions – equations also for further coordinates
- Colors or non-linear transformations can be included



2 algorithms to compute fractals from IFS

a) Deterministic

- Fill a 2D array T by ones in the first and last rows and columns, otherwise by zeroes
- Then apply wi functions on T, store in a different array S

```
for i :=1 to 100 do for j :=1 to 100 do
      if T[i,j]=1 then
        begin
          S[a[1]*i+b[1]*j+e[1],c[1]*i+d[1]*j+f[1]]=1;
          S[a[2] ... ], S[a[3]...] etc. // apply all functions
        end
```

- Then flip T, S, reset the output array, draw cells with T[i,j]=1

- It is possible to start with other (unempty) array of values, the same result
- Check indices not to over/underflow the array boundaries

b) Random iteration

```
x:=0;y:=0; niter:= 1000;
for i:=1 to niter do
    begin
      k := Random(3)+1;
      // choose one number from {1,2,...,n}
      // with equal probability
      newx :=a[k]*x+b[k]*y+e[k];
      newy :=c[k]*x+d[k]*y+f[k];
      x := newx; y := newy;
      if i>10 then plot (x,y)
    end
```

- The starting point would be nice to lie in the attractor but we do not know the attractor in advance => any starting point, e.g., the origin, is OK
- The condition of contraction ensures that after several iterations all the points lie in the attractor

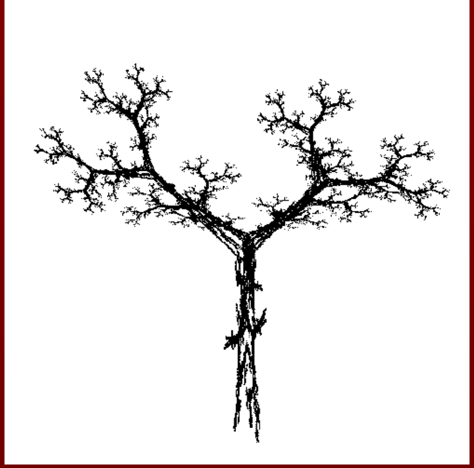
- Ex.: A square

w	a	b	c	d	e	f	p
1	0.5	0	0	0.5	1	1	1/4
2	0.5	0	0	0.5	50	1	1/4
3	0.5	0	0	0.5	1	50	1/4
4	0.5	0	0	0.5	50	50	1/4

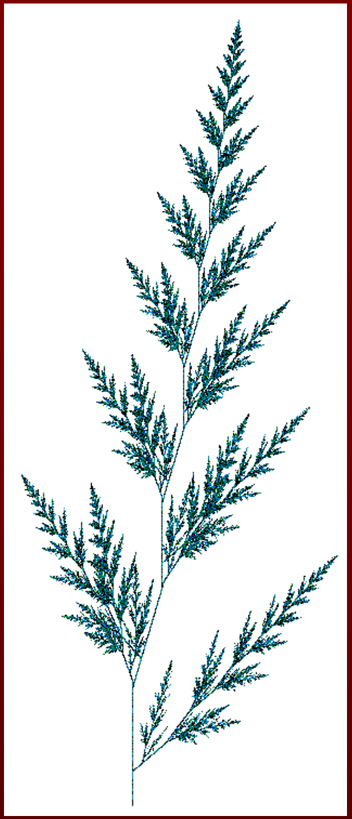
- Ex.: The fern

w	a	b	c	d	e	f	p
1	0	0	0	0.16	0	0	0.01
2	0.85	0.04	-0.04	0.85	0	1.6	0.85
3	0.2	-0.26	0.23	0.22	0	1.6	0.07
4	-0.15	0.28	0.26	0.24	0	0.44	0.07

- Ex.: A fractal tree



???



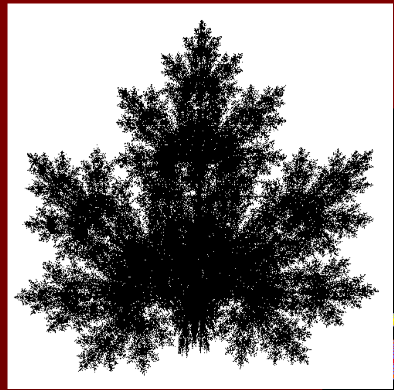
w	a	b	c	d	e	f	p
1	0	0	0	0.5	0	0	0.05
2	0.42	-0.42	0.42	0.42	0	0.2	0.4
3	0.42	0.42	-0.42	0.42	0	0.2	0.4
4	0.1	0	0	0.1	0	0.2	0.15

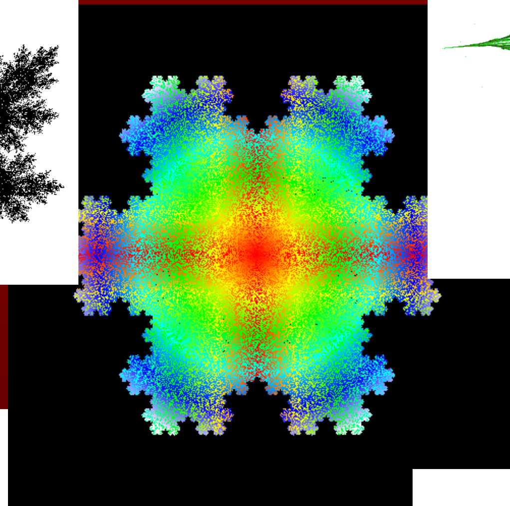
- Ex.: The Cantor discontinuum

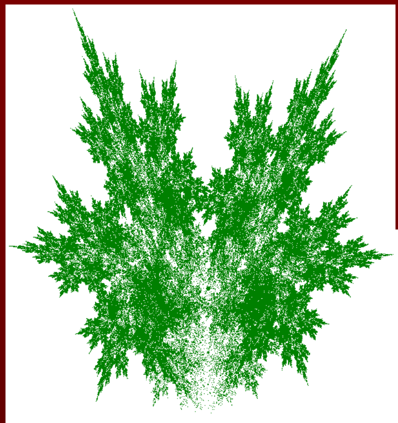
w	a	b	c	d	e	f	p
1	0.33	0	0	0	0	0	0.5
2	0.33	0	0	0	0.67	0	0.5

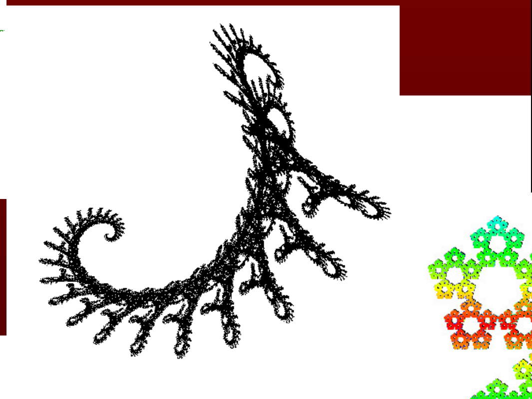
Where did the affinity disappear ???? ;-)

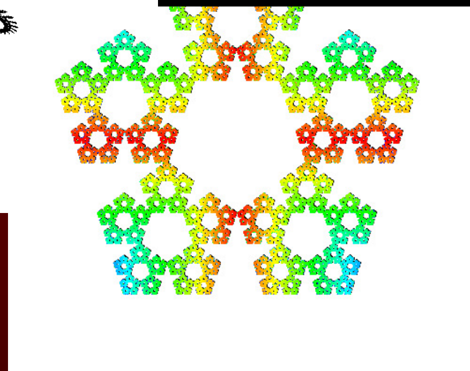
- Further examples

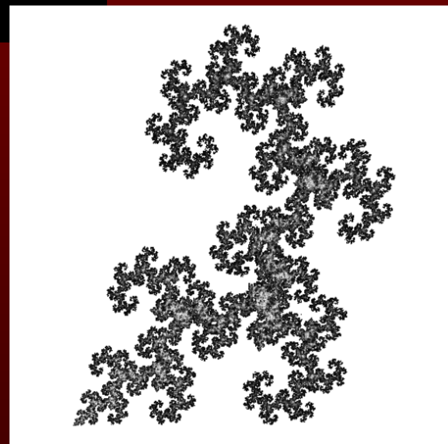








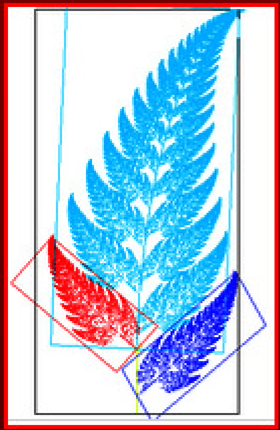




http://www.geocities.com/capecanaveral/lab/1837/index_a.html

IFS – a big role in the fractal compression

- IFS can serve as an image (fractal) representation, it is enough to know the transformation matrix (6 real numbers) and the probability vector
- An image is represented by n functions =>
  7n real numbers – very efficient compression
- Independent of resolution
- Decompression – the image can be done
  of any size
- Fundamental problem: to find transformations



- How to find the transformations?
  - Subdivide the image into the same or different areas, an adaptive subdivision using a quadtree, or subdivision into triangles
  - Goal: maximal self-similarity
  - Next step: apply transformations and compare similarity
  - Time-demanding, equality improbable, usually only some similarity => always a lossy compression
  - Found transformations = compressed image representation

2. Modification: Chaos Game

- The simplest is to try in hand

1. Draw 3 triangle vertices and number them (1,2   3,4   5,6)
2. Pick the starting point anywhere
3. Toss a dice
4. Place a mark in the middle of the path between the last point and the vertex whose number was provided by the dice
5. Repeat since 3

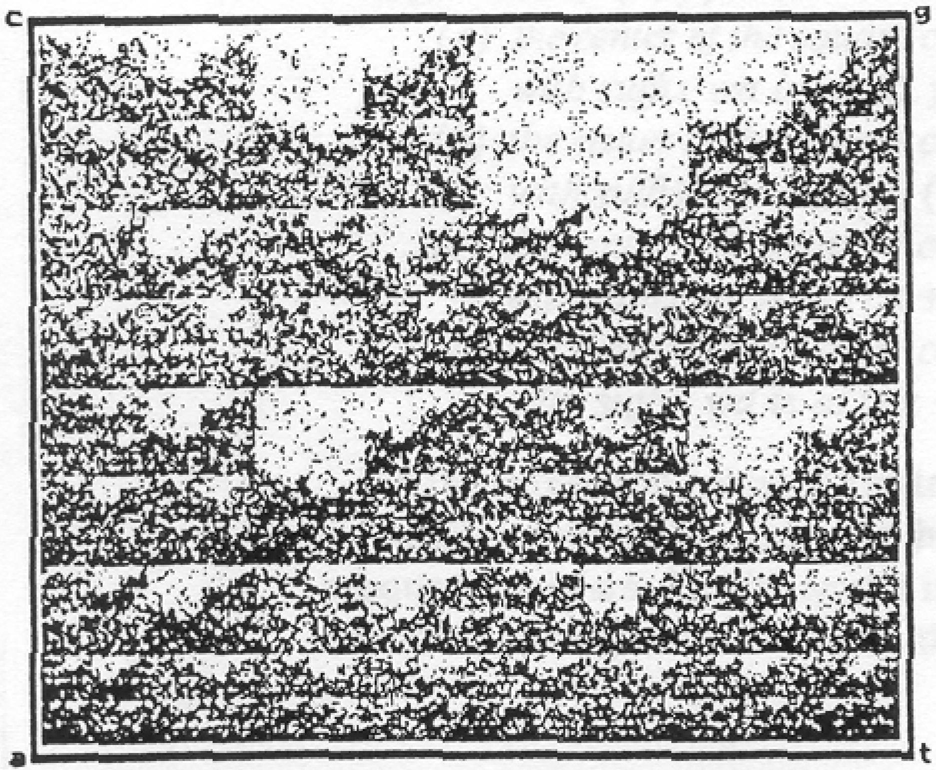
- The attractor - the Sierpinski triangle

- 5, 6, 7 vertices – an n-gon with patterns
- 8 and more – a filled n-gon without the centre
- 4 – a regularly filled square
- Chaos Game is in fact an IFS

- If the probability is irregular, the same attractor but a different shading (the same holds for a general IFS)
- If, e.g., the square is irregularly filled although the probabilities are the same => a bad random number generator

- Use: e.g., the square can represent a 1D sequence in a 2d form, keeping the structure of the sequence, if any exists
- A structure => non-randomness

- Ex.: DNA sequence – formally a string of characters a,c,g,t (or u) => a square with adequately marked corners



CGR of human beta globin region on chromosome 11 (HUMHBB) (73,357 bases).

- If the alphabet >= 4, rather n equal non-overlapping squares than n-gon (n-gon is not regularly filled)
- Non-uniformity in the sequence leads to a non-uniformity in the image
- Ex.:DNA, 24 classes of equivalence of aminoacid triads
- Ex.: a similarity of writing characteristics of different writings by one author

3. Possible modifications

a) R.A.Bowman, 1995

Slightly different IFS equations:

X = SC*(XP-XF(I))+XF(I)

Y = SC*(YP-YF(I))+YF(I)

where I – a random index of one of given vertices

(XP,YP) – last drawn point

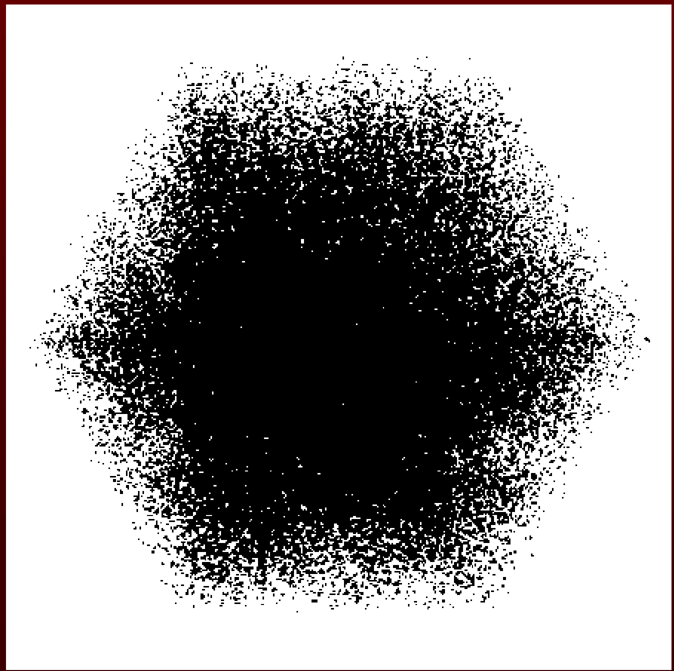
SC=1/B, where B is a strength of attraction

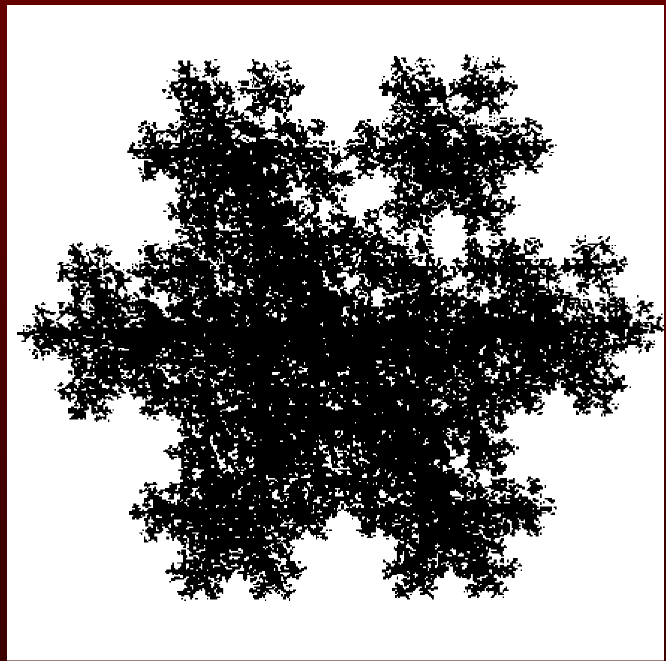
in each vertex, B>1

It iterates 100 000 times

- Ex.: A snowflake modification – 6 outer vertices, 5 inner, a missing vertex spoils the symmetry
- B is changed from 2 to 6

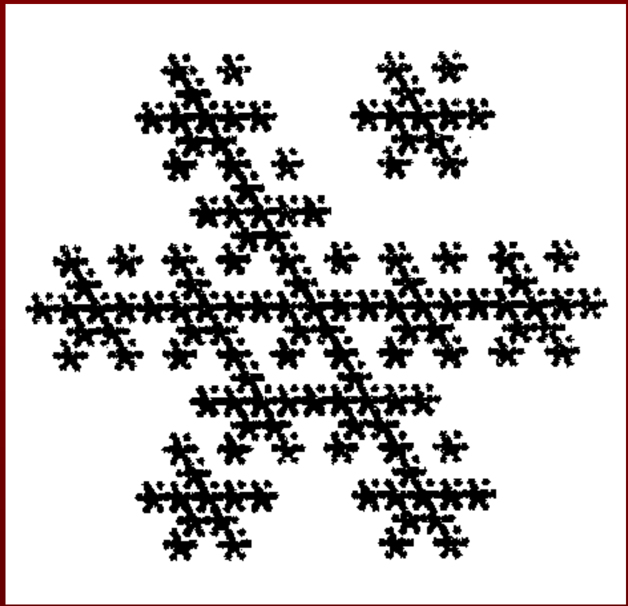
B=2

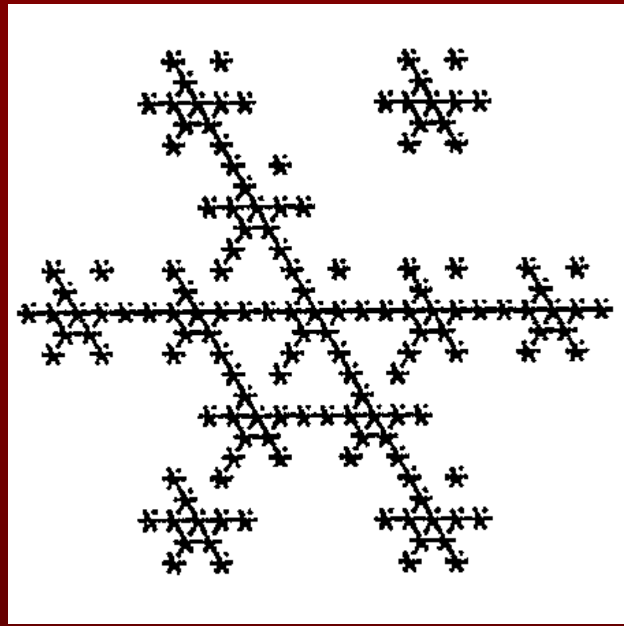




B=3

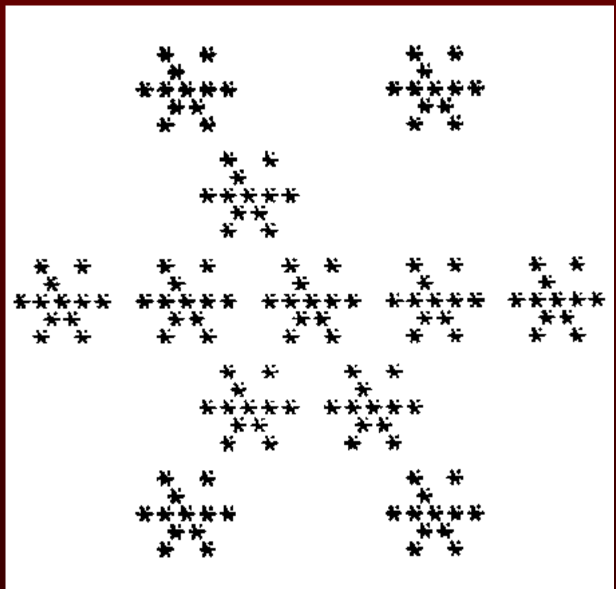
B=4





B=5

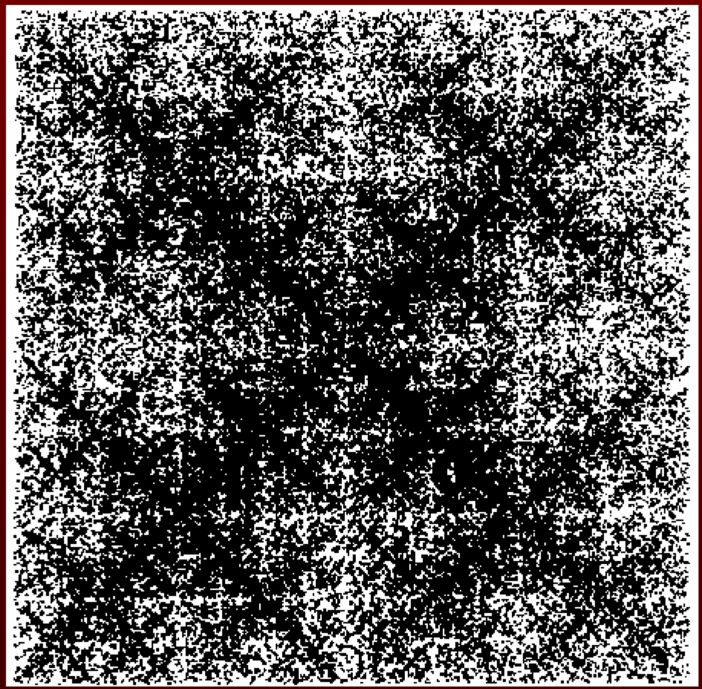
B=6

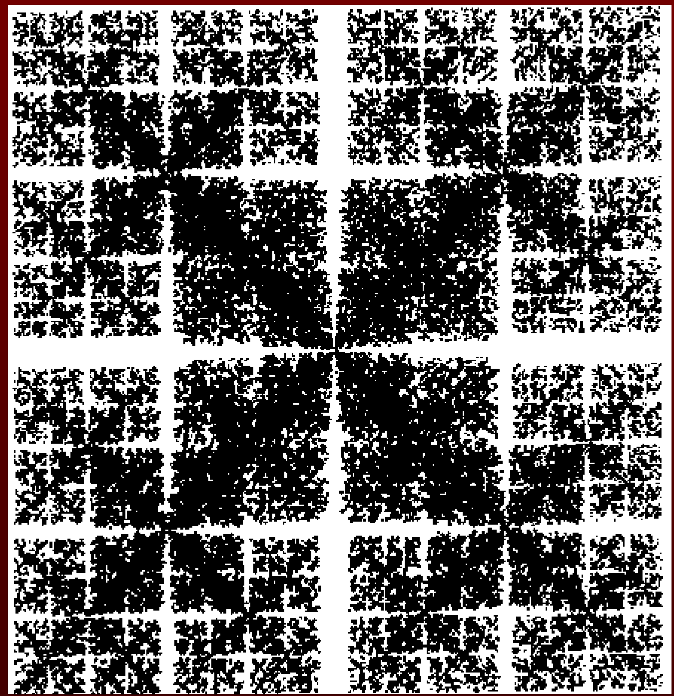


Point	Coordinates	(x,y)
1	0.0	0.0
2	1.0	0.0
3	0.5	0.866025447845459
4	−0.5	0.866025447845459
5	−1.0	0.0
6	−0.5	−0.866025447845459
7	0.5	−0.866025447845459
8	0.5	0.0
9	−0.25	0.4330127239227295
10	−0.5	0.0
11	−0.25	−0.4330127239227295
12	0.25	−0.4330127239227295

- Ex.: 5 vertices without changes while B is changed

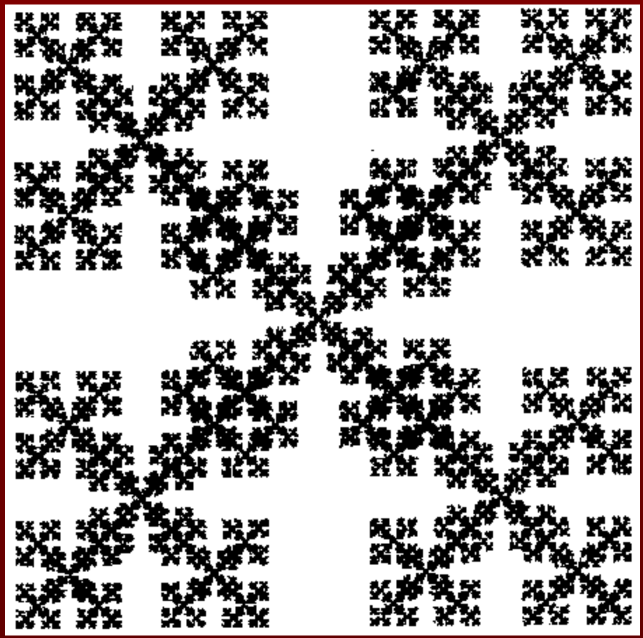
B=2

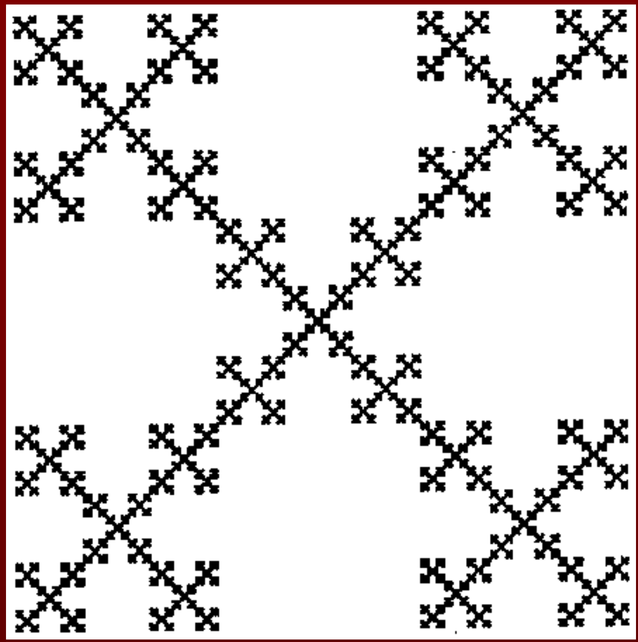




B=2.1

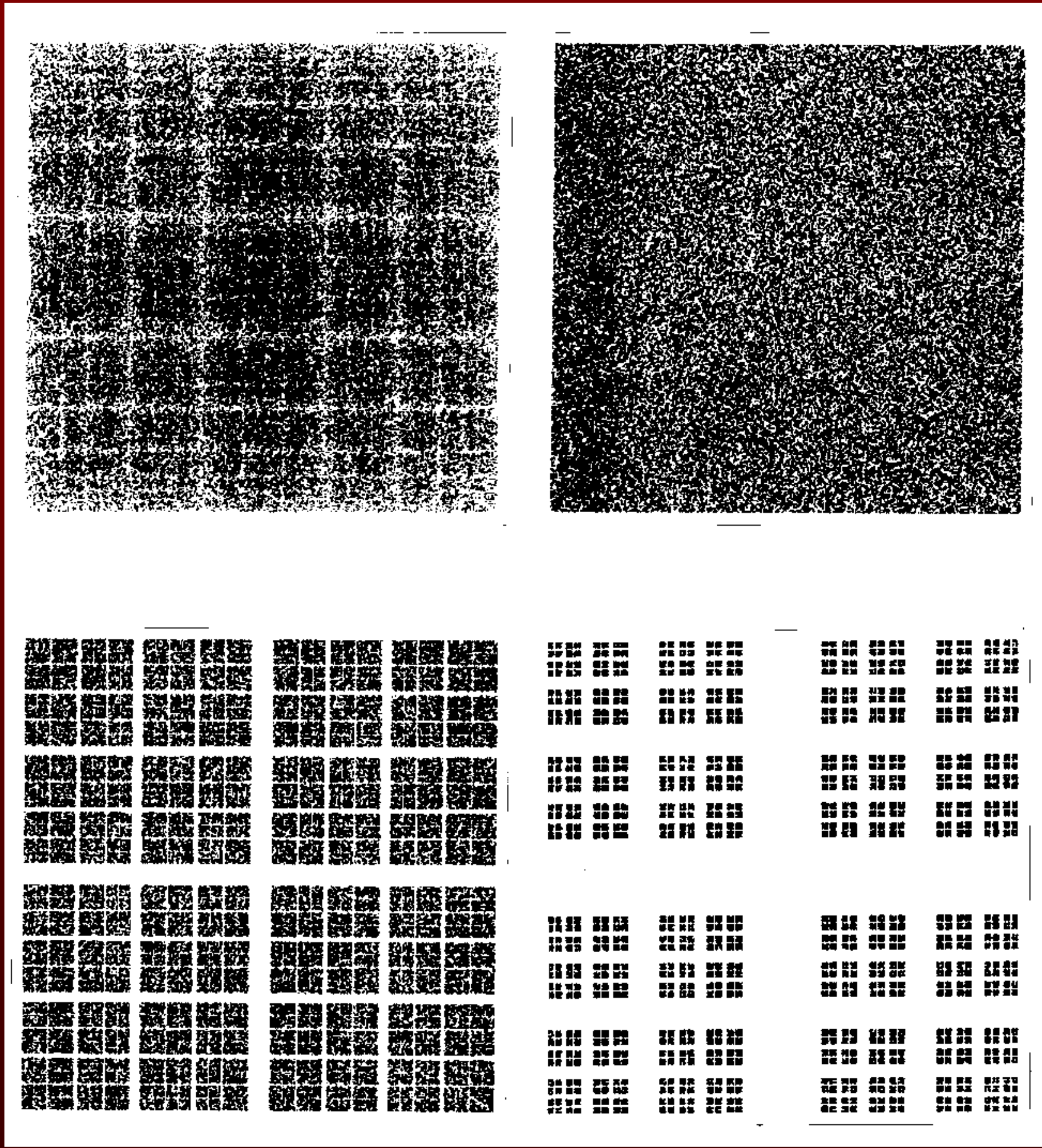
B=2.4



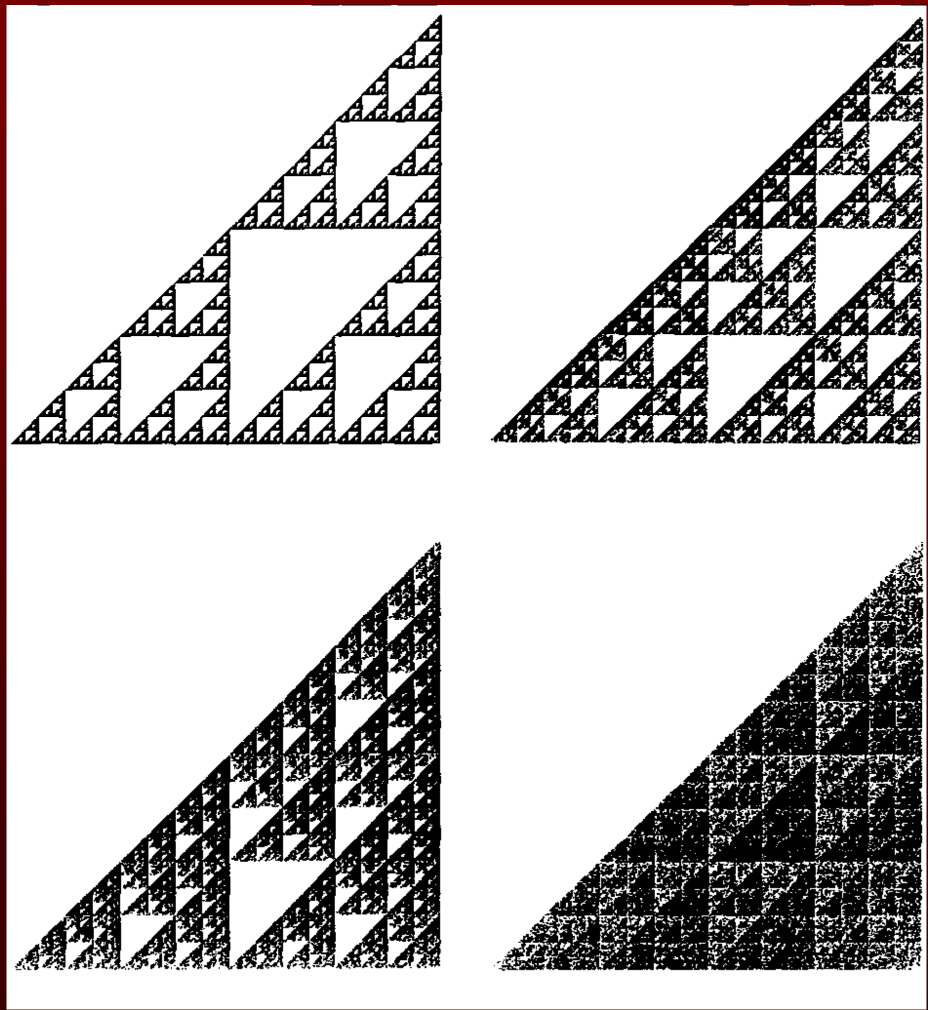


B=3

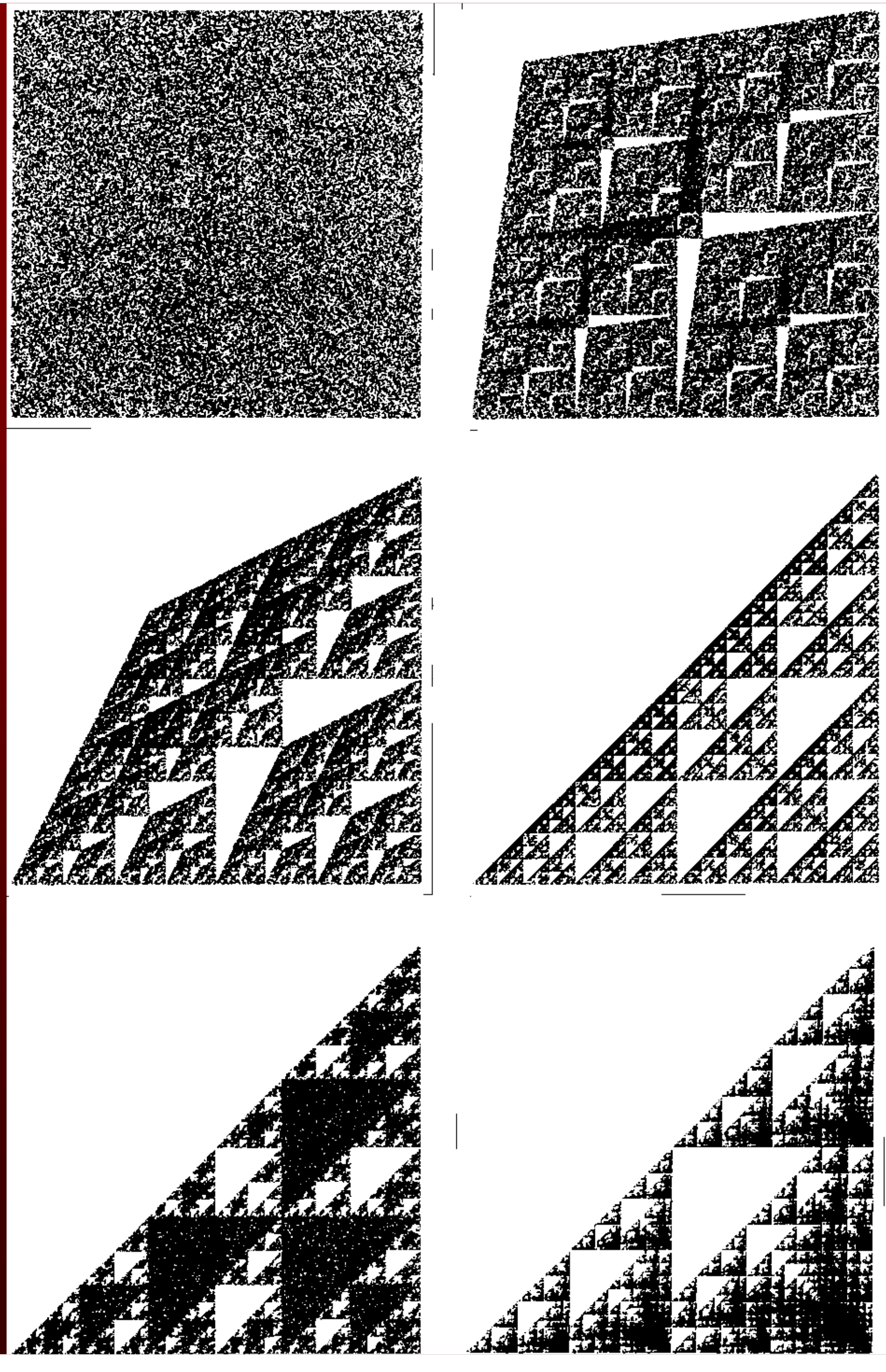
B 1.7, 2,
2.1, 2.4



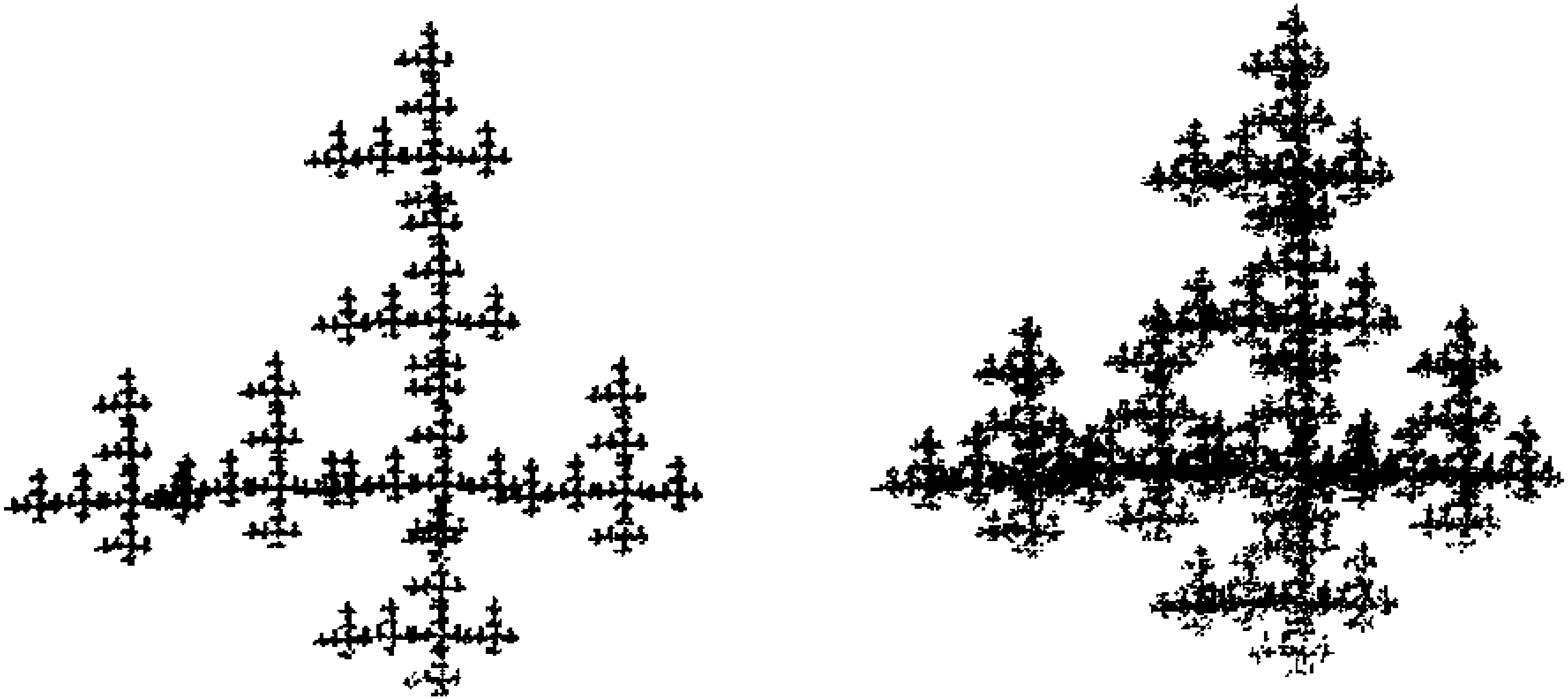
a) Original situation
b) The centre of weight added
c) Another point in the centre of the right edge added
d) Another point in the centre of the bottom edge added



4th vertex moved to the triangle



Free art – "Forest Lake" – winter and summer



- Rotations can be included

  X=SC*

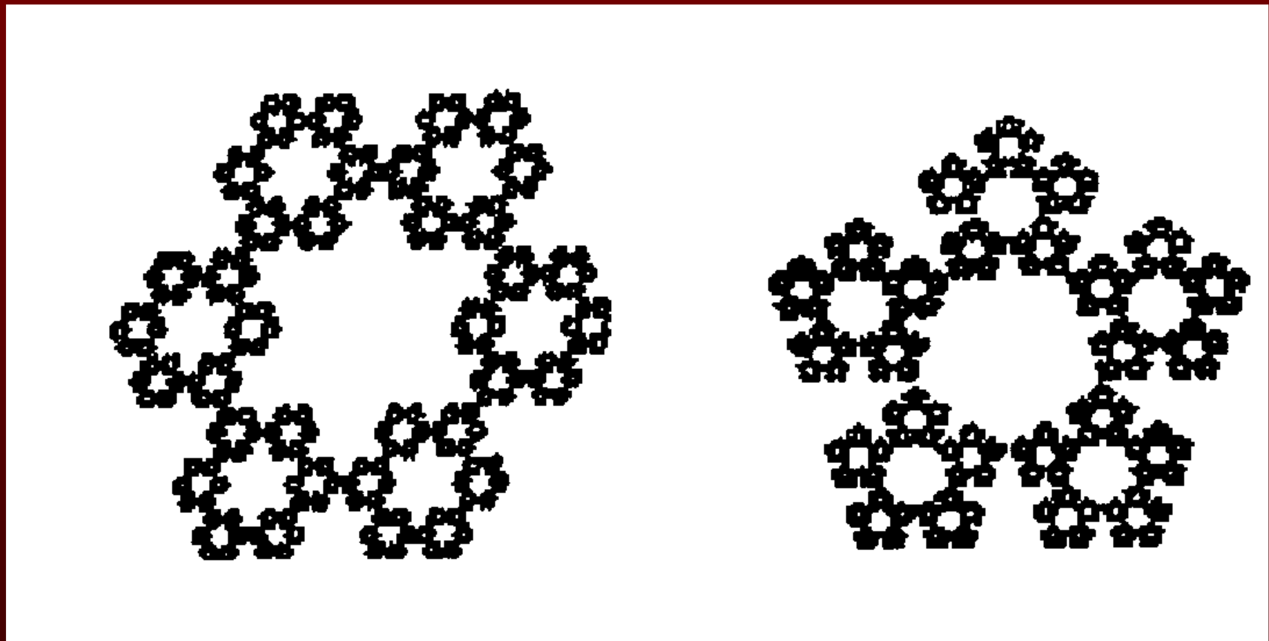
  (COS(TH)*(XP-XF(I))-SIN(TH)*(YP-YF(I)))+XF(I)

  Y=SC*

  (SIN(TH)*(XP-XP(I))+COS(TH)*(YP-YF(I))+YF(I)

- Another possible modification: various Bs for each vertex (instead of SC then SC(I)) b)R. L. Dewaney, 1995

A change of the distance in which we go to a vertex:



c) I.Kolingerová, P. Lobaz

Constrained fractals: iterations include some constraining area

Compute x_{i+1},y_{i+1}

if (x_{i+1},y_{i+1}) outside a constraining area then

begin

$x_i := x_{i+1}; \ y_i := y_{i+1};$

plot (x_i, y_i)

end

- If we enter the constraining area, we leave the old point

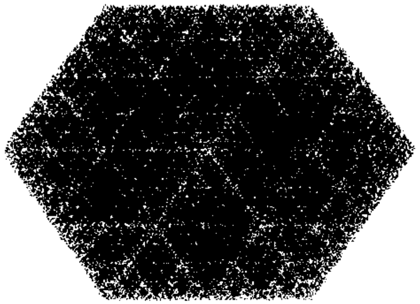


Figure 1: Chaos Game on 6 points

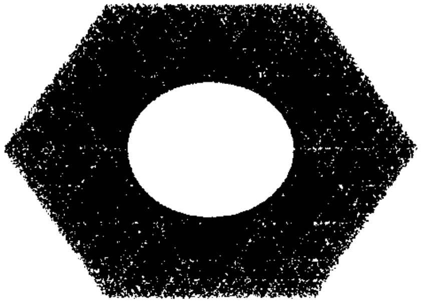


Figure 2: A circle as a constraining area

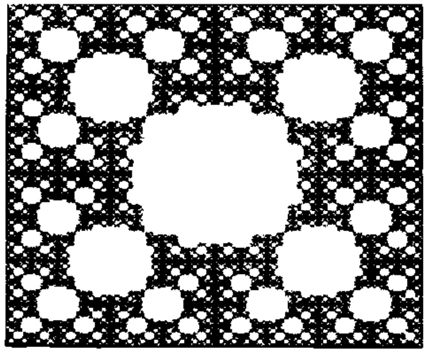


Figure 3: Fractal with a constraint : a square + a circle

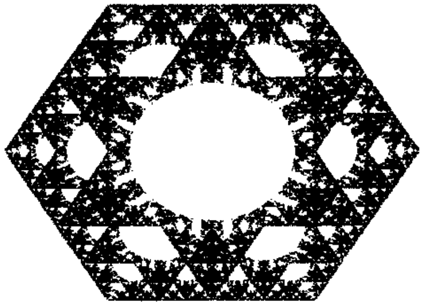


Figure 4: Fractal with a constraint : a hexagon + a circle

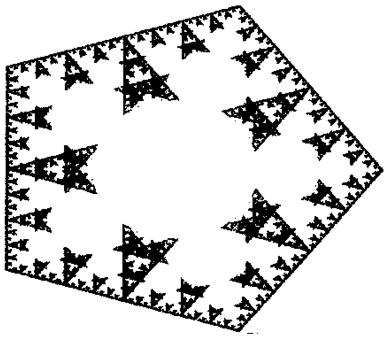


Figure 5: Fractal with a constraint : a pentagon + a circle

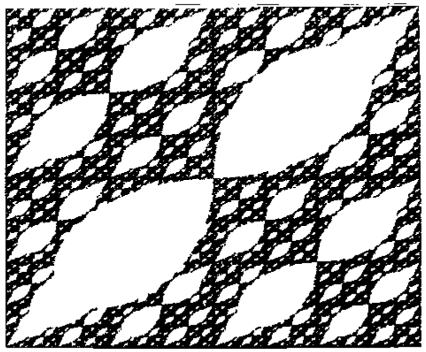


Figure 6: Fractal with a constraint : a square + $(x^2+y^2)^{1.5} \leq 2axy, xy \geq 0$

- A fractal can be used as a constrained area
- Up to 10 vertices, then not distinct enough

Figure 7: Fractal with a constraint : a hexagon + $(x^2+y^2)^2 \leq ay(3x^2-y^2)$

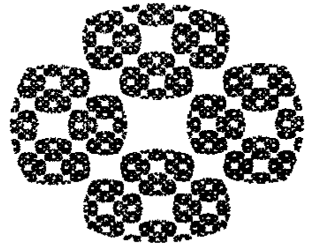


Figure 8: Fractal with a constraint : a square + an outside of a circle

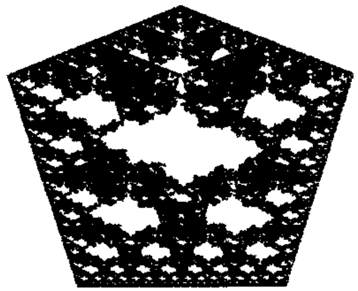


Figure 9: Fractal with a constraint : a pentagon + a filled Julia set

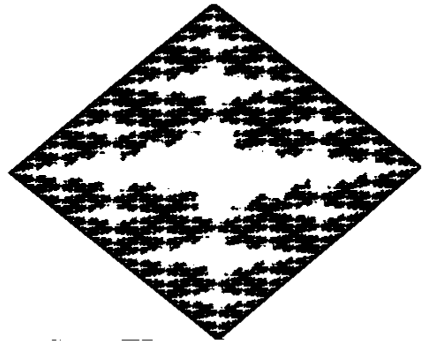


Figure 10: Fractal with a constraint : a quadruple + a filled Julia set

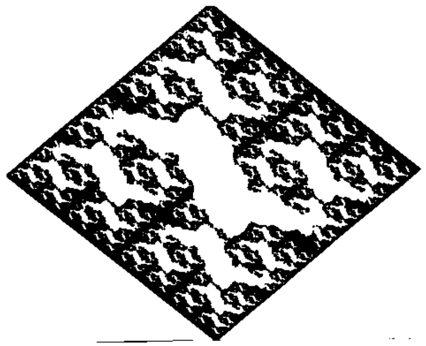


Figure 11: Fractal with a constraint : a quadruple + a filled Julia set

- Points can be colored according to the number of hits