Generating Self-Symmetrical Fractals by Hyperincursive Automata and Multiple Reduction Copy Machine

Daniel M. DUBOIS
Centre for Hyperincursion and Anticipation in Ordered Systems,
CHAOS asbl, Institute of Mathematics, B37, University of Liège
Grande Traverse 12, B–4000 LIEGE 1, Belgium
Fax: + 32 4 366 94 89 – Email: Daniel.Dubois@ulg.ac.be

Mathieu BELLY
Liège, Belgium
Mathieu.Belly@skynet.be

Abstract

This paper shows that different algorithmic methods can generate self-symmetrical Sierpinski fractals. A first category deals with a hyperincursive generator based on a composition rule applied to a defined path in the frame. A second category of algorithms is based on a recursive generator obeying certain symmetries. This paper will consider generalised Sierpinski fractals generated by modulo 2 and modulo 3. Even and odd modulo give rise to very different properties of symmetry.

Keywords : Sierpinski Gasket, Fractal, Recursive Generator, Hyperincursive Frame, Algorithms.

1 Introduction

Let us recall a simple generator of Sierpinski gasket based on cellular automata. A one-dimensional network of cellular automata is represented by a vector of automata states starting with initial values at time t=0. A set of rules defines how the states change at every clock time. A simple rule consists of computing the value of the state of each automaton at time t+1 by the sum modulo M of itself and its left neighbour at the preceding time t, for each clock time t=0,1,2,3, etc. The following recursive automata exhibit fractal structures

$$X(n,t+1) = [\, X(n,t) + X(n-1,t) \,] \bmod M \qquad (1)$$

with t=0,1,2,... and n=1,2,..., starting with initial conditions X(n,0), n=1,2,... at time t=0 and boundary conditions X(0,t) at each time step t=1,2,... , where mod M is the modulo M. For M=2, the fractal is the Sierpinski gasket given in Figure 1a. At each clock time, the order in which the computations are performed is without importance (parallel iterations).

	n = 0 1 2 3 4 5 6 7 8	n = 0 1 2 3 4 5 6 7 8
t = 0	0 1 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0
t = 1	0 1 1 0 0 0 0 0 0	0 1 1 1 1 1 1 1 1
t = 2	0 1 0 1 0 0 0 0 0	0 1 0 1 0 1 0 1 0
t = 3	0 1 1 1 1 0 0 0 0	0 1 1 0 0 1 1 0 0
t = 4	0 1 0 0 0 1 0 0 0	0 1 0 0 0 1 0 0 0
t = 5	0 1 1 0 0 1 1 0 0	0 1 1 1 1 0 0 0 0
t = 6	0 1 0 1 0 1 0 1 0	0 1 0 1 0 0 0 0 0
t = 7	0 1 1 1 1 1 1 1 1	0 1 1 0 0 0 0 0 0
t = 8	0 1 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0

Figure 1a-b: Fractal Sierpinski gasket computed from recursive eq. 1 and incursive eq. 2.

Dubois (1992) proposed a new type of automata (the fractal machine) where the order in which the computations are performed is to be ruled. The following hyperincursive automata give rise to fractals of the same class as Sierpinski gasket

$$X(n,t+1) = [X(n,t)) + X(n-1,t+1)] \bmod M \qquad (2)$$

This equation is an incursion, an inclusive or implicit recursion, because the state of an automaton is a function of another automaton at the future time. This is computed in a sequential order, in giving initial conditions X(n,0) and boundary conditions X(0,t+1) at the future time t+1, for each time t=0,1,2, ... Figure 1b gives the Sierpinski gasket from the incursive eq. 2 with M = 2. Such incursive automata will be called hyperincursive automata because there are a lot of different paths which can be defined for computing successively the automata, as this will be shown in this paper. Let us notice that such hyperincursive system sometimes may exhibit uncertainty and indecidability when the system defined itself its boundary conditions in a self-referential way (Dubois, 1996).

This paper shows that different algorithmic methods can generate self-symmetrical fractals. The Sierpinski fractal is taken as an example.
A first category deals with global permutations of rows of the fractal frame from hyperincursive automata. Self-symmetrical Sierpinski fractals will be generated by modulo 2 and 3. With an odd modulo 3, very different properties of symmetry, defined in Dubois (1996c), exhibit original self-symmetrical fractals, never presented in the literature at our knowledge.
A second category of algorithms is based on local translations, permutations and rotations inside the fractal frame. Peitgen et al (1992) uses a recursive generator obeying certain symmetries. These fractals are based on modulo 2.
Let us first introduce the concept of fractal dimension related to self-symmetry.

2 Fractal Dimensions and Self-Symmetry

B. B. Mandelbrot (1983) defined a fractal dimension dealing with fractional dimension instead of integer dimension. There is no more a characteristic length to define the scale of the system. There is an scale invariance: in looking at a fractal pattern at different scales, a similar or affine pattern appears. The fractal dimensions can be computed by box counting (Peitgen et al, 1992) from the relation

$$D = [\ln N(d_{n+1}) - \ln N(d_n)]/\ln (d_n/d_{n+1}) \qquad (3)$$

where $N(d_n)$ represents the number of boxes, of length d_n, containing at least one automaton at state different of 0, to cover the pattern, in considering different scales n.
This fractal dimension can be related to the measure of information of Shannon.
The box counting and information are embedded in the Rényi q^{th} order dimensions

$$D_q = I_q(s)/\ln(1/s) \qquad (3a)$$

where

$$I_q(s) = (1/(1-q))\ln \Sigma\, p_k{}^q \qquad (3b)$$

is the q^{th} order Rényi information. In eqs. 3ab, p_k are the automata state probabilities in the k^{th} box of length s, in assuming that they sum up to 1, $\Sigma\, p_k = 1$.
For $q = 0$

$$I_0(s) = \ln N(s) \qquad (3c)$$

where N(s) is the number of non empty boxes of linear size s. So $D_0 = D$, the usual box counting fractal dimension.
For $q = 1$

$$I_1(s) = -\Sigma\, p_k \ln p_k \qquad (3d)$$

is the Shannon information, and D_1 is the information dimension.
For $q = 2$, D_2 is the correlation dimension, etc. In general $D_p \geq D_q$ for any $p<q$. For fractals with uniform structure, like those considered in this paper, all dimensions D_q have the same value.

Very curiously, the fractal dimension is not dependent on the symmetry properties of the fractal patterns. So, many different fractal patterns with the same fractal dimension can be generated with different symmetries.

Euclidean geometry deals with integer space-time dimensions. If all the points of a space can be filled with points, the fractal dimension is the Euclidean dimension: D=1 for a line, D=2 for a surface, and D=3 for a volume. Let us consider the following Euclidean space of dimension 2 in Figs 2 ab. The space in Fig. 2a is divided in 4 boxes of length 1/2 and Fig. 2b is divided in 16 boxes of length 1/4. All the boxes can be filled with points.
In Fig. 2a, $d_1=1/2$ and $N(d_1)=4$, and in Fig. 2b, $d_2=1/4$ and $N(d_2)=16$, so
$D = [\ln N(d_2) - \ln N(d_1)]/\ln (d_1/d_2) = \ln (16/4)/\ln(4/2) = 2$
and the dimension is D = 2 for the two dimensions Euclidean space.

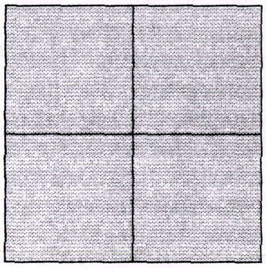


Figure 2a

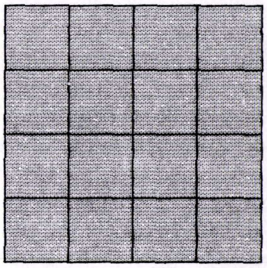


Figure 2b

In fractal geometry, the space cannot be filled with points. This is not an Euclidean geometry. Let us consider the Sierpinski gasket at Figures 3ab. The space in Fig. 2a is divided in 4 boxes of length 1/2 and Fig. 2b is divided in 16 boxes of length 1/4. All the boxes cannot be filled with points. In Fig. 3a, $d_1=1/2$ and $N(d_1)=3$, and in Fig. 3b, $d_2=1/4$ and $N(d_2)=9$, so $D = [\ln N(d_2) - \ln N(d_1)]/\ln (d_1/d_2) = \ln (9/3)/\ln(4/2) = \ln (3/2)$, and the dimension is $D = \ln (3/2) = 1.585$ for this two dimensions Sierpinski fractal space. The pattern is self-similar.

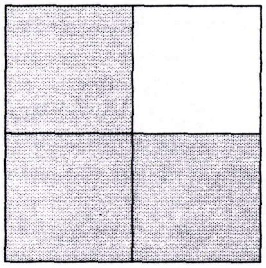


Figure 3a

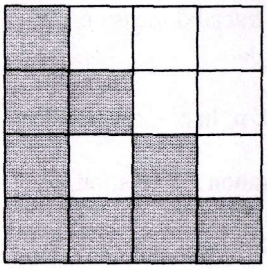


Figure 3b

There are a set of different fractals that can be generated with the same fractal dimension.

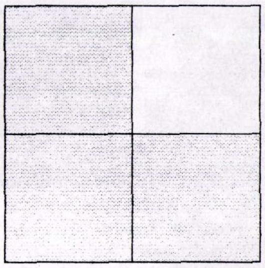


Figure 3a'

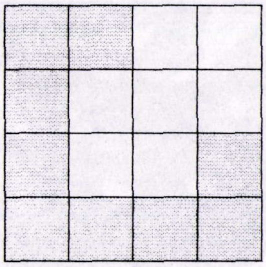


Figure 3b'

For example, Fig. 3a' is the same as Fig. 3a but the Fig. 3b' is generated in permuting the first 2 lines and the last two columns of Fig. 3b. The fractal in Fig. 3ab has the same fractal dimension as Fig. 3a'b'. These fractals are self-similar and are also self-symmetrical, but their symmetries are different.

So the fractal dimension does not take into account the self-symmetry property of fractals. It could be of interest to define a symmetry dimension for self-similar patterns in view of obtaining a higher order information about self-symmetry properties of patterns.

Let us show an example of generation of fractals with the Multiple Copy Reduction Machine.

With a computer, you make 3 copies of any initial pattern, like the picture of the cat "Coquine" in Figure 4a, at a scale 1:2 and you paste the 3 copies in the Figure 4b with the same symmetry as the Sierpinski generator. You make then 3 copies of this figure 4b with a scale 1:2 and you generate the Figure 4c. After several successive reductions of scale by 1:2 with 3 copies, you finally obtain the Figure 4f. What is the surprise?
The picture of the cat disappears progressively (in fact this picture becomes so small that one cannot see the cat) with the emergence of the Sierpinski gasket given by fractal triangles. So the generative process destroys the local information given by the picture to replace it by the a pattern, the information of which being the symmetry of the generating process. This confirms the fact that the main information in such a fractal process deals with the symmetry property of the generator. The information of the final fractal pattern is embedded implicitly in the symmetry of the dynamics of the fractal process independently of the initial condition given by any pattern.
With the same initial condition, different fractal patterns emerge depending on the symmetry property of the generator.

Let us show that a set of patterns with the same fractal dimension can be generated with different self-symmetries.

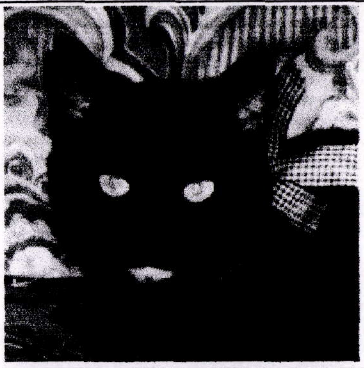


Figure 4a: initial pattern given by a cat.

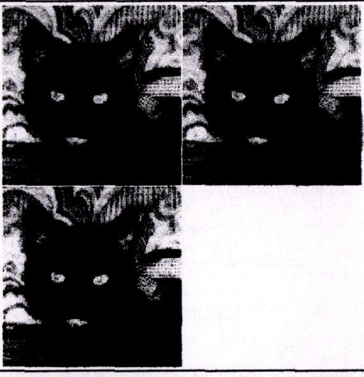


Figure 4b: 3 copies at scale 1:2.

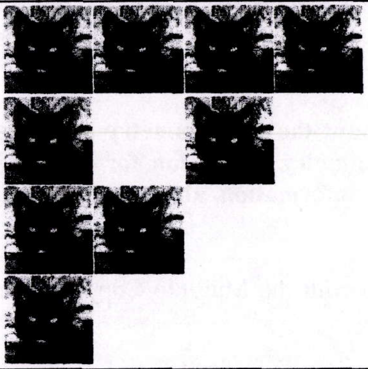


Figure 4c

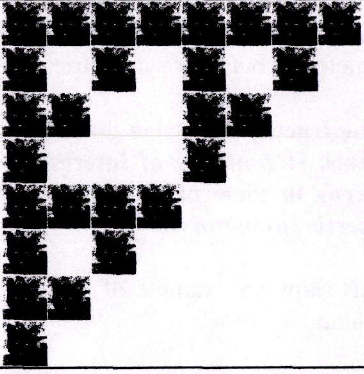


Figure 4d

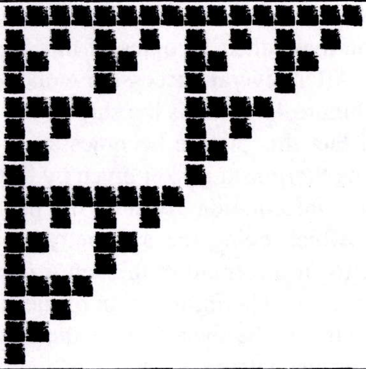


Figure 4e

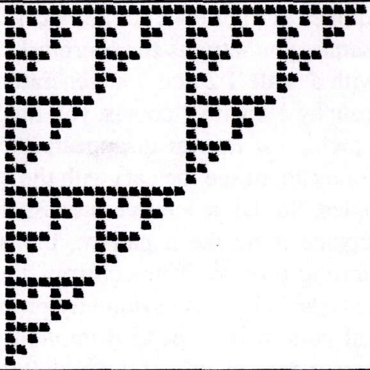


Figure 4f: emergence of a fractal pattern.

3 Hyperincursive Automata

The general theory of hyperincursivity (Dubois & Resconi, 1992) deals with dynamical systems with inclusive recursion called incursion.
In this framework, a new type of cellular automata, called hyperincursive automata, was proposed from the fractal machine (Dubois, 1992; Dubois and Resconi, 1992).

Hyperincursive automata are defined by:

1. A frame.
2. A composition rule.
3. A path.

The path must be explicitly defined in order to compute the successive automata states in an ordered way. With the same frame and the same composition rule, many different processes can be generated in choosing different paths.

Two-dimensions space automata will be considered for explaining the importance of the paths in the computation of hyperincursive automata.
A two-dimensions extension of eq. 2 is given by

$$X(i,j) = [\, X(i,j-1) + X(i-1,j) \,] \bmod M \quad \text{with } i=1,2,\ldots \text{ for each } j=1,2,\ldots \tag{4}$$

The computation of this eq. 4 is made in considering a path given by the successive ordered iterations of the automata lines by lines (index j) for each column (index i).

With M=2, Figure 5a gives the simulation of eq. 4 with the conditions $X(0,1)=1$, $X(i,1)=0$ for $i=1,2,3,\ldots$ and $X(0,j)=0$ for $j=2,3,\ldots$ This is the Sierpinski gasket, a fractal pattern.

Another fractal pattern is generated with M=3, given in Figure 6a.

In choosing different paths for the iterations with the same frame and the same composition rule, a lot of different fractal patterns can be generated from eq. 4. We can say that these fractals obey a "fractal dimension invariance".

In defining ruled paths by n(i) and m(j), eq. 4 becomes

$$X[n(i),m(j)] = (\, X[n(i),m(j-1)] + X[n(i-1),m(j)] \,) \bmod M \tag{4a}$$

where n(i) and m(j) are new ordered paths as functions of $i = 1,2,\ldots,I$ for each $j = 1,2,\ldots,J$.
This is a forward hyperincursive automaton.

A backward hyperincursive automaton is defined by

X[n(i),m(j)] = (X[n(i),m(j+1)] + X[n(i+1),m(j)]) mod M (4b)

where i = I,I−1,I−2,...,1 and j = J,J−1,J−2,...,1, with final (X[n(I),m(J)]) and boundary conditions.

A lot of ordered paths exist. This paper will consider only a few ones in view of explaining the new concept of "ruled path".

The sole condition is that the ruled paths generate patterns with the same fractal dimension.

Let us explain very simply some ruled paths.

3.1 Modulo 2 Self-Symmetrical Sierpinski Fractals (Dubois, 1996c)

Figures 5abcdefghi show 9 fractals belonging to the class of Sierpinski gasket generated from eqs. 4 with M=2.

These fractals are similar to fractals given by Peitgen et al (1992), with a recursive generator obeying certain symmetries called the Multiple Copy Reduction Machine.

But our hyperincursive method is totally different. The computation starts with an initial value equal to 1 for a first automaton, all the others being at 0. Then we apply the eqs. 4 in choosing different algorithms for defining the path [(n(i),m(j)].

Figures 5abcdefghi

The Sierpinski fractal 5a, is computed from eq. 4a, n(i) = i, i = 1,2,3,... for each m(j) = j, j = 1,2,3,... , that is to say lines by lines, from the left to the right, as already shown. This means that the path index follows the successive order 1,2,3,4,5,6,7,8,...

A simple way to generate another path is to consider the recursive rule for n(i) and m(j): starting with the sequence 1,2, you add 2 to obtain (3,4), and you put the inverse after to obtain 1,2,**4,3**; then you repeat the procedure: you add 4 to obtain (5,6,8,7) and put the inverse to obtain: n(i) = 1,2,4,3,**7,8,6,5**; and so on.

A recursive algorithm can perform this ordering.

In taking m(i) = n(i), the last fractal in Figure 5i is obtained.

The fractal 5e in the middle in the Figure 5 is also symmetrical and was computed with the backward eq. 4b.

All the asymmetrical fractals are generated by one path n(i) or m(j) and the current index j or i, respectively.

3.1.1 Part of Basic Listing for a few Modulo 2 Self-symmetrical Fractals

A

```
'
program DDSierp
xmin = 1
xmax = size / 2        'size = 256
ymin = 1
ymax = size / 2

For i = 0 To size: For j = 0 To size
    m(i, j) = 0
Next j: Next i
    m(0, 1) = 1

For Y = ymin To ymax
  For X = xmin To xmax
    m(X, Y) = (m(X − 1, Y) + m(X, Y − 1)) Mod 2
  Next
Next
```

B

```
n(1) = 1: n(2) = 2
x1 = 2

For z = 1 To (Log(size) − Log(x1)) / Log(2) − 1
  For X = 1 To x1
    n(x1 + X) = x1 + n(x1 − X + 1)
  Next
  x1 = x1 + x1
Next
' second inverse permutation function
 For i = 0 To xmax
  p(n(i)) = i
Next i
```

```
' 'different fractals
```

C

```
For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = (xmax − n(ii) + 1) ' ii  ' n(ii)
    n2 = (ymax − n(jj) + 1) ' jj  ' n(jj)
    PSet (ii, jj), (m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = ii  ' n(ii)
    n2 = (ymax − n(jj) + 1)
    PSet (ii, jj), (m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = n(ii)
    n2 = (ymax − n(jj) + 1)
    PSet (ii, jj), (m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = n(ii)
    n2 = n(jj)
    PSet (ii, jj), (m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = ii
    n2 = n(jj)
    PSet (ii, jj), (m(n1, n2) * 6 + 9)
  Next
Next
```

D

```
For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = p(ii)
    n2 = (ymax − p(jj) + 1)
    PSet (ii, jj),(m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = p(ii)
    n2 = p(jj)
    PSet (ii, jj),(m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = ii
    n2 = p(jj)
    PSet (ii, jj),(m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = ii
    n2 = jj
    PSet (ii, jj),(m(n1, n2) * 6 + 9)
  Next
Next

For jj = 1 To ymax
  For ii = 1 To xmax
    n1 = (ymax − n(n(ii)) + 1)
    n2 = (ymax − n(jj) + 1)
    PSet (ii, jj),(m(n1, n2) * 6 + 9)
  Next
Next
```

```
For jj = 1 To ymax
    For ii = 1 To xmax
        n1 = (xmax – p(ii) + 1) 'ii   'n(ii)
        n2 = (ymax – p(jj) + 1) 'jj   'n(jj)
        PSet (ii, jj), (m(n1, n2) * 6 + 9)
    Next
Next
```

```
For jj = 1 To ymax
    For ii = 1 To xmax
        n1 = n(ymax + 1 – n(jj))
        n2 = n(ii)
        PSet (ii, jj),(m(n1, n2) * 6 + 9)
    Next
Next
```

```
For jj = 1 To ymax
    For ii = 1 To xmax
        n1 = ii 'n(ii)
        n2 = (ymax – p(jj) + 1)
        PSet (ii, jj), (m(n1, n2) * 6 + 9)
    Next
Next
```

3.2 Modulo 3 Self-Symmetrical Sierpinski Fractals (Dubois, 1996c)

Figures 6,7,8 show fractals generated with eqs. 4ab with M = 3.

The first fractal 6a was computed in from eq. 4a using the current indexes: n(i) = i, i = 1,2,3,... and m(j) = j, j = 1,2,3,... as already shown.
Instead of considering sequences of length 2^s, s = 1,2,3,... as in the preceding case (with modulo 2), the sequences are now chosen with length 3^s, s = 1,2,3,... due to the modulo 3.

But with such an odd modulo, this is not possible to create mirror symmetries as in the preceding case.

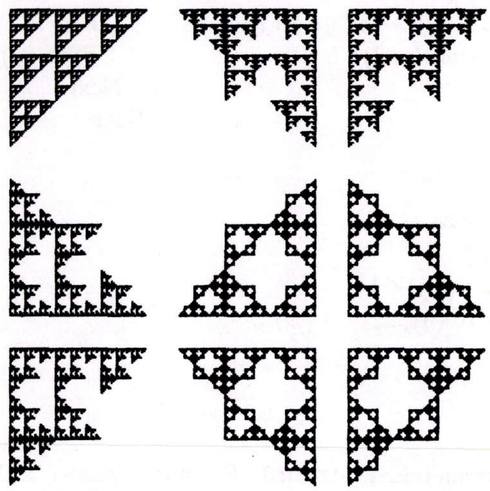
Let us show the basic idea for the ordering of the path.

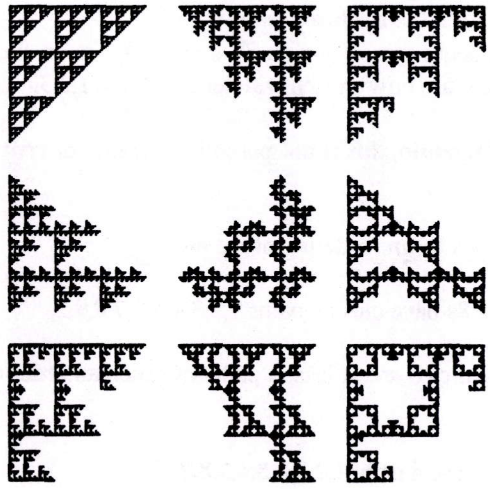
The first fractals 6a, 7a, 8a have current paths 1,2,3,4,5,6,7,8,9,...

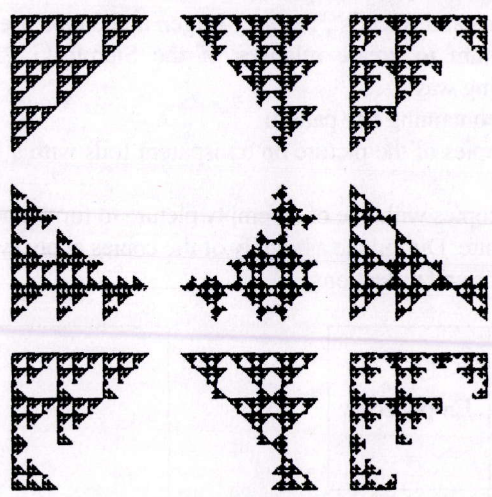
The other fractals in Figures 6 use at least a path with such an ordering for n(i) or/and m(j): 1,2,3,**6,5,4**,7,8,9,...

The fractals in Figures 7 use a path: 1,2,3,**6,5,4,9,8,7**,...
and the fractals in Figures 8, a path: 1,2,3,4,5,6,**9,8,7**,...

Three types of inversion have been applied giving rise to different self-symmetries in the fractals with the same fractal dimension.


Figures 6abcdefghi


Figures 7abcdefghi


Figures 8abcdefghi

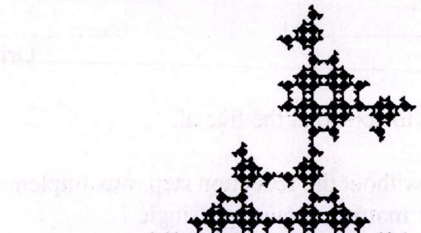
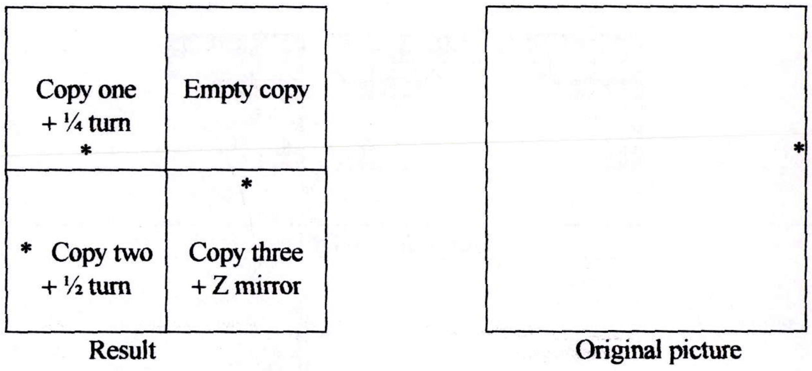


Figure 9: Enlargement of fractal 8e

4 Multiple Reduction Copy Machine

Let us show the algorithms used by Peitgen et al, 1992.
In chapter 5.3 of "Chaos and fractals", Peitgen, Jürgen and Saupe use a "multiple reduction copy machine" algorithm to create relatives of the Sierpinski gasket. This algorithm proceeds in the following way:
Take a square picture containing any pattern.
Make three reduced copies of the picture on transparent foils with a linear reduction factor of ½.
Assemble these three copies with one of an empty picture to form a new square of the same size as the starting picture. During the assembly of the copies apply symmetry operations on the copies, like rotations and reflections.



This procedure is iterated to construct the fractal.

A similar procedure, but without the reduction step, was implemented on Boolean matrixes while starting with a 1 x 1 matrix containing a single 1.
This gives for the simple Sierpinski gasket:

$$\begin{array}{|c|}\hline 1 \\ \hline\end{array} \quad \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 1 & 1 \\ \hline\end{array} \quad \begin{array}{|cc|cc|}\hline 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ \hline\end{array} \quad \begin{array}{|cccc|cccc|}\hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline\end{array}$$

and so on ...

For the particular case of the Sierpinski gasket itself, the operation is equivalent to an iterated Kronecker product with the following matrix:

$$\begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}$$

If we add a ¼ turn rotation on cell (1,1) and an S mirror on cells (1,2) and (2,1) we obtain:

$$\begin{array}{|c|} \hline 1 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 1 & 1 \\ \hline \end{array} \quad \begin{array}{|cc|cc|} \hline 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ \hline \end{array} \quad \begin{array}{|cccc|cccc|} \hline 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline \end{array}$$

A different fractal is generated for all different combinations of the symmetry operations token 3 by 3 (Identity, 1/4 turn, ½ turn, ¾ turn, horizontal mirror, vertical mirror, S mirror and Z mirror). The whole set of 224 different fractals obtained is identical to the one published by Peitgen, Jürgen and Saupe.

4.1 A Few Typical Fractals

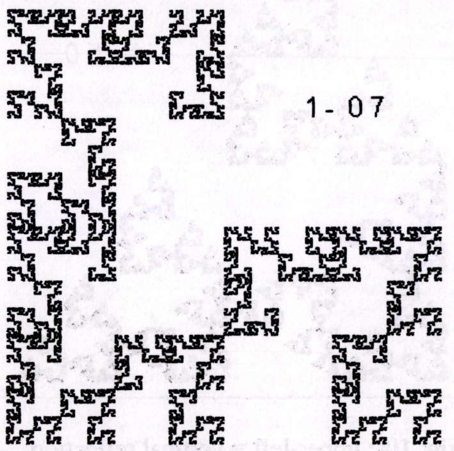


Figure 10a: upper-left = ¼ turn
lower-left = copy
lower-right = S reflection

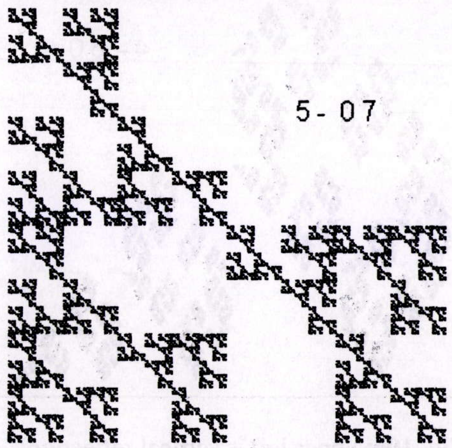


Figure 10b: upper-left = Z reflection
lower-left = copy
lower-right = S reflection

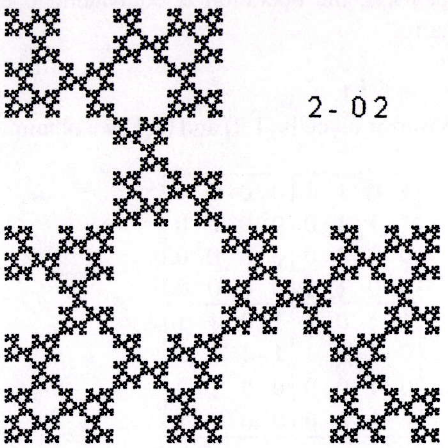


Figure 10c: upper-left = ½ turn
lower-left = copy
lower-right = ½ turn

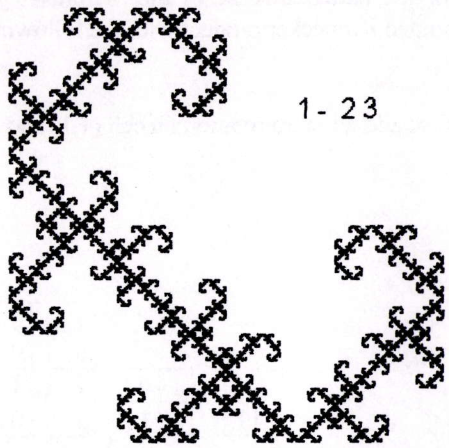


Figure 10d: upper-left = ¼ turn
lower-left = ½ turn
lower-right = ¾ turn

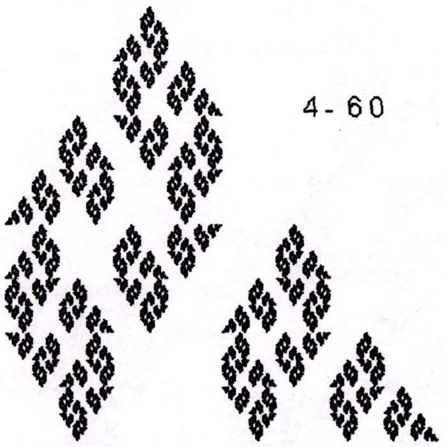


Fig. 10e: upper-left = vertical reflection
lower-left = horizontal reflection
lower-right = copy

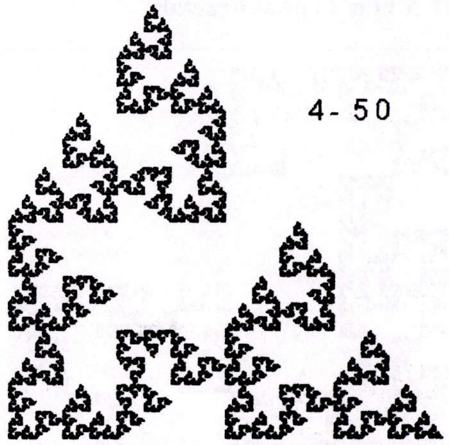


Fig. 10f: upper-left = vertical reflection
lower-left = Z reflection
lower-right = copy

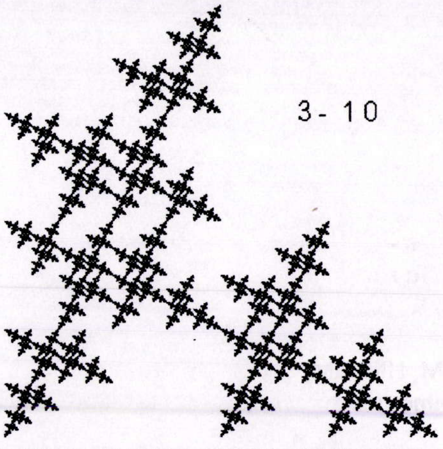
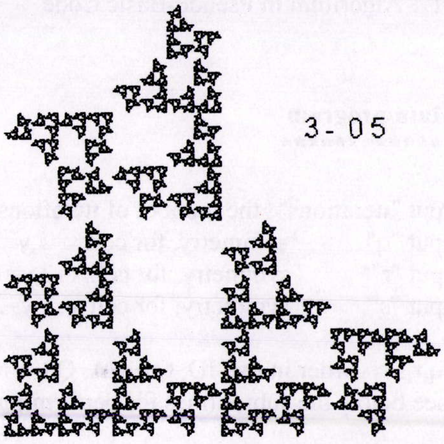
	

Figure 10g: upper-left = ¾ turn
lower-left = ¼ turn
lower-right = copy
	Figure 10h: upper-left = ¾ turn
	lower-left = copy
	lower-right = Z reflection

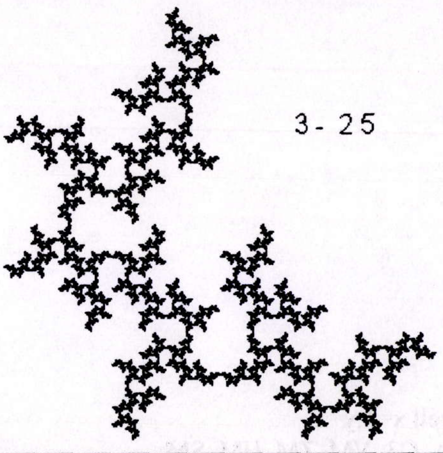
	

Figure 10i: upper-left = ¾ turn
lower-left = ½ turn
lower-right = Z reflection	

4.1.1 Algorithm in Pseudo-Basic Code

```
'

 Main program
 *************

input "iterations" ' the number of iterations
input "q"        ' symmetry, for cell x=1,y=1          s -
input "r"        ' symmetry, for cell x=2,y=1          q r
input "s"        ' symmetry, for cell x=1,y=2

' "q,r,s" = order in list: ID, Q1, HA, Q3, VM, ZM, HM, SM
'  see below the subroutines for performing symmetries

p = 0            ' p takes 2 values 0 or 1
n = 1 - p        ' n takes 2 values 0 or 1
                 ' these values change at each iteration

l = 1            ' size of source square matrix = 1 x 1

a = 1
b = 1
f(p, a, b) = 1   ' f(p, a, b) is the source matrix
                 ' f(n, a, b) is the target matrix


For iter = 1 To iterations

        For i = 1 To l      'x

                For j = 1 To l  'y

                        a = i: b = j               ' cell x=1,y=1
                         On q GoSub ID, Q1, HA, Q3, VM, ZM, HM, SM
                        if f(n, a, b) = 1 then set pixel of coordinates (a,b)
                        else clear that pixel

                        a = l + i: b = j           ' cell x=2,y=1
                        On r GoSub ID, Q1, HA, Q3, VM, ZM, HM, SM
                        if f(n, a, b) = 1 then set pixel of coordinates (a,b)
                        else clear that pixel
```

```
                  a = i: b = l + j           ' cell x=1,y=2
                  On s GoSub ID, Q1, HA, Q3, VM, ZM, HM, SM
                  if f(n, a, b) = 1 then set pixel of coordinates (a,b)
                  else clear that pixel

                  a = l + i: b = l + j  ' cell x=2,y=2
                  f(n, a, b) = 0        ' always 0
                  clear pixel of coordinates (a,b)

          Next j

     Next i

     p = n          ' source matrix becomes target
     n = 1 - p      ' target matrix becomes source
     l = l + l      ' double matrix size

Next iter

END


'Subroutines for performing the symmetries
*****************************************

ID:               ' identity                 VM:                        ' vertical  mirror
f(n, a, b) = f(p, i, j)                      f(n, a, b) = f(p, l + 1 - i, j)
Return                                       Return

Q1:     ' quater turn clockwise              ZM:                        ' mirror with
f(n, a, b) = f(p, l + 1 - j, i)              orientation Z
Return                                       f(n, a, b) = f(p, j, i)
                                             Return

HA:               ' half turn                HM:                        ' horizontal mirror
f(n, a, b) = f(p, l + 1 - i, l + 1 - j)      f(n, a, b) = f(p, i, l + 1 - j)
Return                                       Return

Q3:               ' 3 quater turn            SM:                ' mirror with orientation S
f(n, a, b) = f(p, j, l + 1 - i)              f(n, a, b) = f(p, l + 1 - j, l + 1 - i)
Return                                       Return
```

5 Conclusion

The main purpose of this paper was to generate self-symmetrical fractals based on the Sierpinski Gasket.

Two types of algorithms were used.

At one hand, hyperincursive automata, developped by D. M. Dubois, deal with automata computations in a sequential order where the order in which the computations of the states of the successive automata is ruled. This defines paths of propagation of activation of automata. The state of an automaton depends on the states of other automata at any distance.

At the other hand, the Multiple Copy Reduction Copy Machine as described by H.-O. Peitgen, H. Jürgens and D. Saupe was used to generate self-symmetrical fractals with rotations and mirrors rules.

The curious fact is that the Fractal, Information, Correlation, etc Dimensions at any order are equal for fractals with uniform structure and does not depend of their symmetries. So a lot of fractals with different self-symmetries are defined by the same fractal dimensions.

Symmetry is one of the most important property in any systems.

It would be of high interest to find a Symmetry Dimension to characterise fractals with different self-symmetries.

References

Dubois, D., *Mathematical fundamentals of the fractal theory of artificial intelligence*, Invited paper, in New Mathematical tools in artificial intelligence, Communication & Cognition - Artificial Intelligence, 1991, vol 8, N°1 pp. 5-48.

Dubois D., *Fractal Algorithms for holographic Memory of inversible neural Networks*, Invited paper, in Issues in Connectionism: part II, Communication & Cognition Artificial Intelligence, 1991, vol. 8, N°2, pp. 137-189.

Dubois D., "*The hyperincursive fractal machine as a quantum holographic brain*", in Designing New Intelligent Machines: The Fractal Machine and the Huygens' Machine (COMETT programme), Communication & Cognition - Artificial Intelligence, 1992, vol. 9, N°4, pp. 335-372.

Dubois D. (1992), The fractal machine, Presses Universitaires de Liège, 375 p.

Dubois D. M. (1996a), "Introduction of the Aristotle's Final Causation in CAST: Concept and Method of Incursion and Hyperincursion". In F. Pichler, R. Moreno Diaz, R. Albrecht (Eds.): Computer Aided Systems Theory - EUROCAST'95. Lecture Notes in Computer Science, *1030*, Springer-Verlag Berlin Heidelberg, pp. 477-493.

Dubois D. M. (1996b), "A Semantic Logic for CAST related to Zuse, Deutsch and McCulloch and Pitts Computing Principles". In F. Pichler, R. Moreno Diaz, R. Albrecht (Eds.): Computer Aided Systems Theory - EUROCAST'95. Lecture Notes in Computer Science, 1030, Springer-Verlag Berlin Heidelberg, pp. 494-510.

Dubois D. M. (1996c), "Generation of Self-Symmetrical Fractals by Hyperincursive Automata", in Advances in Modeling of Anticipative Systems, ed. by G. E. Lasker, D. Dubois, B. Teiling, published by The International Institute for Advanced Studies in Systems Research and Cybernetics, pp. 46-50.

Dubois D. M. (1997). Generation of fractals from incursive automata, digital diffusion and wave equation systems, Invited paper. *Biosystems* 43, pp. 97-114.

Dubois D. M. (1998), Hyperincursive Methods for Generating Fractals in Automata Related to Diffusion and Wave Equations. Invited paper. Int. J. General Systems, Vol. 27(1-3), pp. 141-180.

Dubois D., and G. Resconi (1992), HYPERINCURSIVITY: a new mathematical theory, Presses Universitaires de Liège, 260 p.

Mandelbroot B. B. (1983), The Fractal Geometry of Nature, New York, Freeman.

Peitgen H.-O., H. Jürgens, D. Saupe (1992), Chaos and Fractals: New Frontiers of Science, Springer-Verlag.

Sierpinski W. (1915), Sur une courbe cantorienne dont tout point est un point de ramification. C. R. Acad. Paris 160, 302.

Sierpinski W. (1916), Sur une courbe cantorienne qui contient une image biunivoque et continue de toute courbe donnée. C. R. Acad. Paris 162, 632.