QL MAGAZINE

Suplemento especial

El conjunto de Mandelbrot

El conjunto de Mandelbrot

LISTADO 1. Fractal copo de nieve

```
100 WINDOW 512,256,0,0
110 MODE 4:PAPER 0
120 CLS
130 PENDOWN:TURNTO 0
140 LINE 40,80
150 FOR j = 0 TO 3
160   FOR i=1 TO 4
170     linea j,60
180     TURN -90
190   END FOR i
200   PAUSE
210 END FOR j
220 STOP
230 :
240 DEFine PROCedure linea (n,long)
250   LOCal long4
260   IF n = 0
270     MOVE long
280     RETurn
290   ELSE
300     long4 = long / 4
310     linea n-1,long4
320     TURN 90
330     linea n-1,long4
340     TURN -90
350     linea n-1,long4
360     TURN -90
370     linea n-1,long4
380     linea n-1,long4
390     TURN 90
400     linea n-1,long4
410     TURN 90
420     linea n-1,long4
430     TURN -90
440     linea n-1,long4
450   END IF
460 END DEFine linea
```

¿Cuánto hay de aquí al puerto más cercano? —preguntó el pescador desde su barca al caminante. Depende cuán próximo vaya a la orilla —contestó éste.

Esta extraña conversación entre un pescador y un caminante nos permite introducirnos en un tema fascinante y complejo —¿Cuál es la longitud de la costa de un país?— De la respuesta del caminante podemos deducir que depende de cuán finamente midamos. Es un hecho cierto, que al observar un mapa de una costa a gran escala vemos una distribución de cabos y golfos, asombrósamente parecida a la de un mapa visto a pequeña escala; en los dos casos aparecen entrantes y salientes de la misma escala de magnitud que la de observación. Se dica pues que la costa marítima es una línea fractal, esto es, que independientemente de cuán cerca la observemos presenta siempre el mismo aspecto.

No es único este ejemplo. En matemáticas se conocen desde hace mucho tiempo definiciones para las curvas que integran objetos más o menos extraños como son curvas a puntitos, o con picos.

Cada vez que se encontraban los matemáticos con objetos que parecían curvas, pero que no eran «curvas a simple vista» era preciso ampliar la definición del concepto de curva para incluir o eliminar estos nuevos objetos.

Cuando aparecieron curvas con picos, los matemáticos pensaron que estas curvas se portarían razonablemente bien, ¡cuán equivocados estaban! En la segunda mitad del siglo pasado empezaron a aparecer curvas que tenían picos ¡¡en todos sus puntos!! Una de las curvas más extrañas fue la presentada en 1890 por **Giuseppe Peano.** Esta curva tenía la propiedad de pasar por todos los puntos interiores y de la frontera de un cuadrado que la contenía, y si bien la curva es totalmente continua (no contiene saltos) es totalmente imposible determinar la dirección que seguirá en ninguno de sus puntos.

Posteriormente aparecieron otras curvas de este tipo, curvas a las que se dio el nombre genérico de «Curvas de Peano», de las que es corriente encontrarse ejemplos de curvas abiertas y cerradas.

En el caso de las curvas de Peano cerradas (como las gomas elásticas), se presenta otra dificultad, no se puede distinguir qué puntos del cuadrado están dentro y cuáles fuera de la superficie encerrada por la curva.

Para ver un ejemplo de una de estas fronteras exóticas sígase la siguiente receta:

Cójase un cuadrado de tamaño medio. Métase en la cacerola a fuego... (...) Un momento, creo que me estoy liando.

Cójase un cuadrado de tamaño medio. Tómese cada una de sus cuatro aristas y (sí, ahora va bien) divídase en cuatro trozos iguales. Sustituyanse los dos trozos interme-

dios por sendos cuadrados de arista la longitud de uno de los trozos que estamos eliminando (y eliminando el lado que reposa sobre la arista inicial), dejando uno de los cuadrados hacia adentro y otro hacia afuera. Obtenemos así una curva en «copo de nieve cuadrado», que es generada por uno de los programas que acompañan este artículo (para ver la relación, dependiendo de la profundidad no se borran las fases iniciales, cosa que puede remediarse con un buen CLS).

Iterando obtenemos una curva muy curiosa con las propiedades siguientes: el perímetro del cuadrado es ahora de longitud infinita (tan grande como se quiera) pues en cada iteración hemos doblado su longitud; sin embargo, aquí está lo sorprendente: la superficie encerrada por la nueva frontera es exáctamente igual a la del cuadrado original por haber añadido en cada iteración lo mismo que se quita.

Lo mismo que en el caso de curvas, el comportamiento fractal aparece en muchos otros objetos matemáticos, entre otros el llamado con-

Listado 2. Conjunto de Mandelbrot versión lenta

```
100 inicializa
110 REPeat lazo
120   rellena
130   aumenta
140 END REPeat lazo
150 :
160 DEFine PROCedure rellena
170   dx=dxi/256
180   dy=dyi/256
190   FOR i=0 TO 255
200     xx = XI + dx * i
210     yy1 = yi + 255 * dy
220     yy2 = yi + 128 * dy
230     bisecy INT (i),0,127,HUIDA (xx,yy1),HUIDA (xx,yy2)
240     yy1 = yi + 127 * dy
250     yy2 = yi
260     bisecy INT (i),128,255,HUIDA (xx,yy1),HUIDA (xx,yy2)
270   END FOR i
280 END DEFine rellena
290 :
300 DEFine PROCedure bisecy (x%,y1%,y2%,c1%,c2%)
310   LOCal xcc,yc1,yc2
320   xcc = XI + dx * x%
330   yc1 = yi + dy * (255 - INT((y1%+y2%)/2))
340   yc2 = yi + dy * (255 - INT((y1%+y2%)/2)-1)
350   IF c1% = c2% THEN
360     linea x%,y1%,x%,y2%,c1%
370   ELSE
380     IF (y2%-y1%) = 1 THEN
390       BLOCK 2,1,2*x%,y1%,color% (c1%)
400       BLOCK 2,1,2*x%,y2%,color% (c2%)
410     ELSE
420       bisecy x%,y1%,INT((y1%+y2%)/2),c1%,HUIDA(xcc,yc1)
430       bisecy x%,INT((y1%+y2%)/2)+1,y2%,HUIDA(xcc,yc2),c2%
440     END IF
450   END IF
460 END DEFine bisecy
470 :
480 DEFine PROCedure linea (xa%,ya%,xb%,yb%,c%)
490   BLOCK (2*(xb%-xa%+1)),(yb%-ya%+1),(2*xa%),ya%,color% (c%)
500 END DEFine linea
510 :
520 DEFine PROCedure inicializa
530   MODE 8:WINDOW 512,256,0,0
540   PAPER 2:CLS:CSIZE 3,1
550   INPUT '  #x?'!XI\'  #y?'!yi\'  #dx?'!dxi\'  #dy?'!dyi
560   PAPER 0:CLS
570 END DEFine inicializa
580 :
590 DEFine FuNction color% (x%)
600   IF x%=1000 THEN RETurn (0)
610   IF x% > 300 THEN RETurn (7)
620   IF x% > 100 THEN RETurn (6)
630   IF x% > 32 THEN RETurn (4)
640   RETurn (1)
650 END DEFine color%
660 :
670 DEFine PROCedure aumenta
680   x% = 0
690   y% = 0
700     box
710   END REPeat loza
720 END DEFine aumenta
730 :
740 DEFine PROCedure box
750   BLOCK dx%-1,1,x%,y%,7
760   BLOCK 2,dy%,x%+dx%-2,y%,7
770   BLOCK dx%-1,1,x%,y%+dy%-1,7
780   BLOCK 2,dy%,x%,y%,7
790 END DEFine box
800 :
810 DEFine PROCedure nuevo
820   ddx = dxi / 512
830   ddy = dyi / 256
840   XI = XI + x% * ddx
850   yi = yi + (256-y%-dy%+1)*ddy
860   dxi = dxi * dx% / 512
870   dyi = dyi * dy% / 256
880 END DEFine nuevo
```

junto de Mandelbrot, llamado así en honor de **Benoit B. Mandelbrot,** investigador de IBM en Nueva York, que ha impulsado enormemente el estudio de las figuras fractales como la mostrada antes. En el caso del conjunto de Mandelbrot, calificado por **John H. Hubbard** de la Universidad de Cornell como «El objeto más complicado de las matemáticas», no es una curva la que muestra comportamiento fractal, o por lo menos no una curva en el sentido tradicional, pues se trata de un conjunto de punto del plano complejo, y si bien su representación es un poco sosa, haremos una serie de pantallas bonitas que se podrá bien fotografiar o bien sacar a impresora con una rutina como la que se publicó en un número anterior de TODOSPECTRUM.

Como el conjunto de Mandelbrot aparece trabajando con números complejos, es conveniente repasarlos un poco, para ello hemos incluido un apéndice al final del artículo con algunos de los conceptos y fórmulas más utilizados en lo que viene a continuación.

Sea la función que toma un número complejo y nos devuelve otra dada por $f(z)=z^2+c$ donde c es una constante y z un número complejo. Si partimos inicialmente de z=0+0i y el resultado de la función lo volvemos a introducir en ella, obtendremos una sucesión de números complejos dada por:

$$c;\ c^2+c;\ (C^2+c)^2+c;...$$

Dependiendo del valor que inicialmente demos a la constante c, obtendremos una sucesión u otra. La pregunta del millón de créditos intergalácticos es: ¿Para qué valores de la constante c, la sucesión está acotada?, dicho de otra forma, se trata de encontrar los valores de c que hagan que la sucesión que genera se quede siempre razonablemente cerca del punto 0+0i. Estos puntos son los que van a pertenecer al conjunto de Mandelbrot.

Para medir cuanto de cerca está un número complejo del origen, se toma lo que se conoce como su módulo, que se corresponde con la raíz cuadrada de la suma del cuadrado de la parte real más el cuadrado de la parte imaginaria.

Por unos resultados de la teoría de funciones complejas se demuestra que si un elemento de la sucesión tiene un módulo mayor que 2, entonces a la larga los módulos de los números de la sucesión crecerán de manera ilimitada, con lo que podemos afirmar que los valores de c que en un elemento de la sucesión por ellos creados excede su módulo del valor 2 no pertenecen al conjunto que estamos intentando delimitar.

Sin embargo sólo hemos dicho que algunos valores de c no pertenecen al conjunto. Es difícil decir a priori si un valor dado de c pertenece o no al conjunto que estamos buscando pues nada nos dice si después de 500 iteraciones el módulo aún es menor que 2 que la iteración 501 no se nos vaya. Por esto se suele dar un límite máximo de iteracio-

Listado 3. Programa fuente en ensamblador

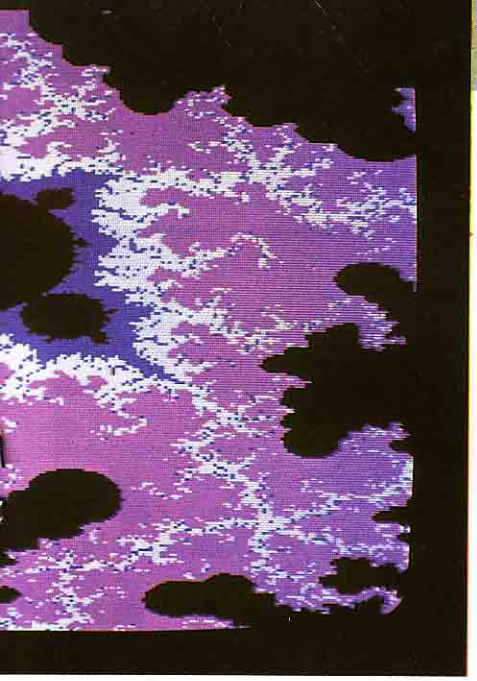
```
;-------------------------------------------------------------
;       Esta rutina assembler, lista para ser llamada por el BASIC
; permite ampliar zonas del conjunto de Mandelbrot.
;
;       Para ello, se teclea HUIDA x,y y devuelve el numero que
; hace falta para que su modulo pase de 2.
;
;       Ver Investigacion y Ciencia de Octubre de 1.985
;
;               Copyright Gerardo Izquierdo 10-85
;-------------------------------------------------------------

;-------------------------------------------------------------
;       Igualdades iniciales
;-------------------------------------------------------------

;---------- Vectores ----------
ut_scr          equ     $C8
ut_err0         equ     $CA
bp_init         equ     $110
ca_gtfp         equ     $114
bv_chrix        equ     $11A
ri_exec         equ     $11C
ri_execb        equ     $11E
;---------- Trap #2 ----------
io_open         equ     $1
io_close        equ     $2
;---------- Trap #3 ----------
sd_fill         equ     $2E
;---------- Errores ----------
err_bp          equ     -15
;---------- SuperBasic ----------
bv_rip          equ     $58
;-------------------------------------------------------------
;       Inicializar el procedimiento
;-------------------------------------------------------------
inicio  lea.l   proc_def,a1
        move    bp_init,a2
        jsr     (a2)
        rts

;-------------------------------------------------------------
;   Tabla de definicion de procedimientos
;-------------------------------------------------------------
proc_def dc.w   0               ; 0 procedimientos
         dc.w   0
         dc.w   1               ; 1 funcion
         dc.w   wandel-*        ; comienzo
         dc.b   5,'HUIDA'       ; nombre
         align
         dc.w   0,0

;-------------------------------------------------------------
;       Cargar los valores de x,y
;-------------------------------------------------------------
wandel
        move.l  %60,d1          ; Reservamos sitio para 10 numeros
        move.w  bv_chrix,a2
        jsr     (a2)
        move.w  ca_gtfp,a2      ; poner en el stack los parametros
        jsr     (a2)
        dc.b    1_x             ; x xl#xl-yl#yl xl
        dc.b    $A              ; xl#xl-yl#yl+x xl
        dc.b    s_xl            ; xl
        dc.b    1_yl            ; yl xl
        dc.b    $E              ; xl#yl
        dc.b    $16             ; xl#yl xl#yl
        dc.b    $A              ; 2#xl#yl
        dc.b    1_y             ; y 2#xl#yl
        dc.b    $A              ; 2#xl#yl+y
        dc.b    $16             ; 2#xl#yl+y 2#xl#yl+y
        dc.b    s_yl            ; 2#xl#yl+y
; En estos momentos, xl e yl tienen el nuevo valor.
        dc.b    $16             ; yl yl
        dc.b    $E              ; yl#yl
        dc.b    !_xl            ; xl yl#yl
        dc.b    $16             ; xl xl yl#yl
```



nes, pasadas las cuales sin que el módulo del elemento en cuestión sobrepase 2, declaramos «por definición» que el punto representado por c pertenece al conjunto.

Para hacer una representación del conjunto en la pantalla del QL será necesario efectuar una correspondencia de los pixels de la pantalla con los puntos del plano complejo, y colorear estos pixels según un criterio, criterio que puede ser necesario modificar.

Damos a continuación dos versiones del programa: una de ellas es para largos períodos de tiempo y la otra es para gente con prisas.

En las dos versiones se nos pide que demos las coordenadas de la esquina inferior izquierda de la pantalla y la longitud y altura del rectángulo que va a ser la pantalla sobre el plano complejo.

Las dos versiones del programa utilizan una única subrutina en código máquina encargada de efectuar el grueso de los cálculos. A esta función se le pasan como parámetros las coordenadas del punto complejo c, y devuelve un número entero entre 1 y 1000 que representa el número de iteraciones que han sido necesarias para que el módulo del elemento en cálculo de la sucesión sobrepasase el valor de 2. Realmente, para evitarse una raíz cuadrada, compara el cuadrado del módulo con 4, que nos da lo mismo.

El código de la rutina está dado tanto en asembler como en un programa cargador en SuperBASIC para aquellos que no posean un asembler, siendo en este caso necesario el teclearse el programa basic poniendo mucho cuidado, y ejecutándolo por «RUN», lleva a cabo una prueba elemental del contenido de las líneas «DATA» y se graba el resultado en el fichero «HUIDA_CDE», que será necesario cargar con:

```
a=RESPR(256):LBYTES
mdv1_huida_cde,a:CALL a
```

Antes de ejecutar cualquiera de los programas BASIC que se explican a continuación. El motivo de que la carga no vaya en los programas es que no es infrecuente el estar dando «RUN» al programa para cambiar de imagen, y cogería cada vez memoria, por lo que es más conveniente de esta forma y se tarda sólo un momento.

Esta función hace uso abundante del paquete de instrucciones de coma flotante que suministra el SuperBASIC, intentando optimizar el uso del juego de instrucciones, al punto de hacer tan solo una llamada previa, y otra llamada a la rutina de cálculos, esta vez dentro de un bucle que tiene como límite el ya señalado de 4 ó de 1.000 iteraciones.

Se podrá advertir un pequeño error, caso de que el punto pasado como parámetro tenga un módulo mayor que 2, se devuelve 1 en vez de cero. Sinceramente no he corregido este problema por no afectar en absoluto a los resultados, efectivamente las zonas del plano complejo que se exploran devuelven valores altos de la función.

```
        dc.b    $E              ; x1#x1 y1#y1
        dc.b    $A              ; x1#x1+y1#y1 = R
        dc.b    1_4             ; 4 R
        dc.b    $C              ; R-4
        dc.b    0               ; fin
;-------------------------------------------------------------------------
;
;                       zona de variables
;               +-----------------------+ <-- (A6,A4.L)
;               !           y           !
;               +-----------------------+
;               !           x           !
;               +-----------------------+
;               !           y1          !
;               +-----------------------+
;               !           x1          !
;               +-----------------------+
;               !           4           !
;               +-----------------------+
;-------------------------------------------------------------------------
        cmp.w   #2,d3           ; 4Hay dos parametros?
        bne     errr_bp         ; NO! error
        clr.l   -4(a6,a1.l)     ; ponemos 2 ceros (x1 e y1).
        clr.l   -8(a6,a1.l)
        clr.l   -12(a6,a1.l)
        move.l  a1,a4
        add.l   #12,a4
        sub.l   #18,a1          ; actualizamos el puntero de pila
        move.l  #$40000000,2(a6,a1.l)       ; ponemos 4
        move.w  #$0803,0(a6,a1.l)
        clr.l   d4              ; empezamos a contar
lazo1
        addq.w  #1,d4           ; a     adimos 1 al contador
        moveq   #0,d7           ; siempre cero
        lea.l   calc_1,a3       ; lista de calculos
        move.w  ri_execb,a2     ; orden de calcular
        jsr     (a2)            ; R-4 en 0(A6,A1.l)
        tst.l   d0
        bne.s   retorno
        add.w   #6,a1           ; restauramos pila
        tst.l   -4(a6,a1.l)     ; miramos signo de R-4
        bge.s   final           ; si >=0 terminar
        cmp.w   #1000,d4        ; #1000 iteraciones?
        blt.s   lazo1           ; NO seguir
;-------------------------------------------------------------------------
;       Devolvemos el resultado y terminamos
;-------------------------------------------------------------------------
final
        moveq   #0,d0
        move.l  a4,a1           ; tomamos el indice bueno de la pila
        subq    #2,a1           ; a     adimos 2 octetos
        move.l  a1,bv_rip(a6)   ; salvamos el nuevo valor
        move.w  d4,0(a6,a1.l)   ; cargamos ah@ el numero de iter.
        moveq   #3,d4           ; el tipo es entero
        rts                     ; terminamos
;-------------------------------------------------------------------------
;       Error y retorno
;-------------------------------------------------------------------------
errr_bp
        moveq.l #err_bp,d0
retorno rts
;-------------------------------------------------------------------------
;       Definiciones de los calculos
;-------------------------------------------------------------------------
1_x             equ     -12
1_y             equ     -6
1_x1            equ     -24
1_y1            equ     -18
s_x1            equ     -23
s_y1            equ     -17
1_4             equ     -30
calc_1  dc.b    1_x1            ; x1
        dc.b    $16             ; x1 x1
        dc.b    $16             ; x1 x1 x1
        dc.b    $E              ; x1#x1 x1
        dc.b    1_y1            ; y1 x1#x1 x1
        dc.b    $16             ; y1 y1 x1#x1 x1
        dc.b    $E              ; y1#y1 x1#x1 x1
        dc.b    $C              ; x1#x1-y1#y1 x1
```

Listado 4. Conjunto de Mandelbrot versión rápida.

```
100 inicializa
110 REPeat lazo
120   rellena
130   aumenta
140 END REPeat lazo
150 :
160 DEFine PROCedure inicializa
170   factor = 8
180   MODE 8:WINDOW 512,256,0,0
190   PAPER 2:CLS:CSIZE 3,1
200   INPUT '   #x?''XI\'   #y?''yi\'  #dx?''dxi\'  #dy?''dyi
210   PAPER 0:CLS
220 END DEFine inicializa
230 :
240 DEFine PROCedure rellena
250   CLS
260   dx = dxi * factor / 256
270   dy = dyi * factor / 256
280   x0=XI-dx
290   y=yi-dy
300   FOR j=0 TO 255 STEP factor
310     y=y+dy
320     x=x0
330     FOR i=0 TO 511 STEP 2*factor
340       x=x+dx
350       BLOCK 2*factor,factor,i,256-factor-j,color%(HUIDA(x,y))
360     END FOR i
370   END FOR j
380 END DEFine rellena
390 :
400 DEFine FuNction color% (x%)
410   IF x%=1000 THEN RETurn (0)
420   IF x% > 300 THEN RETurn (7)
430   IF x% > 100 THEN RETurn (6)
440   IF x% > 20 THEN RETurn (((x% DIV 3) MOD 4) + 2)
450   RETurn (1)
460 END DEFine color%
470 :
480 DEFine PROCedure aumenta
490   x% = 0
500   y% = 0
510   dx% = 512
520   dy% = 256
530   OVER -1
540   box
550   REPeat loza
560     tecla=CODE(INKEY$(-1))
570     box
580     SELect ON tecla
590       = 192:IF x% > 0 THEN x%=x%-2
600       = 200:IF x% + dx% < 512 THEN x%=x%+2
610       = 216:IF y% + dy% < 256 THEN y%=y%+1
620       = 208:IF y% > 0 THEN y%=y%-1
630       = 193:IF dx% > 10 THEN dx%=dx%-2
640       = 201:IF dx% < 512 THEN dx%=dx%+2
650       = 217:IF dy% < 256 THEN dy%=dy%+1
660       = 209:IF dy% > 5 THEN dy%=dy%-1
670       = 10:nuevo:OVER 0:RETurn
680       = REMAINDER
690     END SELect
700   dx% = 512
710   dy% = 256
720   OVER -1
730   box
740   REPeat loza
750     tecla=CODE(INKEY$(-1))
760     box
770     SELect ON tecla
780       = 192:IF x% > 0 THEN x%=x%-2
790       = 200:IF x% + dx% < 512 THEN x%=x%+2
800       = 216:IF y% + dy% < 256 THEN y%=y%+1
810       = 208:IF y% > 0 THEN y%=y%-1
820       = 193:IF dx% > 10 THEN dx%=dx%-2
830       = 201:IF dx% < 512 THEN dx%=dx%+2
840       = 217:IF dy% < 256 THEN dy%=dy%+1
850       = 209:IF dy% > 5 THEN dy%=dy%-1
860       = 10:nuevo:OVER 0:RETurn
870       = REMAINDER
880     END SELect
890     box
900   END REPeat loza
910 END DEFine aumenta
920 :
930 DEFine PROCedure box
940   BLOCK dx%-1,1,x%,y%,7
950   BLOCK 2,dy%,x%+dx%-2,y%,7
960   BLOCK dx%-1,1,x%,y%+dy%-1,7
970   BLOCK 2,dy%,x%,y%,7
980 END DEFine box
990 :
1000 DEFine PROCedure nuevo
1010   ddx = dxi / 512
1020   ddy = dyi / 256
1030   XI = XI + x% * ddx
1040   yi = yi + (256-y%-dy%+1)*ddy
1050   dxi = dxi * dx% / 512
1060   dyi = dyi * dy% / 256
1070 END DEFine nuevo
```

Esta función, verdadera alma de los programas, a pesar de estar escrita en assembler no es una panacea universal, pues efectuar mil veces el elevar un número complejo al cuadrado, sumarle otro, guardar el resultado, hallar el cuadrado del módulo (más rápido que hallar el módulo) y compararlo con cuatro lleva un tiempo aproximado de tres segundo y medio, valor más que respetable si tenemos en cuenta que una pantalla con muchos puntos con valor 1.000 puede llegar a tardar dos días en calcularse completa punto a punto, cosa que hace el primer programa, pero que tendremos la seguridad de tener cada punto coloreado correctamente.

La descripción de este primer programa es la siguiente, para empezar pide como ya habíamos dicho la posición del pixel situado en la esquina inferior izquierda de la pantalla y los valores del ancho y alto de la pantalla, esto sirve para dar un factor de escala (parecido al comando SCALE del SuperBASIC, pero entre otras diferencias, introduciendo también el valor del ancho x. Con estos valores y el de la variable «factor» se procede al siguiente cálculo: como es conveniente trabajar con la instrucción BLOCK y no con la POINT, pues la BLOCK permite direccionar pixels directamente y la POINT no, se calcula el número de bloques que va a haber por línea y columna, dividiendo 512 ó 256 entre 2*factor y factor respectivamente (factor se aconseja que sea potencia de 2), cuando mayor sea factor, más baja resolución obtendremos, pero antes terminará el dibujo. Una vez obtenido el tamaño del bloque mínimo a dibujar, es necesario calcular el ancho y alto de éstos, pero referido esta vez al plano complejo donde estamos trabajando, para ello dividiremos el ancho y alto de la pantalla entre el número de bloques a dibujar respectivamente en horizontal y vertical. Trabajando ahora con dos bucles FOR anidados, vamos calculando los valores de la función HUIDA en cada punto considerado (nótese que para a cada bloque se le da el valor de su esquina inferior izquierda). Una vez obtenido este valor sólo hay que pasarlo por una función que nos devuelve un número entre 0 y 7, función a la que se debe la belleza de las imágenes que acompañan el artículo. Recomandamos para esta función que devuelva 0 sólo en el caso de que se le pase 1.000 como parámetro, dejando a la voluntad de los lectores el modificarla, pues dependiendo del aumento con el que estemos mirando el conjunto habrá que colorear de una forma y otra para obtener resultados más o menos interesantes, o para resaltar la estructura filamentosa del conjunto.

Para efectuar un trabajo más rápido, pero a costa de algunos errores, está el segundo programa que trabaja de forma recursiva.

Por los manuales del SuperBASIC, sabemos que se pueden escribir programas que sean recursivos. El significado de esta palabra es bastante oscuro y aterrador (algo parecido al código máquina) y representa la posibilidad de resolver un problema teniendo en cuenta que sabemos resolver el mismo problema pero con datos ligeramente más sencillos.

El problema en nuestro caso es pintar una columna de pixels, para ello seguimos el siguiente método: Obtenemos por medio de HUIDA el valor de los pixels superior e inferior de la columna, si tienen el mismo valor, coloreamos toda la co-

```
100 RESTORE
110 a=RESPR (256)
120 FOR i=0 TO 9
130   y=0
140   FOR j=0 TO 9
150     READ x
160     POKE_W a+i*20+j*2,x
170     y=y+x
180   END FOR j
190   READ x
200   IF y <> x THEN PRINT "Error en linea - ";1000+i*10
210 END FOR i
220 SBYTES mdv1_huida_cde,a,256
1000 DATA 17402,10,13432,272,20114,20085,0,0,1,12,71328
1010 DATA 1352,21833,17473,0,0,8764,0,60,13432,282,63196
1020 DATA 20114,13432,276,20114,3139,2,26112,96,17078,-26372,73991
1030 DATA 17078,-26376,17078,-26380,10313,-9732,0,12,-27652,0,-45659
1040 DATA 18,11708,16384,0,-26622,15804,2051,-26624,17028,21060,30807
1050 DATA 32256,18426,50,13432,286,20114,19072,26148,-11524,6,118266
1060 DATA 19126,-26372,27654,3140,1000,28126,28672,8780,21833,11593,123552
1070 DATA 88,15748,-26624,30723,20085,28913,20085,-6122,5646,-4586,83956
1080 DATA 3596,-3062,-5650,3606,2810,2582,-4330,3816,5646,2786,11800
1090 DATA 3072,0,0,0,0,0,0,0,0,0,3072
```

lumna del color correspondiente, si no dividimos la columna por el centro y formamos dos subcolumnas a las cuales damos el mismo tratamiento, y así ¿hasta cuando? Hasta que tengamos una columna de sólo dos pixels de alto, en cuyo caso coloreamos cada pixel con su color y seguimos a otra cosa.

Este método tiene dos inconvenientes, el primero es que cuando estamos tratando una columna en la que los pixels superior e inferior tienen valores distintos, al dividirla por el centro, el pixel correspondiente será siempre distinto de uno de los dos extremos, por lo que siempre se fuerza una serie de subdivisiones hasta llegar al caso de una columna de dos pixels; el otro problema al que hacíamos referencia es que cada vez que subdividimos una columna, es necesario recalcular los valores extremos de la columna original, siendo necesario en los casos extremos el hacer este cálculo hasta 8 veces (¿dónde está el ahorro?).

La solución a estos problemas viene dada en el programa que presentamos, el primero de ellos se soluciona, cuando al dividir una columna en dos partes no tomamos un solo punto intermedio, sino que la subcolumna superior es la delimitada por el pixel superior y central de la original, y la subcolumna inferior es la delimitada, no por el pixel central, sino por el inmediato inferior y el inferior de la original; el segundo problema se soluciona pasando al procedimiento recursivo que calcula el color de cada columna, no sólo las posiciones de los pixels extremos, sino también sus valores, con lo que al subdividir sólo tiene que calcular los valores de los nuevos puntos y llamarse a sí mismo recursivamente.

Los dos programas tiene al final unos procedimientos que se encargan, una vez terminada la pantalla, de permitir la ampliación de una zona de esta a voluntad; se consigue esto por medio de un rectángulo que aparece en la pantalla al terminarla, este rectángulo se mueve con las teclas de cursor y se vería su tamaño con la tecla ALT junto a las de cursor (más fácil es que probéis que explicarlo); una vez que tenemos la zona que nos interesa de la pantalla recuadrada, hay que dar «ENTER» con lo que se empezará a trazar ese recuadro ampliado a toda la pantalla.

La imagen de la portada es la representación de la zona del plano complejo que tiene como esquinas inferior izquierda −0.96+0.23i y−0.86+0.33i, por lo que habrá que introducir para verla x=−0.96 y=0.23 dx0.1 dy=0.1; la función color% necesaria para verla es la dada en el programa lento.

Si queréis ver el conjunto en su totalidad, probar con x=−2.1 y=−1.5 dx=3 dy=3; con lo que aparecerá una vista «a pájaro» del conjunto. Es muy curioso comprobar que en las proximidades del conjunto observamos con la suficiente ampliación copias miniatura de él, copias que no son exactamente iguales al original, pues un sorprendente teorema nos asegura que el conjunto es conexo, esto es que cada una de las reproducciones miniatura está unida al conjunto principal por un filamento más o menos grueso.

Otras vistas nos permiten vislumbrar espirales estilizadas filamentos aparentemente caóticos y electrizados y «ojos» que nos miran curiosos desde las profundidades de las matemáticas.

APENDICE

Un número complejo se define como suma de un número real y un número imaginario, los números imaginarios son aquellos que tienen como factor el número «i» o unidad imaginaria, este número es la raíz cuadrada de −1 y por lo tanto tiene como cuadrado −1, si intentamos obtener las potencias sucesivas de i, teniendo en cuenta que $i^2 = -1$, y que por definición $i^0 = 1$ tendremos:

potencia	0	1	2	3	4	5	6	7	8
resultado	1	i	-1	-i	1	i	-1	-i	1

Como vemos es un comportamiento bastante curioso.

Gerardo Izquierdo