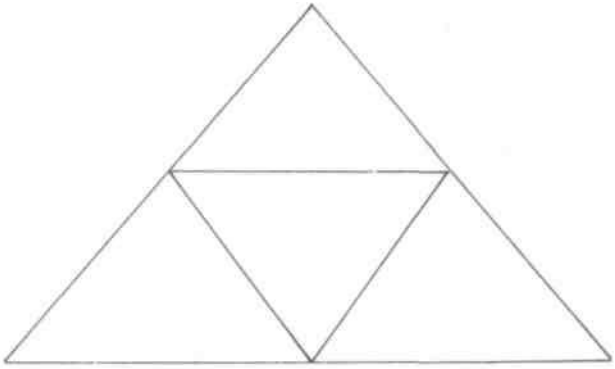
FRACTALES, SU ESTRUCTURA Y OBTENCIÓN

Ing. JESUS RUBEN AZOR MONTOYA
Profesor Titular de la Cátedra Estadística Aplicada
en la Facultad de Ingeniería, Universidad de Mendoza

SÍNTESIS HISTÓRICA

La revolución de los fractales se inició silenciosamente en 1872, cuando Weiertrass definió formalmente la existencia de curvas continuas sin derivada.

En aquel entonces parecían casos-patológicos sólo aptos para Iniciados. Lo siguió Peano que propuso una curva (o línea, ya que es difícil encontrar el término justo) que sin dejar de ser eso, es decir una línea, podía cubrir totalmente la superficie. Sierpinski en 1912, hace lo inverso: halla superficies de área nula por subdivisión fractal. El Triángulo de Sierpinski se forma quitándole a un triángulo otro interior, lo que reduce la superficie a 3/4 de su valor original:



Al repetir la operación un número infinito de veces, la superficie se anula.

En la figura 1, se aprecia la estructura obtenida después de 2.500 iteraciones. Algunos grandes matemáticos quedan desconcertados y como demostración se citan las cartas que escribieron Hermite y Poincaré, hablando del "horror" que provocan estas funciones o la "poca utilidad" que pueden tener.

El tema es retomado con gran vigor por Benoit Mandelbrot, un matemático polaco, educado en Francia, quien comienza a dar forma a las ideas

dispersas subyacentes de esos "raros objetos matemáticos". El estaba trabajando en la empresa IBM, pero pasa a la Universidad de Harvard en 1962, luego de estudiar problemas lingüísticos y de estadísticas de errores en sistemas de comunicación, temas que aparentemente no tenían relación con la geometría pero que luego unirá a través de su incesante elaboración del tema fractales, palabra que creara a partir de la expresión latina "fractus", quebrado. (Luis F. Rocha, Revista Telegráfica Electrónica).

Mandelbrot, seguido por Richard Vos, también de IBM, mostró que los fractales aleatorios generados por computadora ofrecían imágenes increíblemente realistas de los objetos de la naturaleza.

Al poco tiempo, empresas y universidades retomaron el trabajo. A la fecha, el uso de los fractales para generar imágenes naturales, se está difundiendo ampliamente.

Michael Barnsley, profesor de matemáticas del Georgia Institute of Technology de Atlanta, a principios de la década del ochenta, comienza un trabajo de análisis de imágenes para conseguir un conjunto de fractales que iterados aleatoriamente produjeran una imagen original (target). Los elementos relativamente pequeños de código que describen ese conjunto podrían entonces reemplazar el gran volumen de datos que se necesitaban para reconstruir la imagen original. En 1984, Barnsley y Demko habían logrado ya resultados interesantes, que despertaron el interés de la Agencia de Proyectos de Investigación Avanzada para la Defensa (DARPA) de Arlington, Virginia, que necesitaban una mejor comprensión de imágenes para todo uso: desde almacenar en forma más compacta las fotos de reconocimiento digital hasta reducir el tiempo necesario para reconocer un objeto en las fotos o transmitirlas a un puesto de comando en el campo de batalla. Los resultados llevaron a comprensiones de imágenes mayores que 1000 a 1.

SISTEMAS DE FUNCIÓN ITERADA

Se arrancará explicando cómo un conjunto de códigos de Sistema de Función Iterada (IFS) puede aproximar una imagen natural.

La teoría IFS es una extensión de la geometría clásica. Usa transformaciones afines, explicadas más abajo, para expresar relaciones entre partes de una imagen. Usando sólo estas relaciones, se definen y producen intrincadas figuras.

Las transformaciones afines pueden ser descriptas como combinaciones de rotaciones, escalamientos y traslaciones de los ejes coordenados en el espacio n-dimensional.

Un ejemplo en dos dimensiones es:

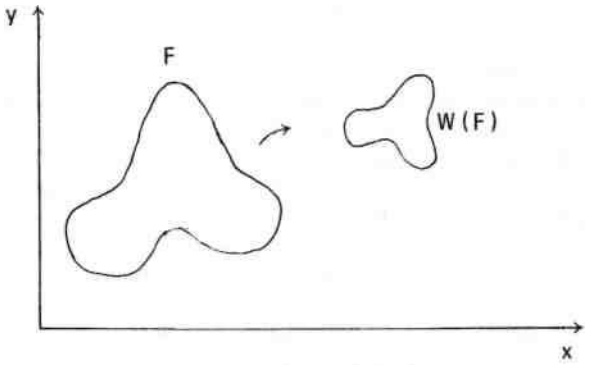
$$W(x, y) = (x + 2y - 3; - x + 3y + 2)$$

o expresado en forma matricial:

$$W\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & +2 \\ -1 & +3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -3 \\ +2 \end{bmatrix}$$

Esta transformación mueve el punto (0;0) a (-3;2) y mueve el (1;-.5) al (-3;-.5). A la transformación anterior se la denota con W. La notación W(S) denota la subimagen W sobre un conjunto de puntos de S.

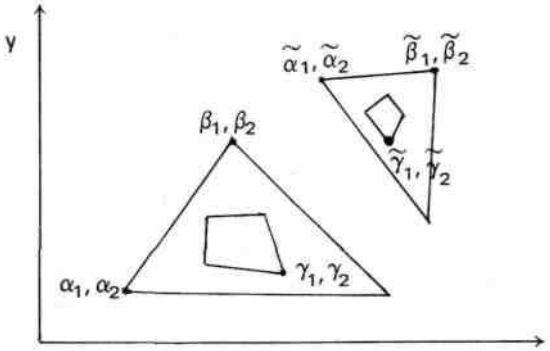
Se verá ahora qué hace W a la figura indicada con F, que está ubicada en el plano x-y:



El resultado es una nueva y comprimida figura, W(F). La transformación afín la ha deformado y movido.

Se dice que la transformación W es contractiva cuando la subimagen de la original presenta un área inferior a ésta.

Otro ejemplo de transformación afín contractiva se muestra a continuación:



La figura cuyas coordenadas se indican con el tilde ~, es una transformación afín contractiva de la otra.

La forma general para la transformación afín es:

$$W\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

Donde los coeficientes a, b, c, d, e, f son números reales. Si se conocen anticipadamente las traslaciones, rotaciones y escalamientos que se combinan para producir W, se pueden generar los valores de los coeficientes como sigue:

$$a = r\cos\varphi \quad ; \quad b = -s\,\mathrm{sen}\,\theta \quad ; \quad c = r\,\mathrm{sen}\,\varphi \quad ; \quad d = s\cos\theta$$

donde r es el factor de escalamiento sobre x, s es el factor de escalamiento sobre y, φ es el ángulo de rotación sobre x, θ es el ángulo de rotación sobre y, e es la traslación sobre x y f es la traslación sobre y.

Para encontrar la transformación afín que lleve de la imagen a la subimagen, basta con encontrar los números a, b, c, d, e, f para los cuales la transformación W tiene la propiedad:

W (Figura Grande) $\approx$ Figura Chica

Se marcan tres puntos de la Figura Grande y se determinan sus coordenadas (α_1, α_2); (β_1, β_2) y (γ_1, γ_2) respectivamente.

Se pueden determinar los valores de los coeficientes a, b y e resolviendo las tres ecuaciones lineales siguientes:

$$a\,\alpha_1 + b\,\alpha_2 + e = \tilde{\alpha}_1 \quad (1)$$
$$a\,\beta_1 + b\,\beta_2 + e = \tilde{\beta}_1 \quad (2)$$
$$a\,\gamma_1 + b\,\gamma_2 + e = \tilde{\gamma}_1 \quad (3)$$

y se encuentran c, d y f en forma similar a partir de:

$$c\,\alpha_1 + d\,\alpha_2 + f = \tilde{\alpha}_2 \quad (4)$$
$$c\,\beta_1 + d\,\beta_2 + f = \tilde{\beta}_2 \quad (5)$$
$$c\,\gamma_1 + d\,\gamma_2 + f = \tilde{\gamma}_2 \quad (6)$$

Ahora que se sabe qué es una transformación afín contractiva y cómo encontrar una que mapee una imagen fuente de una target deseada, se puede definir un Sistema de Función Iterada. Un IFS es una colección de transformaciones afines contractivas. Por ejemplo, el IFS correspondiente al Triángulo de Sierpinski es el siguiente:

$$W_1 \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$W_2 \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$W_3 \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Cada transformación debe tener también una probabilidad asociada p_i determinativa de su "importancia" relativa a las otras transformaciones. En el caso presente se deben tener p_1, p_2 y p_3. Además, la suma de las probabilidades debe dar 1.

Una forma más cómoda para expresar un IFS es la siguiente (para el ejemplo considerado):

W	a	b	c	d	e	f	p
1	0.5	0	0	0.5	0	0	0.33
2	0.5	0	0	0.5	1	0	0.33
3	0.5	0	0	0.5	0.5	0.5	0.34

ALGORITMO DE ITERACIÓN ALEATORIA

Dado un código IFS es posible su decodificación mediante el empleo del Método de Iteración Aleatoria. Recuérdese que el IFS puede contener cualquier número, m, de transformaciones afines W_1, W_2, . . . , W_m, cada una con una probabilidad asociada. Los siguientes pasos resumen el método:

1. Inicialización, x = 0; y = 0.
2. Para n = 1 a 2500, hacer los pasos 3 a 7.
3. Elegir k de modo que sea uno de los números 1, 2, ..., m; con probabilidad p_k.
4. Aplicar la transformación W_k al punto (x, y) para obtener $(\tilde{x}, \tilde{y})$
5. Poner (x, y) igual al nuevo punto $x = \tilde{x}, y = \tilde{y}$.
6. Si n> 10, dibujar (x, y).
7. Loop.

Aplicando este procedimiento a la tabla vista anteriormente, produce la figura del fractal denominada Triángulo de Sierpinski. Variando los factores de escala se puede lograr un efecto de zoom en la imagen.

CALCULO DE IFS Y APLICACIÓN DE ITERACIONES, EN BASIC

El programa adjunto a este trabajo, presenta dos opciones. La primera, calcula el código IFS a partir del target cuyas coordenadas se solicitan en la línea 20. Posteriormente el programa solicita las coordenadas de los sub-conjuntos que componen el target (Véase Teorema del Collage, más adelante) en la línea 35, calculando de este modo los códigos IFS, correspondientes a la primera transformación, segunda, . . ., m-ésima.

A continuación se ejecuta la segunda opción, la cual se puede desencadenar de dos modos: a- como continuación de la primera (lo dicho) o b- en forma directa, si se incorporan los códigos IFS en las Sentencias DATAs comprendidas entre las líneas 600 a 639.

Es esencial que las probabilidades incluidas en p() sumen en total 1. Para mayor eficiencia de las transformaciones, se listan en orden descendente de probabilidad: la transformación de probabilidad más alta va primera y la de más baja última.

El programa incluye variables para reescalar y trasladar el origen para acomodar el rango de los puntos que están siendo ploteados, a los límites de la pantalla. Si la imagen es demasiado amplia, se debe disminuir **xsc**; si los puntos están demasiado juntos, se debe aumentar. El mismo ajuste, pero en el sentido vertical, se logra con **ysc**. Para mover la imagen a izquierda, derecha, arriba o abajo se deben ajustar **xof** e **yof**.

Estos ajustes se pueden hacer por prueba y error, cambiando los factores de escala y desplazamiento.

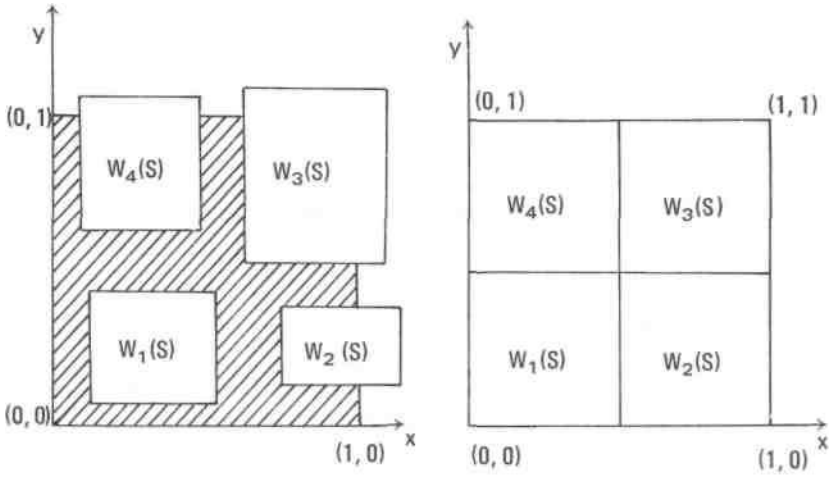
Debido a la enorme cantidad de cálculos a realizar por el método de iteración aleatoria, se recomienda el uso de un compilador. De ser posible, si el sistema de computación tiene coprocesador de punto flotante, tanto mejor.

Llama la atención que los primeros 10 puntos no son ploteados. Esto está dado por el bailoteo aleatorio del punto hasta "asentarse" sobre la imagen. Es tal como una pelota de fútbol arrojada al campo de juego poblado por jugadores expertos: hasta que alguno gane el control de la pelota, su movimiento es impredecible, o al menos independiente de las acciones de los jugadores. Pero eventualmente un jugador toma la pelota, y su movimiento se convierte en un resultado directo de la destreza de los jugadores. El hecho de que las transformaciones usadas sean contractivas, garantiza que la "pelota" eventualmente caerá en los pies de un jugador y que permanecerá bajo su control después de ello.

¿Cómo se sabe que el algoritmo de iteración aleatoria producirá la misma imagen una y otra vez, independientemente de la secuencia particular de elecciones aleatorias que hayan sido hechas? Este notable resultado fue sugerido primeramente por experimentos en matemáticas de gráficos de computadora y posteriormente, con una rigurosa fundamentación teórica, por el matemático del Georgia Institute of Technology, Jhon Elton.

EL TEOREMA DEL COLLAGE

Este teorema permite hallar un método sistemático para encontrar las transformaciones afines que producirán una codificación IFS de una imagen deseada. Para ilustrar el método, se parte de la figura de un cuadrado relleno S en el plano x-y, con sus vértices en (0,0); (1,0); (0,1); (1,1):

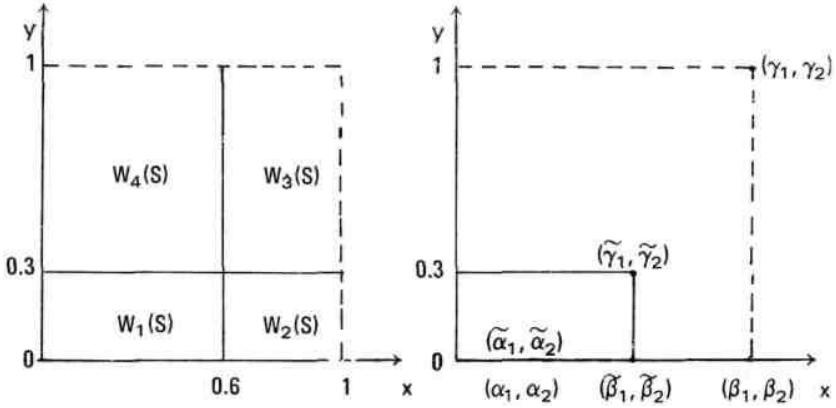


El objetivo es elegir un conjunto de transformaciones afines contractivas, en este caso W_1 , W_2, W_3 y W_4 de modo que **S** sea **aproximado tanto como sea posible por la unión de las cuatro subimágenes $W_1(S) \cup W_2(S) \cup W_3(S) \cup W_4(S)$.**

La figura anterior muestra, a la izquierda, a S junto con cuatro transformaciones afines no superpuestas. A la derecha, las transformaciones afines han sido ajustadas de tal modo que la unión de las imágenes transformadas cubra el cuadrado.

En resumen, **la imagen debe quedar constituida por la unión de transformaciones afines contractivas de sí misma.**

A modo de ejemplo de aplicación, considérese el caso anterior pero con las subimágenes que componen el cuadrado siendo rectángulos (los cuales evidentemente son transformaciones afines del cuadrado target):



Para la primera transformación, figura de la derecha, resulta:

$\alpha_1 = 0$ $\alpha_2 = 0$ $\beta_1 = 1$ $\beta_2 = 0$ $\gamma_1 = 1$ $\gamma_2 = 1$

$\tilde{\alpha}_1 = 0$ $\tilde{\alpha}_2 = 0$ $\tilde{\beta}_1 = 0.6$ $\tilde{\beta}_2 = 0$ $\tilde{\gamma}_1 = 0.6$ $\tilde{\gamma}_2 = 0.3$

Planteando el sistema de ecuaciones simultáneas (1) a (6) de la sección "Sistemas de Función Iterada" en el caso de esta transformación:

0.a + 0 . b + e = 0	(1')	0.c + 0.d + f = 0	(4')
1.a + 0.b+e =0.6	(2')	1 . c + 0.d + f = 0	(5')
1.a + 1.b+e =0.6	(3')	1.c + 1.d + f = 0.3	(6')

Resolviendo, resulta:

a = 0.6 b=0 c = 0 d = 0.3 e=0 f = 0

Obrando del mismo modo para el resto de las transformaciones, se obtiene la siguiente tabla:

W	a	b	c	d	e	f	p
1	0.6	0	0	0.3	0	0	0.18
2	0.4	0	0	0.3	0.6	0	0.12
3	0.4	0	0	0.7	0.6	0.3	0.28
4	0.6	0	0	0.7	0	0.3	0.42

Cuando el algoritmo de iteración aleatoria es aplicado a este código IFS, el cuadrado es regenerado.

El ejemplo precedente tipifica la situación general: se necesita encontrar un conjunto de transformaciones afines contractivas que causen que la imagen target sea aproximada por la unión de las transformaciones afines del mismo.

ASIGNACIÓN DE PROBABILIDADES

Una vez que se han definido las transformaciones, se necesita asignar probabilidades a ellas. Diferentes elecciones de probabilidades no llevan necesariamente a imágenes diferentes, pero sí afectan la tasa a la cual se rellenan las zonas de la imagen. Sean las transformaciones afines W_i correspondientes a una imagen I:

$$W_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ d_i \end{bmatrix}$$

donde i = 1, 2, ... m. Entonces la cantidad de tiempo que el punto "móvil" gasta aleatoriamente en la subimagen W_i es aproximadamente igual a:

$$\frac{\text{área de } W_i}{\text{área de I}}$$

Mientras la expresión ad - cb no sea cero, el cálculo standard dice que la razón es igual a la determinante de la matriz de transformación para W_i. Así una buena elección para la probabilidad p_i es:

$$\frac{a_i\, d_i - b_i\, c_i}{\sum_{k=1}^{n} |a_k\, d_k - b_k\, c_k|}$$

Considerando que ninguno de estos números p_i vaya a ser cero. Un valor nulo deberá ser reemplazado por un valor positivo muy pequeño, tal como 0.001, y las otras probabilidades correspondientemente ajustadas para mantener a la suma de todas en 1**.**

APLICACIONES

Las imágenes digitalizadas (imágenes convertidas a bits para ser procesadas por computadoras) demandan grandes cantidades de memoria para almacenarlas. Por ejemplo, una fotografía aérea puede ampliarse hasta unos

2.700 centímetros cuadrados (52 x 52 cm) lo que para una resolución de 300 x 300 pixeles, con 8 bits significativos por pixel, necesita 130 Megabytes de memoria para ser almacenada.

Usando técnicas clásicas de compresión se puede lograr, en el mejor de los casos, una relación de 10 a 1. El requerimiento en este caso sería de 13 Megabytes.

Mediante el uso de técnicas fractales, se han logrado relaciones superiores de 10.000 a 1, necesitándose para el ejemplo anterior tan sólo 13.000 bytes. En casos extremos esta relación de compresión puede superar 1.000.000 a 1.

Para trabajar prácticamente con herramientas fractales, se comienza usando técnicas de procesamiento de imagen como separación de color, detección de flanco, análisis espectral y de variación de textura; dividiendo la imagen en segmentos (técnica semejante al coloreo automático de películas de cine). Un segmento puede ser una hoja o una nube; o bien una colección más completa de pixeles: un paisaje marino, por ejemplo, que puede incluir espuma, rocas y bruma.

Luego se consultan estos segmentos en una librería de fractales. Esta librería no contiene fractales literales, que requerirían grandes cantidades de memoria. En su lugar la librería contiene conjuntos relativamente compactos de números, llamados códigos de **Sistema de Función Iterada (IFS)** que reproducirán los fractales correspondientes.

Además el sistema de catalogación de la librería es tal que imágenes que son semejantes son agrupadas juntas: códigos cercanos se corresponden con fractales cercanos.

Esto hace posible establecer procedimientos automáticos de búsqueda en librería para encontrar fractales que se aproximen a una imagen target dada.

El Teorema de Collage garantiza que siempre se puede encontrar un código IFS adecuado, amén de dar el método para hacerlo.

Una vez que se han consultado todos los segmentos en la librería y encontrado sus códigos IFS, se puede prescindir de la imagen digitalizada original y dejar los códigos, alcanzando de este modo relaciones de compresión de 10.000 a 1 o más.

Barnsley presentó en SIGGRAPH'87 una secuencia de video animada, "A Cloud Study" (estudio de una nube) donde la imagen estaba codificada a una relación superior en 1.000.000 a 1 y podía ser transmitida sobre líneas ISDN (un concepto para comunicaciones de voz y datos integrados).

Dado que las técnicas fractales de compresión de imágenes son de cómputo intensivo, es necesario trabajar con computadoras del tipo Workstation.

También es posible hacerlo con hardware a medida, que agilice el proceso de codificación y decodificación, conectado a un computador personal a través de un puerto serie. El computador personal envía códigos IFS al dispositivo, el cual responde produciendo imágenes de colores complejas

sobre un monitor. A este dispositivo se lo conoce con el nombre de IFSIS (Iterated Function System Synthesizer). Con esto, en un futuro cercano, se podrá lograr la animación de imágenes a pleno color transmitidas por línea telefónica.

Otra área para aplicación futura de la codificación IFS es el análisis automático de imágenes. Las ventajas que ofrece es que en los problemas de reconocimiento de imágenes está involucrada una búsqueda combinatoria y los tiempos de búsqueda crecen factorialmente con el tamaño de archivo de imagen. Lo excitante de todo esto es ver cómo un campo abstracto de la investigación en matemáticas lleva a nuevas técnicas con implicancias que oscilan entre lo comercial y lo industrial, sólo trabajando con computadoras personales.

Como apéndice, se acoplan al final algunas imágenes fractales como el Triángulo de Sierpinski y un cuadrado, en las figuras 1 y 2; trabajando con el programa dado en BASIC a partir de las tablas de códigos IFS indicada en este trabajo.

También en las figuras 3 y 4 se muestran un árbol fractal y un helecho, respectivamente. Los códigos IFS correspondientes son:

W	a	b	c	d	e	f	p
1	0	0	0	.5	0	0	.05
2	.1	0	0	.1	0	.2	.15
3	.42	−.42	.42	.42	0	.2	.40
4	.42	.42	−.42	.42	0	.2	.40

para árbol fractal

W	a	b	c	d	e	f	p
1	0	0	0	.16	0	0	.01
2	.2	−.26	.23	.22	0	1.6	.07
3	−.15	.28	.26	.24	0	.44	.07
4	.85	.04	−.04	.85	0	1.6	.85

para helecho.

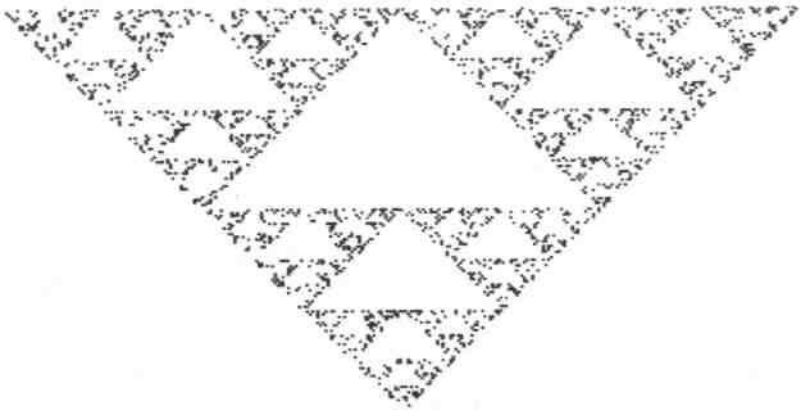


Fig. 1

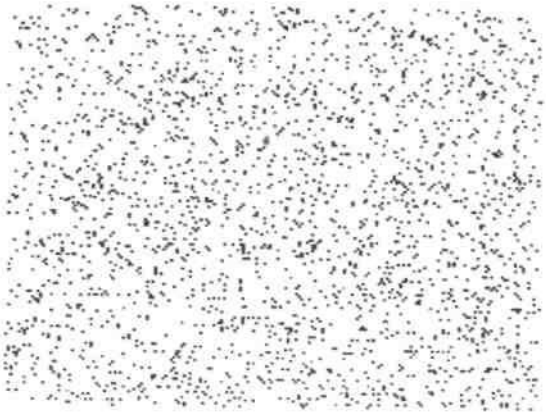


Fig. 2

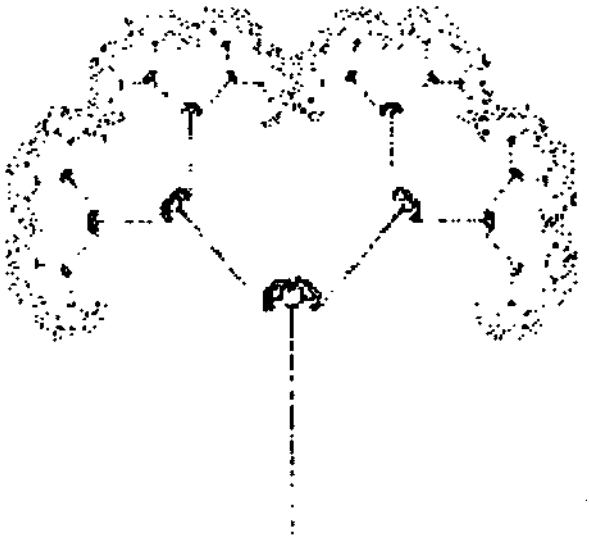


Fig. 3

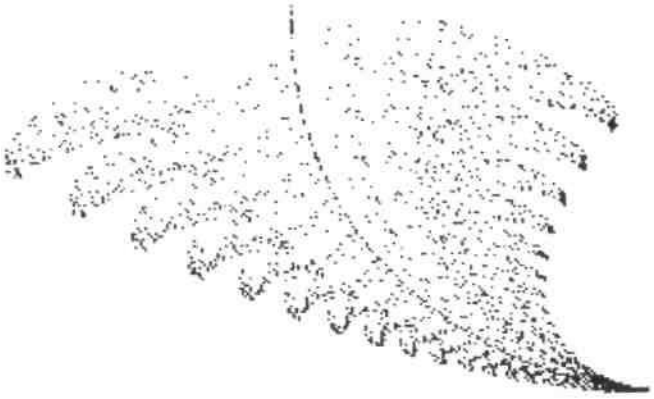


Fig. 4

BIBLIOGRAFÍA

ZORPETTE, Glenn. **Fractales: No Sólo una Foto Bonita.** Revista Telegráfica Electrónica, mayo de 1989.

ROCHA, Luis F. **Estructuras Fractales.** Revista Telegráfica Electrónica, julio de 1988.

BARNSLEY, Michael y SLOAN, Alan. **A Better Way To Compress Images.** Byte, enero de 1988.

```
  5 REM fractales
  8 INPUT "número de transformaciones:" ;m
  9 DIM a(m): DIM b(m): DIM c(m): DIM d(m): DIM e(m): DIM f(m): DIM p(m)
 10 CLS:PRINT "elija:"; PRINT "1 - PARA CALCULAR Y APLICAR CODIGO IFS":
    PRINT "2 - PARA APLICAR CODIGO IFS (completar DATAs)"
 11 INPUT w: IF w = 1 THEN GOTO 20
 12 GOTO 640
 20 INPUT "x1, y1, x2, y2, x3, y3:" ;x1, y1, x2, y2, x3, y3
 22 LET de = x1*y2 + y1*x3 + x2*y3 - y2*x3 - y3*x1 - y1*x2
 24 LET s = 0: FOR i = 1 TO m
 30 PRINT "transformación:" ;i
 35 INPUT "x1, y1, x2, y2, x3, y3:" ;t1, t2, t3, t4, t5, t6
 38 LET a(i) = (y2*t1 + y1*t5 + y3*t3 - y2*t5 - y3*t1 - y1*t3)/de
 40 LET b(i) = (x1*t3 + t1*x3 + x2*t5 - t3*x3 - t1*x2 - t5*x1)/de
 42 LET e(i) = t1 - a(i)*x1 - b(i)*y1
 46 LET c(i) = (y2*t2 + y1*t6 + y3*t4 - y2*t6 - y3*t2 - y1*t4)/de
 50 LET d(i) = (x1*t4 + t2*x3 + x2*t6 - t4*x3 - t2*x2 - t6*x1)/de
 52 LET f(i) = t2 - c(i)*x1 - d(i)*y1
 60 LET p(i) = a(i)*d(i) - c(i)*b(i): LET s = s + p(i)
 90 NEXT i
100 LET q = 0: FOR i = 1 TO m - 1
105 IF p(i) = 0 THEN p(i) = .01: LET q = q + p(i): GOTO 120
110 LET p(i) = p(i)/s: LET q = q + p(i)
120 NEXT i
130 LET p(i) = 1 - q
140 FOR i = 2 TO m
142 IF p(i) >= p(i - 1) THEN GOTO 148
144 LET L = p(i): LET p(i) = p(i - 1): LET p(i - 1) = L: goto 140
148 NEXT i
150 LET pt = 0
160 FOR j = 1 TO m
180 LET pt = pt + p(j)
190 LET p(j) = pt
200 NEXT j: CLS
240 LET xsc = 35: LET ysc = 32: LET xof = 0: LET yof = 0: LET x = 0: LET y = 0
340 FOR n = 1 TO 2500
350 LET pk = RND
355 FOR j = 1 TO m
360 IF pk <= p(j) THEN LET k = j: GOTO 370
365 NEXT j
```

```
370 LET newx = a(k)*x + b(k)*y + e(k)  LET newy = c(k)*x + d(k)*y + f(k)
410 LET x = newx  LET y = newy
460 IF n (= 10 THEN GOTO 500
480 LET x1 = x*xsc + xof  LET y1 = y*ysc + yof
490 PSET (x1, y1)
500 NEXT n  STOP
600 REM data
```