GRAN BIBLIOTECA

AMSTRAD

LOS GRÁFICOS

CON EL LÁPIZ EN LA PANTALLA

EDICIONES INGELEK

GRAN BIBLIOTECA AMSTRAD

18

LOS GRÁFICOS

Director editor:
Antonio M.ª Ferrer Abelló

Director de producción:
Vicente Robles

Director de la obra:
Fernando López Martínez

Redactor técnico:
Daniel Lozano Valverde
Rafael de la Ossa Villacañas

Colaboradores:
L-H Servicios Informáticos
Pilar Manzanera Amaro

Diseño:
Bravo/Lofish

Maquetación:
Carlos González Amezúa

Dibujos:
José Ochoa

Fotografía:
Grupo Gálata

ISBN del tomo: 84-7708-061-5
ISBN de la obra: 84-7708-004-6.
Fotocomposición: Andueza, S. A.
Imprime: Eurosur, S. A.
Depósito Legal: M-20972-1987
Precio en Canarias, Ceuta y Melilla: 435 ptas.

Septiembre 1987

LOS GRÁFICOS

INTRODUCCIÓN

En este volumen de la GRAN BIBLIOTECA AMSTRAD emprenderemos un viaje alucinante por el mundo de los gráficos. Una imagen vale más que mil palabras, y la capacidad de los Amstrad para la creación de imágenes es una de sus características más fascinantes.

En esta batalla a librar con la pantalla del ordenador, queda fuera de combate la serie PCW, dado que el intérprete BASIC facilitado con el aparato (Mallard BASIC), aunque de extraordinaria potencia en otros campos, como por ejemplo la gestión de ficheros, es extraordinariamente parco en lo que al control de gráficos se refiere, por no decir nulo.

Gracias a una gran cantidad de programas de ejemplo, pondremos en práctica los conocimientos teóricos expuestos en el texto. Se ha procurado en todo momento la máxima compatibilidad, aunque no siempre ha sido posible, dado que el 464 no implementa los comandos FILL y MASK; no obstante, en la mayoría de los casos, su concurrencia no es totalmente imprescindible.

Según vamos leyendo los diferentes capítulos, pasaremos del trazado de simples puntos en la pantalla, hasta la creación, traslación, rotación, ampliación y disminución de figuras tridimensionales, sin olvidar la definición de caracteres o la confección de gráficos de negocios.

Esperamos sobre todo que este libro, y los programas que incorpora, se comporten como una herramienta eficaz para nuestras evoluciones en el campo de la creación gráfica por ordenador.

COMENCEMOS POR EL PRINCIPIO

ntes de empezar a dibujar, hemos de conocer a la perfección el lugar donde vamos a realizar nuestros gráficos. Por ello, en este capítulo aprenderemos:

- Qué es la pantalla. Sus dimensiones. Sus modos.
- La utilización de los colores en los textos.
- Emplazar un texto correctamente.
- Qué son y para qué sirven los caracteres de control.

LA PANTALLA

Cualquier deporte, como sabemos, se debe jugar sobre una superficie, de unas determinadas medidas, con unas señalizaciones concretas. Pues bien, para llevar a cabo el noble deporte de los gráficos de ordenador, también precisamos un campo de juego donde podamos demostrar nuestras habilidades.

Es sabido, del mismo modo, que por ejemplo al fútbol se puede jugar sobre una superficie de hierba, tierra, barro o chuzos de punta,

es decir, existen mejores y peores canchas. Lo mismo ocurre en el amplio «campo» de los gráficos. Las pantallas de diferentes ordenadores son diferentes unas de otras. Es lo que venimos a llamar DEFINICIÓN GRÁFICA.

Afortunadamente para nosotros, si poseemos un ordenador de la serie CPC de Amstrad, disponemos de una sensacional pantalla de gráficos. Imaginemos, siguiendo el ejemplo del deporte rey, que el árbitro del encuentro ha llevado a cabo una encomiable labor «a favor» del equipo local, lo cual le ha llevado a perder por goleada, y todos los espectadores de las gradas bajan al campo a agradecerle calurosamente su actuación. Si ese césped fuera su pantalla, podrían reunirse en él hasta 128.000 aficionados.

Cada exaltado hincha representa un punto en nuestra pantalla, llamado pixel, abreviatura de la expresión inglesa *Picture Element* (Elemento de Imagen). Cuantos más puntos individuales pueda representar un micro, mejor será la definición de su pantalla de gráficos. Es por ello por lo que quedaremos plenamente convencidos de la potencia de nuestro Amstrad, al saber que tenemos a nuestra entera disposición, como ya le hemos anticipado, 128.000 pixels de definición.

Pero continuemos con la narración de los acontecimientos que están acaeciendo en el estadio. Tras repartir un par de bates de beisbol por cada espectador, se ha llevado a cabo una distribución de las personas del campo numerando una banda de 0 a 640, y un fondo de 0 a 400. Poco a poco, se irá llamando a cada espectador por su número de coordenada sobre el campo, para felicitar así personalmente al árbitro. El primero será el 0,0, espectador de la esquina inferior izquierda.

Sin embargo, si multiplicamos 640 × 400, descubriremos que obtenemos 256.000 puntos o espectadores. ¿Cómo es posible entonces que haya en la pantalla sólo 128.000 pixels? La solución es muy sencilla. Todo punto sobre la pantalla posee dos coordenadas verticales. ¿Qué significa ésto? Un ejemplo lo dice todo: Tecleemos como comando directo, es decir, sin número de línea de programa y seguido de RETURN, la orden **PLOT 0,0** y observará que un punto aparece en la esquina inferior izquierda de la pantalla. Ahora ejecutemos igualmente **PLOT 0,1** y descubriremos que no aparece en la pantalla ningún punto más.

No obstante, esto no quiere decir que esa coordenada no exista, y para demostrarlo haremos otra prueba: borremos la pantalla con **CLS** y repitamos el último experimento **(PLOT 0,1).** Como podemos ver,

el punto reaparece en la esquina inferior izquierda. Lo que ha ocurrido realmente es que el ordenador ha trazado en la pantalla el segundo punto encima del primero, pues los dos, a pesar de poseer distintas coordenadas, son para él uno solo.

El ordenador posee en la memoria toda la información necesaria para saber exactamente el estado actual de cada punto de la pantalla. Esta información se encuentra ubicada entre las direcciones de memoria 49152 y 65535, en una zona denominada RAM de vídeo. A los que desean saber cómo funciona la RAM de vídeo de su ordenador, les emplazamos al segundo apéndice de este libro, mucho más interesante de lo que a priori pueda parecer.

MODOS DE PANTALLA

Nuestro ordenador posee tres modalidades que le harán disponer de tres pantallas, cada cual de distinta extensión, definición y posibilidades.

MODO 0 - 160×200 PUNTOS - 16 COLORES
MODO 1 - 320×200 PUNTOS - 4 COLORES
MODO 2 - 640×200 PUNTOS - 2 COLORES

Como podemos observar, en todos los modos disponemos de 200 puntos verticales, y teniendo en cuenta que cada carácter de texto tiene ocho puntos de alto, existen 200/8=25 líneas de texto. Pero la longitud de cada línea no es la misma, como tampoco lo es el número de caracteres que es posible escribir en cada una de ellas, según el modo.

En modo 2, cada punto es representado por un pixel. Si cada carácter de texto posee ocho pixels de ancho, tendremos 640/8=80 caracteres por línea. Conectado el MODE 1, cada punto de la pantalla es representado por dos pixels, por lo que sólo poseemos 320 puntos en cada línea horizontal de pantalla, y cada carácter de texto, o al menos la celdilla donde está encerrado, tendrá 16 pixels de ancho. Calculando obtenemos 640/16=40 caracteres de texto por línea.

Finalmente, en modo 0, cada punto es representado por cuatro pixels dispuestos horizontalmente, es decir, tan sólo poseemos 640/4=160 puntos por línea. Cada carácter de texto tendrá ¡32 pixels de ancho! y, por tanto, obtendremos 640/32=20 caracteres de texto por cada línea.

Hemos hablado de los colores que pueden utilizarse en cada modo

de pantalla. Pasemos, pues, a manejar con exactitud esta delicada parte de la programación de nuestro ordenador, que por otra parte no ha sido tratada de forma excesivamente amplia en su manual.

PINTE SU AMSTRAD COLOR DE...

El color es importante, tanto desde el punto de vista de la espectacularidad, como del de la presentación o la vistosidad. Son veintisiete los colores de los que dispone nuestro Amstrad, pero como mucho, sólo podremos emplear dieciséis simultáneamente. Esto es debido a que el ordenador pinta, «colorea» con un número determinado de lápices de colores, denominados plumas (en inglés PEN), cuyo número difiere, como hemos visto, de un modo a otro.

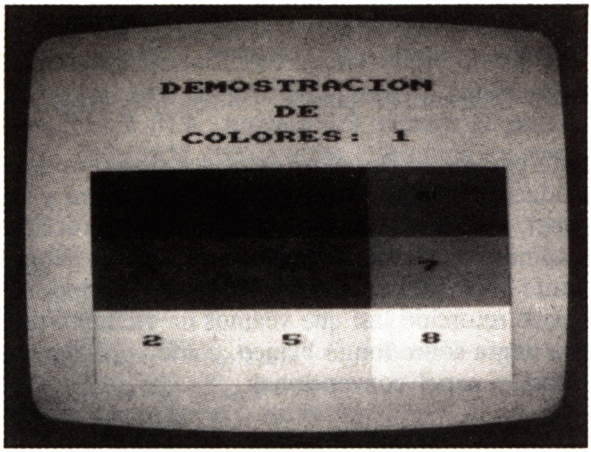
En todos los modos poseemos dieciséis plumas, pero en el modo 2 sólo podremos rellenarlas con dos colores distintos, en el 1 con cuatro y, finalmente, en modo 0, cada pluma podrá rellenarse de 16 colores distintos, a elegir de entre los veintisiete. El programa COLOR ilustra en la pantalla, en dos páginas, todos los colores, del 0 al 26.

COLOR

```
100 MODE 0:INK 0,13:BORDER 13:INK 10,0:PEN 10:PAPER 0:CLS
110 LOCATE 5,1:PRINT"DEMOSTRACION"
120 LOCATE 10,3:PRINT"DE"
130 LOCATE 6,5:PRINT"COLORES:"
140 FOR I=0 TO 2:LOCATE 14,5:PRINT I+1:COLOR=1
150 FOR J=0 TO 8:INK J+1,J+9*I:NEXT J
160 FOR X=2 TO 14 STEP 6
170 FOR Y=8 TO 20 STEP 6
180 WINDOW #1,X,X+5,Y,Y+5
190 PAPER #1,COLOR:PEN #1,10:CLS #1
200 LOCATE #1,2,3:PRINT #1,COLOR+I*9-1
210 COLOR=COLOR+1:NEXT Y:NEXT X
220 IF INKEY$="" THEN GOTO 220
230 NEXT I:GOTO 140
```

Si estamos utilizando un monitor monocromo, tan sólo apreciaremos una gran variedad de tonos grisáceos. Aun así, no nos preocupemos, pues todavía podremos sacar mucho partido de este libro, y la gran mayoría de los programas que lo acompañan nos serán de gran utilidad. El programa base del libro, GRAPHICS CREATOR, no utiliza colores, ni comandos no implementados para el CPC 464, siendo compatible para todos los ordenadores Amstrad de la serie CPC.

La instrucción PEN nos permite elegir la pluma rellenada con el color que deseamos emplear. Cuando el ordenador se conecta, exis-



ten dieciséis plumas rellenadas con los colores por defecto. La tabla adjunta, nos servirá para conocer estas condiciones de partida.

Número de tinta	Color	Número de tinta	Color	Número de tinta	Color
0	Negro	9	Verde	18	Verde intenso
1	Azul	10	Cyan	19	Verde mar
2	Azul intenso	11	Azul celeste	20	Cyan intenso
3	Rojo	12	Amarillo	21	Verde lima
4	Magenta	13	Blanco	22	Verde pastel
5	Malva	14	Azul pastel	23	Cyan pastel
6	Rojo intenso	15	Anaranjado	24	Amarillo intenso
7	Morado	16	Rosado	25	Amarillo pastel
8	Magenta intenso	17	Magenta pastel	26	Blanco intenso

Introduzcamos el modo 0 de pantalla. En la tabla de colores por defecto, es decir, los colores introducidos en las plumas cuando conectamos el ordenador, podemos comprobar que, por ejemplo, la pluma 12 está rellena de tinta 22 (verde brillante) en este modo. Si tecleamos PEN 12, la tinta de todo aquello escrito a partir de ese momento se tornará verde brillante.

En modo 1 la pluma 12 posee, por defecto, el color 1 (negro). Tecleando MODE 1:PEN 12 la tinta se volverá negra. En modalidad cero sólo existen dos colores, donde inicialmente las plumas impares estarán rellenas de tinta amarilla (código 24), y las pares (pluma del papel) de tinta 1 (azul).

Si deseáramos cambiar el color de alguna de ellas, simplemente

habríamos de emplear la instrucción INK, indicando el número de pluma a alterar y el color que deseamos aplicar.

INK <número de pluma>, <color>[,<color>]

El segundo parámetro de color (entre corchetes) es opcional, y su inclusión produciría un flash entre los colores primero y segundo. El fondo, el papel, se cambia de color mediante la instrucción PAPER, seguida del número de pluma. Naturalmente, ésta ya habrá sido rellenada, mediante INK, del color deseado para el papel. Quizás nos parezca un poco retorcido, así que veamos un ejemplo: deseamos escribir en tinta negra sobre fondo blanco, como normalmente solemos hacer con lápiz y papel convencional:

INK 0,0:INK 1,13:PEN 0:PAPER 1

Esta sentencia es válida para todos los modos de pantalla, pues sólo hace uso de un máximo de dos colores y, derivado de ello, un máximo de dos plumas.

Para cambiar el color del marco de la pantalla, parte de nuestro monitor o receptor de televisión que es totalmente independiente del resto de la pantalla, utilizamos el comando BORDER, seguido del número del color elegido para él. Es posible también hacer parpadear el marco introduciendo dos colores separados por comas. Así, por ejemplo, podemos conseguir fácilmente un borde al estilo «La moda de España»:

BORDER 3,24

¡Así de fácil! Es posible realizar la misma operación con el color de tinta de las plumas. Tecleemos:

INK 1,3,24

y a continuación:

PEN 1

o del fondo

PAPER 1

y la tinta en el primer caso, y el papel en el segundo, se tornará parpadeante entre los colores rojo y amarillo brillante. Sólo nos que-

da por ver dos instrucciones más, GRAPHICS PEN y GRAPHICS PAPER, idénticas a las anteriores, pero referidas a los gráficos en alta resolución, los trazados con puntos y líneas. Cuando tecleamos el comando GRAPHICS PEN, seguido del número de pluma, toda línea dibujada será coloreada con la tinta contenida en esa pluma.

D'FLASH

```
100 MODE 0:CLS
110 LOCATE 2,13: PRINT"PARPADEO DEL BORDE"
120 SPEED INK 30,30:BORDER 0,3
130 IF INKEY$="" THEN GOTO 130
140 LOCATE 15,13:PRINT "PAPEL"
150 BORDER 1:SPEED INK 5,10:INK 0,4,9
160 IF INKEY$="" THEN GOTO 160
170 LOCATE 11,13:PRINT "DE LETRA "
180 INK 0,1:SPEED INK 10,20:INK 1,10,15
190 IF INKEY$="" THEN GOTO 190
200 LOCATE 14,13:PRINT"TODO  "
210 SPEED INK 5,5:BORDER 0,3:INK 0,4,9
220 IF INKEY$="" THEN GOTO 220
230 BORDER 1:INK 0,1:INK 1,24
```

El Emplazamiento De Un Texto

Tenemos tres pantallas de texto, según el modo que elijamos para imprimir nuestros mensajes. Cada carácter en la pantalla tiene una coordenada X (horizontal) y una Y (vertical), estando el origen de este sistema de coordenadas situado en el vértice superior izquierdo, cuyas coordenadas X,Y son (1,1) y no (0,0).

D'LOCATE

```
100 MODE 2:CLS
110 INPUT "Coordenada horizontal (1,80)";x
120 INPUT "Coordenada vertical (1,25)";y:CLS
130 FOR i=1 TO x-2:LOCATE i,y:PRINT CHR$(154)
140 IF i=x-2 THEN LOCATE i,y:PRINT CHR$(243)
150 NEXT i
160 FOR i=1 TO y-1:LOCATE x,i:PRINT CHR$(149)
170 IF i=y-1 THEN LOCATE x,i:PRINT CHR$(241)
180 NEXT i
190 LOCATE x,y:PRINT CHR$(233):SOUND 1,200
200 LOCATE x+1,y+1:PRINT "(";x;",";y;")"
210 GOTO 210
```

En la pantalla de texto no hablaremos de tantos pixels de alto por tantos de largo, sino de caracteres de alto por ancho. Cada carácter es una retícula de ocho puntos de ancho por ocho de alto, por lo cual:

MODE 0 - 160/8 = 20 CARACTERES DE ANCHO * 200/8 = 25 DE ALTO

MODE 1 - 320/8 = 40 CARACTERES DE ANCHO * 200/8 = 25 DE ALTO
MODE 2 - 640/8 = 80 CARACTERES DE ANCHO * 200/8 = 25 DE ALTO

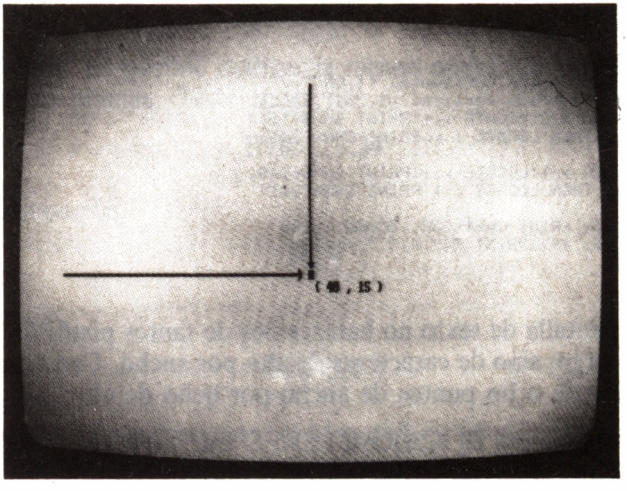
Es decir, en todos los modos existen 25 líneas de arriba a abajo. Para situar un texto en la pantalla, se utiliza la palabra clave PRINT, sin duda la más popular de las órdenes BASIC. Si tecleamos:

PRINT "¿QUE HAY DE NUEVO, VIEJO?"

el ordenador se limitará a escribir el mensaje en la primera línea vacía que encuentre. Pero, ¿qué tal si lo ubicáramos en la línea 16, a partir de la columna 5? Para ello, disponemos de la palabra BASIC LOCATE, que sitúa el cursor de textos en una posición determinada. Después teclearemos la orden PRINT y entre comillas el mensaje que deseamos escribir en la pantalla:

LOCATE 5,16: PRINT "HZGXJF... DE TODOS LOS ESPAÑOLES"

Sin embargo, aún existe otra manera de presentar en la pantalla un mensaje utilizando las coordenadas de la pantalla de gráficos, es decir 640×400. Imaginemos que deseamos colocar el número 1 en la esquina superior derecha, pero no en la retícula que corresponde a ese lugar, sino un par de pixels hacia la izquierda, y otro par hacia abajo.



Esto se consigue gracias a la orden TAG. Observemos el siguiente programa:

```
10 CLS
20 TAG
30 MOVE 630,398
40 PRINT "1"
50 TAGOFF
```

Efectivamente, el número 1 ha sido representado en la posición (630,398) de la pantalla, que son las coordenadas del punto superior izquierdo de la retícula del carácter en esta posición. La orden TAG consigue fundir la pantalla de textos con la de gráficos, MOVE desplaza el cursor de texto a la posición requerida, PRINT imprime el mensaje, y TAGOFF permite que de nuevo los mensajes puedan ser impresos mediante LOCATE. Su ausencia provocaría que todo mensaje fuera a parar a la posición del cursor de gráficos.

Con imaginación se puede emplear la orden TAG para diversas aplicaciones. Probemos con este programa que simula un scroll a lo largo de la pantalla de un mensaje, gracias a TAG.

```
10 CLS:TAG
20 FOR N=-8 TO 410 STEP 2
30 MOVE 0,N
40 PRINT"SUAVE SCROLL CON LA ORDEN TAG";
50 NEXT:TAGOFF
```

No olvidemos poner el punto y coma detrás de las comillas de PRINT, pues anulan la aparición de los gráficos de avance y retorno de carro. Probemos con él y sin él.

CARACTERES ESPECIALES: SU UTILIDAD

Estos son caracteres adicionales para el manejo de la pantalla. Se trata de los comandos CHR$, gracias a los cuales se puede acceder a unas determinadas subrutinas del sistema operativo de gran utilidad. Para llevar a cabo estas acciones necesitamos la siguiente orden:

```
PRINT CHR$ (N)
```

donde N es el número de código del carácter especial. PRINT tiene

en esta ocasión la misión de hacer llegar el carácter N, denominado código de control, a la rutina del sistema operativo correcta.

El primer carácter especial, cuyo nombre de pila es SOH, tan sólo permite visualizar los gráficos de otros caracteres de control. Normalmente, aquellos caracteres ASCII cuyos códigos sean menores de 32 no pueden ser visualizados y son enviados al sistema operativo como caracteres de control. Gracias a SOH podemos representarlos, tal como ocurre al escribirnos en modo gráfico (con TAG); recordemos el efecto que tenía no incluir un punto y coma al escribir un mensaje en modo gráfico. Pongamos en acción SOH tecleando:

```
PRINT CHR$(1) CHR$(N)
```

donde N será el número de código del carácter de control que se desea escribir en la pantalla.

PRINT CHR$(2) tiene por función deshabilitar la presentación del cursor en el transcurso de un programa, mientras CHR$(3) lo activará nuevamente. Comprobarlo es fácil: tecleemos el siguiente programa con la línea 10 y sin ella, ejecutándolo en ambos casos para apreciar su efecto.

```
10 PRINT CHR$(2)
20 INPUT"FIJEMONOS EN EL CURSOR";I$
```

Cuando el programa termine de ejecutarse, el cursor volverá a conectarse automáticamente, aunque si hubiera seguido funcionando, habríamos utilizado para ello la orden PRINT CHR$(3).

El código cuatro equivale a la orden BASIC MODE, debido a lo cual debe ir seguido de un parámetro que indique el modo de pantalla a establecer:

```
PRINT CHR$(4) CHR$(N)
```

donde N es un número de 0 a 2, según sea el modo de pantalla que elijamos. ENQ, el código de control número 5, no tiene una clara utilidad, pues tan sólo imprime en el cursor de gráficos el carácter ASCII que le ordenemos, de forma similar a TAG.

CHR$(6) se halla íntimamente relacionado con CHR$(21), cuyo efecto comentaremos más adelante. PRINT CHR$(7) no ejerce ninguna acción sobre la pantalla, aunque sí es claramente distinguible; lo mejor es que lo probemos (y no se nos olvide hacerlo con el volumen del altavoz en condiciones de ser escuchado).

Los códigos de control del 8 al 11 se encargan de desplazar el cursor por la pantalla. CHR$(8) lo mueve un carácter hacia la izquierda, CHR$(9) a la derecha, CHR$(10) será igual que apretar la tecla de la flecha hacia abajo, y finalmente CHR$(11) lo desplazará hacia arriba, como con la flecha en este sentido.

PRINT CHR$(12) lleva a cabo la misma función que el comando CLS. El número 13, llamado de retorno de carro, tan sólo sitúa el cursor al principio de la línea en que se encuentra. 14 y 15 equivalen a las órdenes BASIC PAPER y PEN respectivamente y CHR$(16) equivale a CLR.

CHR$(17) borra todos los caracteres desde el principio de una línea hasta la posición del cursor; función inversa a la de CHR$(18), que borra desde la posición del cursor hasta el final. Algo parecido ocurre con los caracteres 19 y 20. CHR$(19) borra todos los caracteres desde el principio de la pantalla hasta la posición del cursor, mientras CHR$(20) lo hace desde ésta hasta el final.

CHR$(21) tiene una función interesantísima, dado que desconecta aparentemente la pantalla de textos y, a pesar de que el ordenador recoge toda la información que nosotros le mandamos mediante el teclado, ésta no aparece representada en la pantalla. Observemos el siguiente programa:

```
10 PRINT CHR$(21)
20 INPUT "INTRODUZCA LA CLAVE DE ENTRADA";N$
30 IF N$="BATIBURRO" GOTO 40 ELSE NEW
40 PRINT CHR$(6);"SI ERES TU, NO TE HABIA RECONO-
   CIDO!"
```

Con lo cual se evita que miradas maliciosas intenten averiguar nuestra clave. CHR$(6) activa de nuevo la pantalla. De no introducir como clave la palabra BATIBURRO, comprobemos lo que ocurre.

PRINT CHR$(22) activa el modo transparente, gracias a lo cual podemos superponer dos gráficos, consiguiendo vocales acentuadas o subrayado de texto, por ejemplo.

El número 23 es algo más complicado y extenso. Su función es la de activar las diferentes modalidades del lápiz gráfico en la alta resolución. Puede tener cuatro valores, de 0 a 3. Pero ya hablaremos de él en el próximo capítulo, cuando sepamos utilizar todos los comandos BASIC de alta resolución de gráficos.

El vigésimo cuarto invierte el color de PAPER y PEN, es decir, imprime en «negativo». CHR$(25) equivale al comando SYMBOL,

CHR$(26) a WINDOW. Ambos comandos los utilizaremos más adelante. El 28 equivale a INK y el 29 a BORDER.

El penúltimo carácter sirve para posicionar el cursor en el vértice superior izquierdo, se trata de CHR$(30). Finalmente, CHR$(31) equivale al comando LOCATE.

La utilización de los caracteres de control abre aún más las fronteras de la programación en nuestro CPC, debido a lo cual se utilizan en algunos de los programas de este libro, a fin de que logremos familiarizarnos con ellos y lleguemos a comprender sus amplias posibilidades. Para finalizar este capítulo, presentamos una tabla resumen de las características de todos los códigos de control.

Aquellos en los cuales la columna de parámetros indique un valor superior a 0, habrán de ir seguidos de tantos caracteres como indique dicho dato, para completar el significado del código. Así, la forma general de ejecución de cualquiera de ellos es:

PRINT CHR$(código);[CHR$(Parám.1);]...[CHR$(Parám.N);]

Cód.	Núm. de parám.	Descripción
0	0	Sin efecto.
1	1	Escribe el carácter indicado como parámetro, aunque éste sea un código de control (0 a 31).
2	0	Deshabilita la presentación del cursor.
3	0	Habilita la presentación del cursor.
4	1	Pasa la pantalla al modo indicado por el parámetro.
5	1	Escribe el carácter indicado como parámetro utilizando el GRAPHIC VDU.
6	0	Habilita el TEXT VDU.
7	0	Emite un pitido.
8	0	Una vez que hace válida la posición del cursor, retrocede un carácter.
9	0	Una vez que hace válida la posición del cursor, avanza un carácter.
10	0	Una vez que hace válida la posición del cursor, avanza una línea.
11	0	Una vez que hace válida la posición del cursor, retrocede una línea.
12	0	Borra la ventana actual y desplaza el cursor a su posición de origen (esquina superior izquierda de la ventana).

Cód.	Núm. de parám.	Descripción
13	0	Una vez que hace válida la posición del cursor, lo desplaza a la primera columna de la misma línea de la ventana donde se encuentra.
14	1	Establece el color de tinta para fondo al valor proporcionado por el parámetro.
15	1	Establece el color de tinta para primer término al valor proporcionado por el parámetro.
16	0	Una vez que hace válida la posición del cursor, la borra con la tinta de fondo actual.
17	0	Una vez que hace válida la posición del cursor, borra con la tinta de fondo actual las comprendidas entre ésta y la primera columna de la misma línea de la ventana.
18	0	Una vez que hace válida la posición del cursor, borra con la tinta de fondo actual las comprendidas entre ésta y la última columna de la misma línea de la ventana.
19	0	Una vez que hace válida la posición del cursor, borra con la tinta de fondo actual las comprendidas entre ésta y la esquina superior izquierda de la ventana.
20	0	Una vez que hace válida la posición del cursor, borra con la tinta de fondo actual las comprendidas entre ésta y la esquina inferior derecha de la ventana.
21	0	Deshabilita el TEXT VDU.
22	1	Establece el modo de escritura según el parámetro (1=Opaco; 2=Transparente).
23	1	Establece el modo de escritura de GRAPHIC VDU según el parámetro (1=Opaco; 2=XOR; 3=AND; 4=OR).
24	0	Intercambia los colores actuales de tinta para fondo y primer término.
25	9	Establece la forma de carácter definido. El primer parámetro corresponde al código de carácter a asignar y los ocho restantes a los bytes que configuran su matriz, empezando por la primera línea.
26	4	Establece los límites de la ventana de texto. Los dos primeros parámetros corresponden a las columnas izquierda y derecha (se selecciona el valor menor para la izquierda) y los dos siguientes a las filas (se selecciona el valor menor para la superior).
27	0	Sin efecto especial.

Cód.	Núm. de parám.	Descripción
28	3	Establece los colores a emplear en una tinta. El primer parámetro es el número de tinta y los dos siguientes los colores (si son iguales el color no es intermitente).
29	2	Establece los colores para el marco de la pantalla según sus dos parámetros (si son iguales no se produce intermitencia).
30	0	Desplaza la posición actual del cursor a origen de la ventana (esquina superior izquierda).
31	2	Desplaza el cursor a las coordenadas de la ventana proporcionadas por los parámetros, en la forma columna y fila, respectivamente, teniendo en cuenta que la posición de origen de la ventana se considera la 1,1.

PUNTOS, LÍNEAS Y FORMAS

hora que ya conocemos las características del campo de juego, no será difícil empezar a hilvanar las primeras jugadas. Cuando termine de leer este capítulo, teclear y experimentar con todos sus programas, sabremos:

- Dibujar puntos y líneas por todas partes de la pantalla.
- Cuál es la función del carácter de control CHR$(23).
- Qué es utilizar el modo de tinta XOR, AND y OR.
- Para qué sirven los comandos TEST, MASK y FILL.
- Dibujar circunferencias, elipses y espirales.
- Cómo representar funciones matemáticas.

DIBUJANDO LÍNEAS

Como sin duda recordaremos, la pantalla constituye un sistema de coordenadas de 640*400 puntos en todos los modos. Imaginemos que

deseamos trazar una línea desde la posición X1,Y1 hasta la X2,Y2. Para hacer esto en nuestro CPC, sólo tendremos que teclear:

```
10 MOVE X1,Y1
20 DRAW X2,Y2
```

MOVE tiene por función trasladar el cursor de gráficos hasta la posición especificada X1,Y1 y DRAW dibuja una línea desde allí hasta las coordenadas señaladas X2,Y2. Quizás deseemos que en vez de una línea amarilla, color por defecto al conectar el ordenador, sea de color rojo brillante, quedando así el programa:

```
10 INK 3,6
20 MOVE X1,Y1,3
30 DRAW X2,Y2
```

Así, pues, añadiendo el número de pluma elegida como tercera coordenada de la orden MOVE, la línea será dibujada en el color 6, rojo brillante. Análogamente, podríamos haber añadido el número de pluma a la orden DRAW. En primer lugar, hemos introducido tal color en la pluma tres, la cual hemos utilizado en MOVE.

El origen del sistema de coordenadas de la pantalla es (0,0). BASIC permite cambiar el origen a cualquier otro punto de la pantalla, incluso fuera de ella, mediante:

```
ORIGIN X1,Y1
```

siendo X1 e Y1 las coordenadas del nuevo origen. Si hemos situado éste en la posición (100,100) de la pantalla y deseáramos trazar una línea desde allí hasta el vértice inferior izquierdo, primitivo origen, deberíamos teclear coordenadas negativas:

```
ORIGIN 100,100
DRAW - 100,100
```

Así, pues, podemos entender que el origen de la pantalla ha sido ubicado inicialmente en la esquina inferior izquierda, para evitar las coordenadas negativas.

El Interesante Carácter De Control CHR$(23)

En el capítulo anterior dejamos en la sala de espera este carácter

de control, pues aún no conocíamos el uso del comando DRAW. Cuando normalmente indicamos al ordenador que dibuje una línea, éste activa (o enciende) una serie determinada de puntos, sin tener en cuenta su estado anterior. Si su línea pasa por encima de otra, la actual borrará parte de la anterior.

Si disponemos de una pantalla con varias líneas e introducimos el comando PRINT CHR$(23) CHR$(1), cuando ordenemos dibujar la siguiente línea, el ordenador llevará a cabo una relación lógica XOR entre los nuevos puntos y los primitivos. El resultado es 1 sólo si ambos puntos son diferentes, y únicamente se encenderá cada punto si el estado primitivo y el nuevo son diferentes.

Si tras dibujar una línea, trazamos otra igual y en la misma posición en modalidad XOR (es decir, con PRINT CHR$(23) CHR$(1) delante), el resultado será que la primera se borrará, pues los puntos primitivos y los nuevos son iguales. Probémoslo:

```
10 PRINT CHR$(23) CHR$(1)
20 MOVE 100,100:DRAW 200,200
30 WHILE INKEY$="":WEND
40 MOVE 100,100:DRAW 200,200
```

La modalidad AND, que se activa tecleando PRINT CHR$(23) CHR$(2), hace que el ordenador tan sólo encienda un punto si ya había uno en aquella posición anteriormente (operación lógica AND). Por último, la modalidad transparente, OR, que se activa mediante PRINT CHR$(23) CHR$(3), tiene por función posicionar puntos nuevos sobre puntos viejos. Si el punto nuevo tiene el color cero (negro), el antiguo no se borra.

Algo Más Sobre Líneas

Cuando dibujamos una línea por el método MOVE y DRAW, el cursor gráfico se sitúa en primer lugar en las coordenadas (X1,Y1) de MOVE y, posteriormente, en las (X2,Y2) de DRAW, cuando la línea haya sido dibujada. Es decir, la siguiente línea tendrá como coordenadas de inicio (X2,Y2), y bastará con un DRAW (X3,Y3) para dibujarla, sin necesidad de especificar anteriormente el MOVE (X2,Y2).

Pero existe otra modalidad de dibujo de trazos que el manual llama de posiciones relativas, pero que nosotros llamaremos de incre-

mentos. Para utilizar este modo, se añade a las órdenes BASIC la letra R *(relative):* MOVER, DRAWR.

Si tenemos el cursor gráfico en la posición 100,100 y deseamos desplazarlo hasta la posición 120,120, podemos llevar la operación a cabo por estos dos caminos

MOVE 120,120

o bien

MOVER 20,20

es decir, un incremento de las coordenadas de MOVE de 20 puntos hacia la derecha y 20 hacia arriba. Para colocar el cursor en la posición (80,80), teclearemos MOVER − 20, − 20. Si el cursor está ubicado en 100,100 y deseamos dibujar una línea desde allí hasta la posición 145,130, bastará con:

DRAWR 45,30

NADIE ES PERFECTO

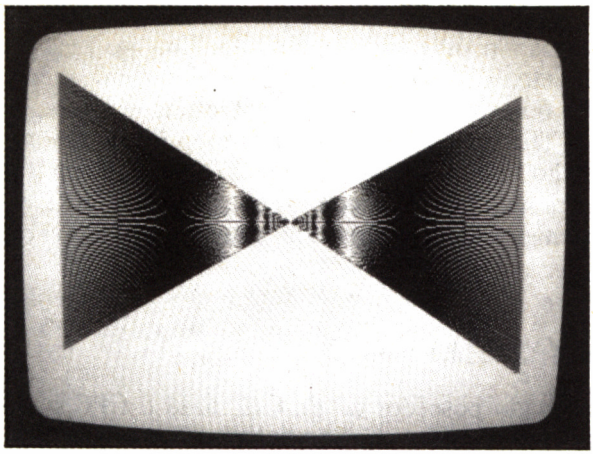
Dibujando en la pantalla simples series de líneas creadas mediante bucles, se pueden realizar gráficos de gran espectacularidad. Este es el caso del programa que listamos a continuación. Sin embargo, estos gráficos sólo son causa de las imperfecciones de la pantalla de nuestro CPC. Los programadores, por supuesto, han logrado sacar de un defecto efectos espectaculares. Comprobemos su acción en todos los modos.

```
D'EF

10 CLS
20 a=INT (RND*27)+2:INK 3,a
30 x=400
40 FOR y=0 TO 400 STEP 4
50 MOVE 0,x,3,1:DRAW 639,y
60 x=x-4
70 NEXT y
80 GOTO 20
```

DIBUJANDO PUNTOS

Por el momento, sólo hemos hablado de las líneas, sin embargo, en un momento dado, quizás necesitemos encender un sólo punto de



la pantalla, y no una serie de ellos. Esto no se consigue dibujando una línea desde unas coordenadas hasta esas mismas, sino con la palabra clave PLOT.

PLOT tiene la misma función que MOVE, aunque además de trasladar la posición del cursor de gráficos, dibuja un punto en esa posición. Así:

PLOT 100,100

dibujará un punto en la posición (100,100), además de trasladar el cursor gráfico hasta allí. Si ahora tecleamos:

PLOTR 20,20

aparecerá otro punto en la posición (120,120), incremento de 20 puntos hacia la derecha y 20 hacia arriba de la posición anterior (100,100) del cursor.

La función TEST (X,Y) nos indica, si el valor retornado es distinto de cero, que el punto situado en la posición (X,Y) está encendido.

```
D'TEST

10 CLS
20 PLOT 500,200
30 FOR n=0 TO 640 STEP 2
40 IF TEST (n,200)<>0 THEN END ELSE PLOT n,200
50 NEXT n
60 GOTO 40
```

Tenemos un punto dibujado en la coordenada (500,200). Una línea va avanzando dibujando puntos mientras no encuentre ningún pixel encendido. TEST comprueba que en las coordenadas donde traza un punto no existe otro encendido. Llegada esa hora, el programa parará. El STEP 2 es debido a que en modo 1 cada unidad posee dos pixel de ancho. Esto es, un punto situado en la posición (0,200) ocupa también la posición (1,200), con lo cual, al hacer TEST en esta posición el ordenador ya encontraría un pixel encendido, el que acabaría de escribir.

RAYAS Y RELLENADOS

El afortunado poseedor de un CPC 6128 o CPC 664 dispone de algunos comandos gráficos que no se hallan implementados en el CPC 464. Pasémosle rápida revista.

Uno de estos comandos es MASK, que permite dibujar rayas o líneas discontinuas en la pantalla. Se utiliza simplemente de la siguiente forma:

MASK número

donde el número se halla en el rango 0 a 255. Porque MASK dibuja líneas en grupos de ocho en ocho pixels, estando unos encendidos y otros apagados, según la forma binaria del número que introduzcamos. Por ejemplo:

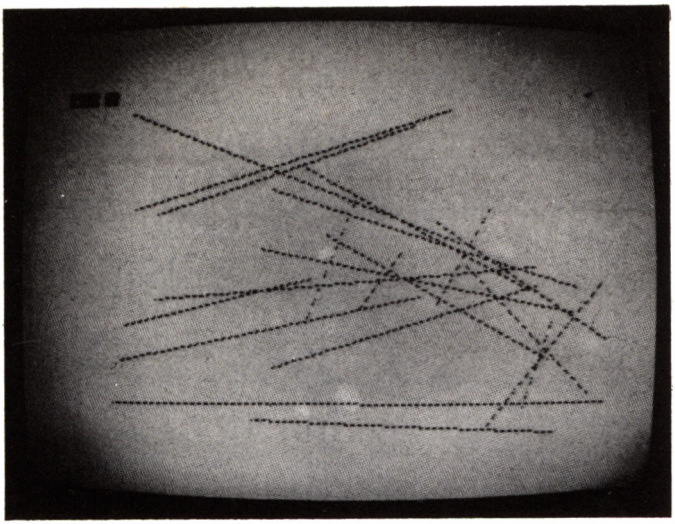
```
MASK 170
DRAW 640,0
```

dibujaría una línea por toda la banda inferior de la pantalla con un pixel encendido (1) y el siguiente apagado (0), pues 170 es el número binario

1 0 1 0 1 0 1 0

D'MASK

```
10 MODE 2:CLS
20 INPUT "Tipo de trazado (0,255)";tipo
30 CLS:MASK tipo
40 LOCATE 1,1:PRINT"MASK";tipo
50 FOR i=1 TO 20
60 MOVE RND*640,RND*400
70 DRAW RND*640,RND*400
80 NEXT i
90 WHILE INKEY$="":WEND
100 GOTO 10
```

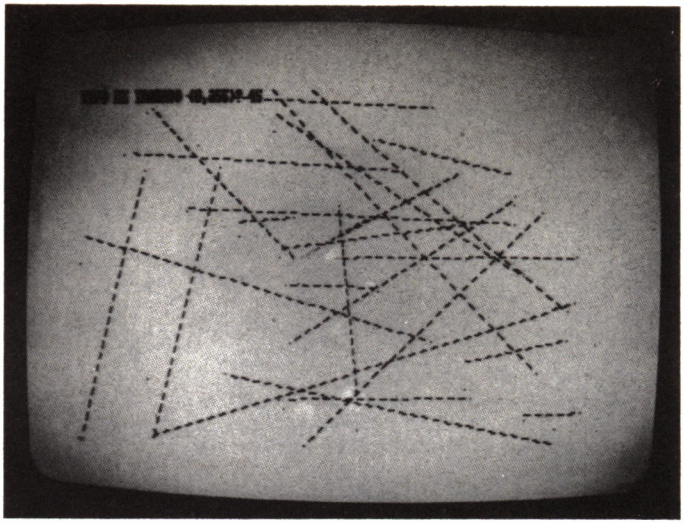


Probemos con valores bajos para apreciar mejor los efectos de la orden MASK.

Por otra parte, si poseemos un CPC 464 no debemos preocuparnos por no tener la orden MASK, dado que es posible simularla desde el propio BASIC. Tecleemos el siguiente programa, válido para todos los ordenadores CPC, equivalente al anterior, pero sin utilizar la orden MASK.

```
D'MASK2

10 CLS
20 INPUT "TIPO DE TRAZADO (0,255)";MAS
30 GOSUB 190
40 MOVE x1,y1
50 A2=(x2-x1)^2 + (y2-y1)^2
60 A1=SQR (A2)
70 MA=MAS*(A1/300)
80 PLOT x1,y1,1,0
90 PLOT x2,y2,1,0
100 XR=(x2-x1)/MA:yr=(y2-y1)/MAS
110 RAYA=0
120 FOR i=1 TO MA
130 IF RAYA=0 THEN RAYA=2:GOTO 150
140 IF RAYA=2 THEN RAYA=0
150 x1=x1+xr:y1=y1+yr
160 DRAW x1,y1,1,RAYA
170 NEXT i
180 GOSUB 190:GOTO 40
190 x1=RND(1)*600:x2=RND(1)*600
200 y1=RND(1)*400:y2=RND(1)*400
210 RETURN
```



Otra palabra BASIC no disponible en el CPC 464 es FILL, que rellena una superficie cerrada con el color de tinta que le indiquemos. Se utiliza de la siguiente manera:

FILL número de tinta

D'FILL

```
10 MODE 0:PAPER 5: BORDER 0: CLS
20 PEN 10:LOCATE 5,3:PRINT "UTILIZACION DEL FILL"
30 MOVE 32,96:DRAWR 0,160,0:DRAWR 160,0,0: DRAWR 0,-160,0: DRAWR -160,0,0
40 MOVE 440,96:DRAW 528,256,3:DRAW 616,96,3:DRAW 440,96,3
50 ORIGIN 320,176:PEN 1
60 FOR I=0 TO 2*PI STEP PI/180:PLOT COS (I)*88,SIN (I)*88,1:NEXT
70 FOR I=1 TO 1000:NEXT
80 MOVE -208,0:FILL 0
90 MOVE 0,0:FILL 1
100 MOVE 208,0:FILL 3
110 GOTO 110
```

DIBUJANDO FIGURAS

Es posible que hayamos reparado en que nuestro ordenador no dispone de ninguna instrucción que dibuje un círculo, tan sólo dando las coordenadas del origen y radio. Este es uno de los minúsculos errores del CPC, debido seguramente a un «olvido», pero fácilmente soslayable con un pequeño programa como el que listamos a continuación.

```
D'CIR2

10 CLS
20 ORIGIN 320,176
30 FOR I=0 TO 2*PI STEP PI/180:PLOT COS (I)*88,SIN (I)*88:NEXT
```

Es la versión más reducida de la función CIRCLE de otros ordenadores, aunque no la más rápida desde luego. Unicamente se trata de un bucle que calcula la posición vertical y horizontal de cada punto del círculo, al tiempo que los dibuja en la pantalla.

Otro modo de dibujar una circunferencia, si cabe más espectacular, es el siguiente:

```
D'CIR1

100 MODE 2:CLS
105 INPUT "EXCENTRICIDAD";E
110 ORIGIN 320,200
120 FOR i=0 TO PI/6 STEP PI/90
130 FOR j=0 TO 2*PI STEP PI/6
140 PLOT COS(i+j)*100*E,SIN(i+j)*100
150 NEXT j:NEXT i
160 GOTO 160
```

donde dividimos las circunferencias en doce partes que vamos trazando punto por punto. El mismo programa también permite dibujar elipses, introduciendo para ello una excentricidad distinta a 1 (por ejemplo 1.2 ó 1.5). Asimismo, observemos que podemos cambiar las coordenadas del origen de la circunferencia, tan sólo variando las de la orden ORIGIN.

Un poco más complicado resulta dibujar espirales, cuya forma, en el siguiente programa, varía según el valor de variable EL. A medida que EL vaya aproximándose a 0, más suaves y circulares serán sus vértices, hasta convertirse para valores como 0.1 ó 0.2 en una espiral circular. Es aconsejable no utilizar valores por encima de 3 ó 4.

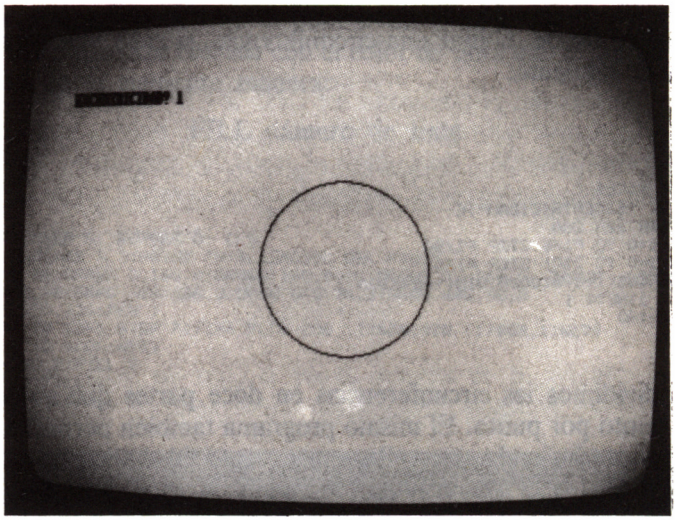
La espiral se consigue, como se puede observar, multiplicando por

dos el radio (r) y aumentándolo así a medida que va avanzando el bucle.

D'ESPI

```
10 CLS
20 INPUT "RESOLUCION ";EL
30 a=320:b=200
40 FOR i=0 TO 50 STEP EL
50 r=i*2:x=a:y=b
60 a=r*SIN(i)+320:b=r*COS(i)+200
70 MOVE x,y
80 DRAW a,b
90 NEXT i
```

El siguiente programa dibuja la función SENO en la pantalla, aunque podemos representar cualquiera definiéndola en la línea 40 de programa. Estos programas se reducen a una serie de cálculos trigonométricos, que hallan las coordenadas de los puntos que han de ser encendidos sobre la pantalla.



D'SIN

```
10 DEG
20 CLG
40 DEF FN f(x)=SIN (x)
50 FOR n=1 TO 640
60 x=x+1.125
80 PLOT n,199+FN f(x)*199
90 NEXT n
100 a$=INKEY$:IF A$="" THEN 110
110 GOTO 10
```

D'COS

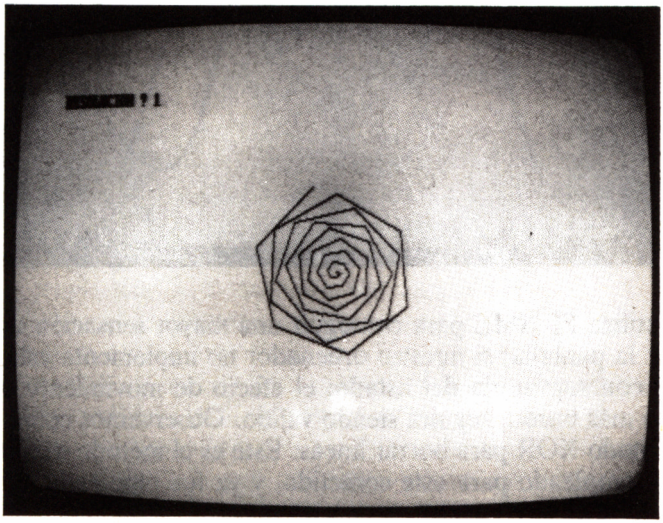
```
10 DEG
20 CLG
40 DEF FN f(x)=COS (x)
50 FOR n=1 TO 640
60 x=x+1.125
80 PLOT n,199+FN f(x)*199
90 NEXT n
100 a$=INKEY$:IF A$="" THEN 110
110 GOTO 10
```

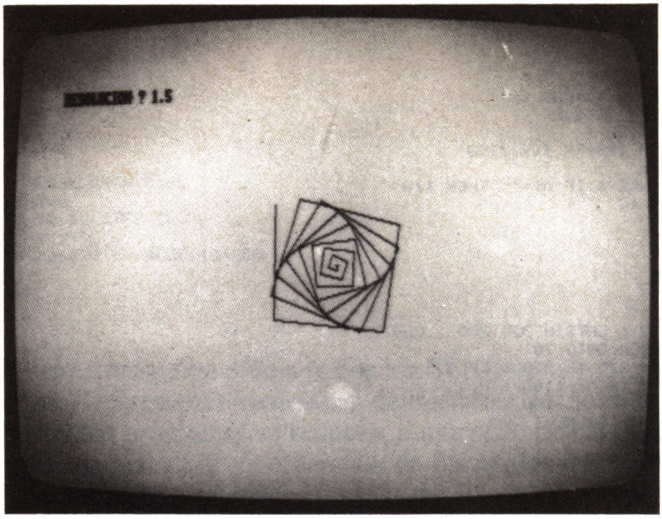
D'SINCOS

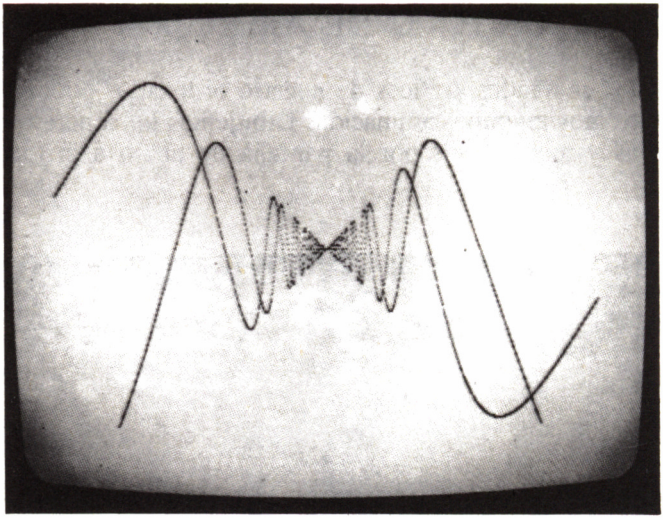
```
10 MODE 2 : ORIGIN 320,200 : CLS
20 ON ERROR GOTO 70
30 FOR X=-320 TO 320 STEP 0.5
40 PLOT X,X*COS (180/X*PI)
45 PLOT X,X*SIN (180/X*PI)
50 NEXT X : SOUND 1,200
60 GOTO 60
70 RESUME 50
```

¿Qué Tal Un Poco De Animación?

Con los comandos gráficos de nuestro ordenador podemos crear fácilmente movimiento o animación. Dibujemos un cohete con dos o tres líneas y movámoslas por la pantalla de abajo a arriba.







Se utiliza FRAME para conseguir una mayor sensación de suavidad por la pantalla; si nuestro ordenador no implementa esta orden, bastará con suprimirla del listado; el efecto de movimiento, aunque un poco más burdo, seguirá siendo válido. Observemos cómo utilizamos el modo XOR para borrar líneas. Este es el método más frecuentemente empleado para este cometido, y ya fue empleado en el programa DEFECTO.

D'FRAME

```
5 MODE 1
10 FOR n=-80 TO 400
20 MOVE 300,n
30 DRAWR 40,0,1,1:DRAWR -20,80,1,1:DRAWR -20,-80,1,1
35 FRAME
40 DRAWR 40,0,1,1:DRAWR -20,80,1,1:DRAWR -20,-80,1,1
50 NEXT n
```

CONVERSIÓN DE DATOS EN GRÁFICOS

Por el momento, hemos podido aprender que cualquier punto de la pantalla queda definido por sus coordenadas X,Y, debido a lo cual dibujar un punto o una línea es bastante fácil. Sin embargo, imaginemos que deseamos complicar un poco más las cosas.

Pensemos en una extraordinaria cancha de tenis. ¿Cómo poder llevar a nuestra pantalla semejante entramado de rectas, sin envolvernos en un complicado sinfín de sentencias MOVE y DRAW?

Pues bien, existe un procedimiento estándar que puede simplificar mucho las cosas, para el cual debemos calcular:

a) Las posiciones X,Y de todos los puntos.
b) El orden de los puntos, es decir, el orden en que deban ser dibujados sobre la pantalla.
c) La conexión entre todos los puntos, es decir, saber si dos puntos consecutivos están unidos o no.

El campo de las estructuras de los datos en los ordenadores está actualmente muy avanzado, sin embargo, vamos a limitarnos únicamente a las estructuras más simples: las matrices. No debemos asus-

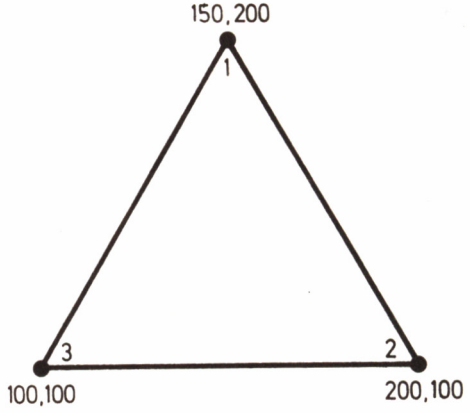
tarnos ni desanimarnos si lo que a continuación explicamos nos empieza a sonar complicado.

Para empezar, dimensionemos, creemos, dos matrices unidimensionales X(n),Y(n), donde n es el número de puntos que serán dibujados. X(1) poseerá el valor de la coordenada X u horizontal del punto número 1, por ejemplo. Como X(1) precede a X(2), las matrices permiten la ordenación de los datos, es decir, el punto X(1),Y(1) será dibujado antes que el X(2),Y(2) y así sucesivamente.

Sin embargo, también hemos de considerar las conexiones que existen entre los puntos, si están o no unidos. Para guardar esta información, dimensionemos una matriz más, esta vez bidimensional K(a,n), donde a vale 1 ó 2 (dimensión a la que hacemos referencia) y n es el número de líneas que serán dibujadas.

Para entender esto, observemos el siguiente cuadro:

n	X(n)	Y(n)	W(1,n)	W(2,n)
1	150	200	1	2
2	200	100	2	3
3	100	100	3	1



Hemos dibujado un triángulo equilátero y calculado cuales serán sus coordenadas sobre la pantalla. Tendremos tres puntos. La matriz W(2,n) nos indica el número de punto que debe unirse al punto W(1,n). Si tenemos 3 billones de puntos y decidimos que el número 317 se una al 4563, utilizaremos W(1,317) y W(2,4563) como principio y final de la línea que deseamos dibujar en la pantalla.

Observemos el siguiente programa que utiliza los datos como medio para dibujar y ahorrar líneas de programa. Dibujamos, para simplificar, la vista superior de una pista de tenis, para evitar el entramado de la red.

TENIS

```
10 REM DIBUJEMOS UNA PISTA DE TENIS VISTA DESDE ARRIBA
20 CLS
30 READ NUM:REM LEEMOS EL NUMERO DE PUNTOS
40 REM DIMENSIONAMOS Y RECOGEMOS LA INFORMACION DE LAS LINEAS DATA
50 DIM X(NUM),Y(NUM)
60 FOR N=1 TO NUM
70 READ X(N),Y(N)
80 NEXT N
90 READ NUML:DIM NL(2,NUML)
100 FOR N=1 TO NUML
110 READ NL(1,N),NL(2,N)
120 NEXT N
130 REM DIBUJAMOS
140 FOR N=1 TO NUML
150 MOVE X(NL(1,N)),Y(NL(1,N))
160 DRAW X(NL(2,N)),Y(NL(2,N))
170 NEXT N
180 GOTO 180
190 DATA 16,220,350,255,350,385,350,420,350,255,290,320,290,385,290
200 DATA 220,213,420,213,255,125,320,125,385,125,220,50,255,50,385,50,420,50
210 DATA 10,1,4,5,7,8,9,10,12,13,16,1,13,2,14,6,11,3,15,4,16
```

Podremos ver que hemos introducido dentro de las matrices creadas al principio del programa la información que tenemos almacenada en las líneas DATA del final del mismo. Sin embargo, este sistema de dibujo no es precisamente el ideal. Existen otras técnicas.

Podemos idear un programa que nos pregunte en primer lugar por el número de puntos, el de líneas, y a continuación, nos invite a teclear las coordenadas X e Y de cada punto, y los números de los puntos que han de ser unidos.

El siguiente programa realiza esa función exactamente, pero además grabará, ya sea en disco o en casete, los datos que hemos introducido.

BOX

```
10 REM
20 REM
30 INPUT"NOMBRE DEL FICHERO";H$
40 OPENOUT H$
50 INPUT"NUMERO DE PUNTOS";NPTS
60 INPUT "NUMERO DE LINEAS";LI
70 WRITE #9,NPTS
80 PRINT"ENTRE X,Y PARES"
90 INPUT"X=";X:INPUT "Y=";Y
100 WRITE #9,X
```

```
110 WRITE #9,Y
120 NEXT I
130 WRITE #9,LI
140 PRINT"NUMERO OF PUNTOS UNIDOS"
150 FOR I=1 TO LI
160 INPUT"DEL PUNTO NUMERO...";SN:INPUT"AL PUNTO NUMERO...";FI
170 WRITE #9,SN
180 WRITE #9,FI
190 NEXT I
200 CLOSEOUT
210 STOP
```

Como habremos podido ver, es la primera vez que empleamos los comandos de entrada y salida de datos. Cuidado con repetir el mismo nombre de fichero. El original pasará a llevar el apellido BAK, pero si se vuelve a reincidir en la repetición, éste desaparecerá.

Ahora ya tenemos guardados en cinta o disco los datos de nuestro dibujo. Pero, ¿cómo convertirlos en líneas en la pantalla? Para ello, necesitamos otro programa que por medio de los comandos BASIC de salida de datos, tome éstos del fichero abierto por BOX (el programa anterior) y los utilice para crear el dibujo.

```
OUTBOX

10 REM
20 REM
30 INPUT"NOMBRE DEL FICHERO";H$
40 OPENIN H$
50 REM
60 INPUT #9,NPTS
70 DIM X(NPTS),Y(NPTS)
80 FOR I=1 TO NPTS
90 INPUT #9,X(I):INPUT #9,Y(I)
100 NEXT I
110 INPUT #9,LI
120 REM
130 DIM LN(2,LI)
140 FOR I=1 TO LI
150 INPUT #9,LN(1,I),LN(2,I)
160 NEXT I
170 CLOSEIN
180 REM
190 CLS
200 FOR I=1 TO LI
210 MOVE X(LN(1,I)),Y(LN(1,I))
220 DRAW X(LN(2,I)),Y(LN(2,I))
230 NEXT I
240 STOP
```

En el programa volvemos a crear tres matrices, donde introduciremos los datos que luego utilizaremos en las dos únicas líneas MOVE y DRAW. Todo se va simplificando poco a poco, pero el cansancio puede acabar con nosotros tras calcular una cantidad considerable de

coordenadas de puntos. ¿Qué tal si fuera el ordenador quien las calculase?

DIBUJAR SIN ESFUERZO

Paso a paso, avanzando, llegamos a otras técnicas para ahorrarnos gran parte del trabajo en la confección de una imagen en la pantalla. Evitamos interminables líneas MOVE y DRAW, molestos cambios de líneas DATA, y ahora soslayaremos el cálculo mental o sobre el papel de las coordenadas de cada uno de los puntos.

Todo esto es posible gracias al programa CREATOR, preparado para ir calculando los datos de un dibujo al mismo tiempo que lo vamos realizando. Una vez terminado el dibujo, grabaremos los datos, almacenados en una matriz, en cinta o disco, y podremos convertirlos de nuevo en gráfico invocándolos bien desde el programa, o bien con una rutina que detallaremos más adelante. CREATOR es el programa base de este libro a partir de ahora, al cual iremos añadiendo, poco a poco, más instrucciones para acabar de convertirlo en un auténtico creador de gráficos en tres dimensiones.

```
5 'DANIEL LOZANO VALVERDE
6 '          &
7 'RAFAEL DE LA OSSA VILLACAÑAS
8 '
10 ZONE 10:DEG:GOSUB 270
20 REM RUTINA DE PRESENTACION Y MENU
30 MODE 1:PAPER 0:PEN 1:BORDER 5:LOCATE 16,3:PRINT "DIBUJERO"
40 RESTORE 50:FOR i=7 TO 13 STEP 2:READ a$,b$:LOCATE 2,i:PRINT a$,b$:N
EXT i
50 DATA "Cursores - Mover","Espacio  - Fijar","Delete   - Borrar"
60 DATA "B        - Cls","O        - Origen","C        - Cargar","G
    - Grabar","Return - Menu/Dibujo"
70 a$=INKEY$:IF a$=CHR$(13) THEN 90 ELSE 70
80 REM PROGRAMA PRINCIPAL
90 MODE 2:PLOT x,y:IF t>1 THEN 410
100 a$=INKEY$:IF a$="" THEN 100 ELSE a=ASC(a$)
110 IF a=240 THEN GOSUB 290:y=y+2:GOSUB 290
120 IF a=241 THEN GOSUB 290:y=y-2:GOSUB 290
130 IF a=242 THEN GOSUB 290:x=x-2:GOSUB 290
140 IF a=243 THEN GOSUB 290:x=x+2:GOSUB 290
150 IF a=32 THEN 310
160 IF a=127 THEN 380
180 IF a=66 OR a=98 THEN 480
200 IF a=79 OR a=111 THEN SOUND 1,200:O=1
220 IF a=13 THEN 30
230 IF a=67 OR a=99 THEN 600
240 IF a=71 OR a=103 THEN GOSUB 520
250 GOTO 100
260 REM INICIALIZACION DE VARIABLES
270 DIM d(1000,1):O=0:t=1:x=320:y=200:d(0,0)=1000000:d(1,0)=x:d(1,1)=y
:RETURN
280 REM MOVIMIENTO DEL CURSOR
290 IF O=1 THEN PLOT x,y,1,1:RETURN ELSE MOVE d(t,0),d(t,1):DRAW x,y,1
,1:RETURN
```

```
300 REM RUTINA FIJAR LINEA U ORIGEN
310 IF t=1000 THEN 450
340 IF O=1 THEN 510
350 REM FIJAR LINEA
360 t=t+1:d(t,0)=x:d(t,1)=y:GOTO 100
370 REM RUTINA DE BORRAR ULTIMA LINEA
380 IF d(t-1,0)=1000000 THEN 100
390 GOSUB 290::x=d(t,0):y=d(t,1):t=t-1:PLOT x,y:GOTO 100
400 REM RUTINA DE DIBUJAR
410 CLS:FOR i=0 TO t:IF d(i,0)=1000000 THEN i=i+1:MOVE d(i,0),d(i,1):G
OTO 430
420 PLOT D(I-1,0),D(I-1,1):DRAW d(i,0),d(i,1),1,1
430 NEXT i:x=d(t,0):y=d(t,1):GOTO 100
440 REM RUTINA DE MEMORIA AGOTADA
450 LOCATE 1,1:PRINT "MEMORIA  AGOTADA"
460 a$=INKEY$:IF a$="" THEN 460 ELSE 90
470 REM RUTINA DE BORRADO TOTAL
480 LOCATE 1,1:INPUT "Esta seguro de que quiere borrarlo todo?";a$:IF
a$="s" OR A$="S" THEN ERASE d:GOSUB 270
490 GOTO 90
500 REM RUTINA DE FIJAR ORIGEN
510 GOSUB 290:IF T=1 THEN T=0 ELSE T=T+1
515 D(T,0)=1000000:T=T+1:D(T,0)=X:D(T,1)=Y:O=0:PLOT X,Y:GOTO 100
520 REM RUTINA DE GRABAR
530 LOCATE 1,1:INPUT "Nombre del fichero";n$
540 OPENOUT n$:WRITE #9,T: FOR n=0 TO t
550 WRITE #9,d(n,0):WRITE #9,d(n,1)
560 NEXT n
570 CLOSEOUT
580 GOTO 90
590 REM RUTINA DE CARGA
600 LOCATE 1,1:INPUT"Nombre de fichero";N$
610 OPENIN N$:INPUT #9,T
620 FOR N=0 TO T
630 INPUT #9,D(N,0):INPUT #9,D(N,1)
640 NEXT N:CLOSEIN
650 GOTO 90
```

CÓMO SE USA CREATOR

Cuando el programa ya ha sido introducido, inmediatamente después de pulsar RUN, el menú de opciones aparecerá en la pantalla. Sólo es posible llevar a cabo las posibilidades que nos brinda el menú desde el modo de CREACION DE DIBUJO, al cual llegaremos pulsando RETURN, tal y como apunta el propio menú.

Ya en modo de creación, podremos mover un cursor mediante, valga la redundancia, las teclas del cursor (flechas). Veremos cómo un punto se desplaza a lo largo y ancho de la pantalla, el cual quedará unido, constantemente, por una línea con el centro de la pantalla, que llamaremos ORIGEN. Si pulsamos la barra espaciadora, la línea quedará fijada y el ORIGEN será ahora el final de esta línea dibujada, primitivo final. Podremos seguir moviendo el cursor y dibujar la línea deseada hasta fijarla de nuevo con la barra espaciadora.

Puede ocurrir en ocasiones que no deseemos empezar a dibujar desde el centro de la pantalla, origen inicial, o que dibujando un objeto, queramos empezar otro desde otro lugar de la pantalla. Para

ello, basta con pulsar «O» y podremos mover el cursor hasta el nuevo origen deseado, el cual fijaremos con la barra espaciadora.

Si en algún momento de la confección del dibujo pensamos que la última línea trazada no se ajusta a nuestras intenciones, podemos borrarla pulsando, obviamente, la tecla ← BORR (en el teclado inglés DEL). Una vez llevado a cabo el dibujo sobre la pantalla, debemos grabarlo en casete o disco pulsando la tecla «G», tras lo cual se nos preguntará el nombre del fichero que deseamos abrir (el nombre del dibujo); al pulsar la tecla RETURN nuestro gráfico quedará grabado.

Si ya hemos llevado a cabo esta operación, y disponemos en el disco o en el casete de un fichero de gráficos, podemos cargarlo («devolverlo» a la pantalla de nuevo) pulsando «C» y su nombre. El dibujo aparecerá en la pantalla de modo de creación.

CREATOR también nos permite eliminar todo lo dibujado hasta el momento pulsando «B», tras confirmar posteriormente que nuestra intención es realmente borrar toda la pantalla.

Ya sabemos cómo usar CREATOR, pero seguramente nos interese saber cómo funciona, nada más fácil...

INTERNÁNDONOS EN CREATOR

El alma del programa, su corazón, es una matriz definida en la línea 270, llamada D. Cada posición de D (de 1 a 1.000, tal y como lo especifica el primer dígito del paréntesis), posee dos valores, 0 y 1 (si el segundo dígito del paréntesis fuera 2, cada valor de D, poseería tres, llamados 0, 1 y 2).

Cada vez que pulsemos el espacio en el modo de dibujo, las coordenadas del lugar donde se encuentre en ese momento el cursor (X,Y), quedarán almacenadas como D(T,0) y D(T,1), siendo T el número de puntos fijados hasta el momento. Cada punto es el final de una línea y el principio de la siguiente.

Sin embargo, cuando desplazamos el origen de un lugar a otro, es decir, empezamos a dibujar otra serie de líneas en otra parte de la pantalla, nuestro Amstrad asignará a D(T,0) el valor 1.000.000, aumentará | en uno, y a continuación situará las coordenadas del nuevo origen. De esta manera, cuando el ordenador dibuje leyendo los datos de la matriz, en las líneas 410-430, el gráfico que hayamos dibujado con anterioridad, y encuentre un valor 1000000 en un D(T,0), hará caso omiso de D(T,1) y utilizará el siguiente par de

números como coordenadas de un nuevo origen, es decir, sólo dibujará un punto, y no una línea desde el punto anterior hasta ese.

Para que podamos comprender a la perfección la totalidad del programa, vamos a llevar a cabo la lectura de CREATOR...

COMO SI DE UN LIBRO SE TRATARA

El programa CREATOR se lee de la siguiente manera:

10 Colócame diez espacios entre cada mensaje PRINT: Desde ahora, todos los cálculos se harán en grados: Vete a la subrutina de la línea 270.

30 Coloca el modo 1: El papel será el de la pluma cero: Usaremos la pluma 1, y el borde será malva: Coloca el cursor en la fila 16 de la columna tercera: Escribe el mensaje «DIBUJERO»,

40 Empieza a leer los datos de la línea 50 por el principio: De 7 a 13, de dos en dos: Lee dos mensajes: Coloca el cursor en la fila 2, columna 7-9-11-13 según toque: Escribe los mensajes: Vuelve si no has terminado.

50-60 Mensajes.

70 Desde ahora almacena en A$ la tecla que sea pulsada: Si se pulsa RETURN vete a la línea 90, si no, espera a que sea pulsado.

90 Coloca el modo 2: Dibuja un punto en las coordenadas del cursor: Si el número de puntos fijados hasta el momento es mayor que 1, vete a la línea 410.

100 Desde ahora almacena en A$ la tecla que sea pulsada: Si no se pulsa ninguna, espera a que lo sea; almacena en la variable A el valor ASCII de la tecla pulsada.

110 Si se pulsa «cursor arriba» vete a la subrutina de la línea 290: Aumenta en dos la coordenada vertical: Vete a la subrutina de la línea 290.

120 Si se pulsa «cursor abajo» y disminuye en dos Y.

130 Si se pulsa «cursor izda.» y disminuye en dos X.

140 Si se pulsa «cursor dcha.» y aumenta en dos X.

150 Si se pulsa la barra espaciadora vete a la línea 310.

160 Si se pulsa ← BORR vete a la línea 380.

180 Si se pulsa «B» o «b» vete a la línea 480.

200 Si se pulsa la «O» o la «o» entonces pita y avisa al ordenador que estamos en el modo de origen asignando a 0 el valor 1.

220 Si se pulsa «RETURN» coloca el menú.

230 Si se pulsa «C» o «c» vete a la línea 600.

240 Si se pulsa «G» o «g» vete a la línea 520.

250 De aquí a la línea 100.

270 Queda dimensionada una matriz, llamada D(1000,1): Por otra parte, el modo de origen es 0, es decir, desconectado: El cursor está colocado en el punto de coordenadas (320,200): Asignamos al primer valor de la matriz el valor 1000000, porque las próximas coordenadas son las de un origen, el inicial, las cuales introducimos en la matriz.

290 Si estamos intentando cambiar el origen, borra el punto (X,Y): Si no, muévete hasta las coordenadas del punto inicial (el último punto fijado) y dibuja una línea desde ahí hasta el cursor. ¡Ah!, y vuelve.

310 Si hemos fijado más de 1000 puntos vete a la rutina de «Esto se ha acabado»:

340 Si estamos intentando cambiar el origen vete a la línea 510.

360 Aumenta en uno el contador de puntos: Introducimos datos de las coordenadas en la matriz y vuelve al bucle principal en 100.

380 Si el valor de la primera coordenada del punto anterior es la de «aviso, que viene un origen», vuelve al bucle principal que no hay línea que borrar.

390 Vete a la subrutina de la línea 290: Introduce en la matriz los valores de las coordenadas actuales del cursor: Retrocede un punto, es decir, olvida el último que hemos fijado: Dibuja el cursor y vuelve al bucle principal.

410 Borra la pantalla: Haz lo siguiente tantas veces como puntos hayamos fijado: Si la primera coordenada de uno de los pares de la matriz es un millón, pasemos al siguiente, coloca un punto allí y vete a la línea 430.

420 Dibuja un punto en las coordenadas del anterior punto (punto inicial de la línea) y una línea desde éste hasta la coordenada del último punto fijado. Pero, ¡cuidado!, no borres nada por el camino.

430 Si no has terminado, vuelve a la 410: Las coordenadas del cursor son las del último punto fijado: Vete al bucle principal.

450 Coloca el cursor de textos en la fila 1 y columna 1: Notifica que esto se ha acabado.

460 Desde ahora almacena en A$ la tecla que sea pulsada: Si no se pulsa ninguna, espera; vete a la línea 90.

480 Coloca el cursor de texto en la fila 1, columna 1: Asegúrate que se desea borrar todo el gráfico. Si la respuesta es «S» o «s» arrasa la matriz, allá él. Ve a la subrutina de la línea 270.

490 Vuelve a empezar de nuevo por la línea 90.

510 Vete a la subrutina de la línea 290: Si acabamos de empezar coloca a T el valor 0; si no, añádele un punto a esta misma variable.

515 Como estamos fijando un origen, colocamos el valor 1000000 en el primer par de la matriz: Ahora apuntâmos que hemos fijado un punto: Introducimos en la matriz las coordenadas del nuevo origen, el cual reconocerá el ordenador por el millón colocado en la posición cero del par anterior: Dibuja un punto en esta posición y vuelve al bucle principal.

530 Coloca el cursor de texto en la fila 1, columna 1: Pregunta el nombre del fichero e introdúcelo en N$.

540-570 Operaciones de grabación de datos.

600 Coloca el cursor en la fila 1, columna 1: Pregunta el nombre del fichero que se desea cargar e introdúcelo en N$.

610-650 Operaciones de carga de datos.

Esto es todo, así de fácil. A medida que vayamos estudiando manipulaciones con los datos introducidos dentro de una matriz, iremos aumentando y sofisticando nuestro programa CREATOR.

LISTA DE VARIABLES

La presente lista de variables, nos ayudará a comprender aún mejor el comportamiento de CREATOR.

D - Matriz central y única donde son introducidas todas las coordenadas de todos los puntos fijados sobre la pantalla.

T - Lleva la cuenta de los puntos fijados hasta un momento determinado. Al principio es 1 y su valor máximo es 1000.

A - Es igual al valor ASCII de la tecla pulsada dentro de la pantalla de dibujo.

O - Indica al ordenador si deseamos cambiar el origen (O=1) o si no es así (O=0).

X,Y - Coordenadas del cursor que nosotros movemos por la pantalla.

N$ - Nombre del fichero grabado o por grabar.

GRÁFICOS EN DOS DIMENSIONES

En este capítulo aprenderemos a mover y manipular gráficos en dos dimensiones por la pantalla de nuestro ordenador; sin embargo, antes de nada vamos a repasar algunas nociones básicas sobre los sistemas de coordenadas en dos dimensiones.

SISTEMA DE COORDENADAS

Sean dos ejes perpendiculares, uno vertical y otro horizontal. El punto de intersección entre ambos es el origen. Llamaremos Y al vertical y X al horizontal. La parte de encima del origen del eje Y, y la parte a la derecha del origen del eje X son positivas, siendo negativas las zonas inferior e izquierda de los ejes Y y X, respectivamente.

Hemos dividido, pues, el espacio en cuatro cuadrantes. La pantalla de nuestro ordenador se encuentra inicialmente en el cuadrante superior derecho, dado que el origen de ésta se halla en la esquina inferior izquierda, centro de los cuadrantes. Esto no implica que las

coordenadas de cualquier punto que se encuentre en nuestra pantalla posea unas coordenadas positivas, pues como sabemos, podemos cambiar la situación del origen.

Ya sabemos representar un punto o líneas entre puntos, por cualquier parte de la pantalla, pero ¿y si quisiéramos mover un objeto dibujado, cambiarlo de tamaño o rotarlo?

Toda técnica cuya finalidad sea la de realizar cualquiera de estas operaciones se llama, en la jerga de los gráficos, transformadora, siendo las principales transformaciones la rotación, la traslación y la alteración del tamaño.

Para llevar a cabo toda esta serie de transformaciones utilizaremos las matrices algebraicas, un magnífico sistema para manipular datos.

Para entender con claridad toda esta serie de cambios, modificaciones o transformaciones de un gráfico, no hay nada como ver un ejemplo. Para ello, incluimos en este capítulo el programa CREATOR 2D, el cual «MERGEado» con el programa CREATOR del capítulo anterior, nos permitirá, tanto crear gráficos como transformarlos, es decir, trasladarlos, rotarlos, aumentarlos y disminuirlos.

CREATOR2

```
5 'EXTENSION DEL PROGRAMA CREATOR
6 'POR DANIEL LOZANO VALVERDE
7 '          &
8 'RAFAEL DE LA OSSA VILLACA\AS
40 RESTORE 50:FOR i=7 TO 19 STEP 2:READ a$,b$:LOCATE 2,i:PRINT a$,b$:N
EXT i
50 DATA "Cursores - Mover","Espacio  - Fijar","Delete   - Borrar","R
      - Rotar","T         - Trasl.","A         - Aumentar"
60 DATA "B         - Cls","O         - Origen","C         - Cargar","G
     - Grabar","","","","","Return - Menu/Dibujo"
170 IF a=82 OR a=114 THEN SOUND 1,200:o=1:r=1
190 IF a=84 OR a=116 THEN SOUND 1,200:o=1:tr=1
210 IF a=65 OR a=97 THEN 770
270 DIM d(1000,1):R=0:TR=0:O=0:t=1:x=320:y=200:d(0,0)=1000000:d(1,0)=x
:d(1,1)=y:RETURN
320 IF r=1 THEN 670
330 IF tr=1 THEN 720
660 REM RUTINA DE ROTACION
670 r=0:o=0:LOCATE 1,1:INPUT "Angulo";a
680 FOR i=1 TO t:IF d(i,0)=1000000 THEN 700
690 xr=d(i,0)-x:yr=d(i,1)-y:c=SQR(xr*xr+yr*yr):ang=a+ATN(yr/xr)+180*(x
r<0):d(i,0)=x+c*COS(ang):d(i,1)=y+c*SIN(ang)
700 NEXT i:GOTO 90
710 REM RUTINA DE TRASLACION
720 tr=0:o=0:xr=x-d(t,0):yr=y-d(t,1)
730 FOR i=1 TO t:IF d(i,0)=1000000 THEN 750
740 d(i,0)=d(i,0)+xr:d(i,1)=d(i,1)+yr
750 NEXT i:GOTO 90
760 REM RUTINA DE AUMENTO
770 LOCATE 1,1:INPUT "Fraccion de aumento";au
780 FOR i=1 TO t:IF d(i,0)=1000000 THEN 800
790 d(i,0)=d(i,0)*au:d(i,1)=d(i,1)*au
800 NEXT i:GOTO 90
```

TRASLACIÓN DE GRÁFICOS

Como es lógico, es posible que nos interese desplazar el gráfico por la pantalla. Esto a priori puede parecer complicado, pero es algo realmente sencillo. Existen dos técnicas de traslación de gráficos: en una de ellas se trata de trasladar el origen de la pantalla, y en la otra cambiar todas las coordenadas de los puntos que componen nuestro gráfico.

La técnica del cambio de origen es la más simple de las dos, pues basta con hacer uso de la instrucción ORIGIN. Por ejemplo, si deseamos desplazar el gráfico veinte puntos hacia arriba, deberemos incluir en el programa:

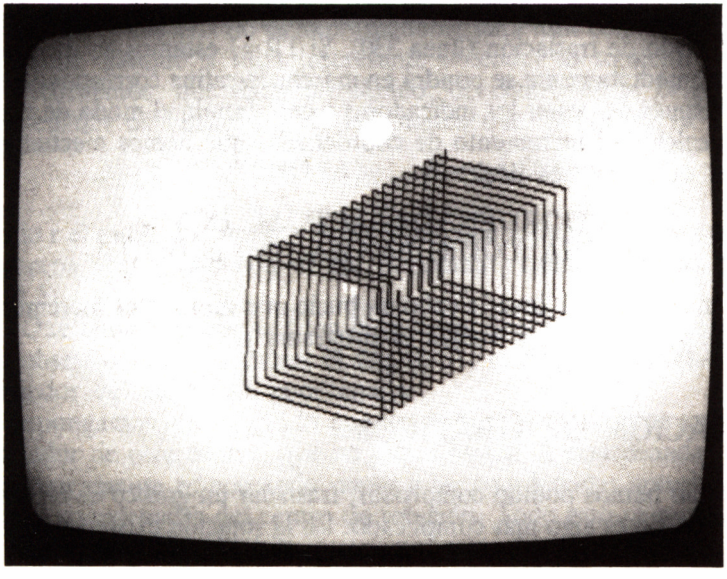
ORIGIN 0,20

Como se podrá comprobar, ORIGIN cambia el emplazamiento del origen que inicialmente se encuentra en la esquina inferior izquierda de la pantalla. Pero esta instrucción tiene un pequeño problema; si ahora deseáramos desplazar el gráfico cuarenta puntos hacia la derecha, deberíamos escribir

ORIGIN 40,20

en vez de

ORIGIN 40,0



por lo cual, para utilizar esta técnica hay que tener en cuenta la situación actual del origen.

El segundo método posee la ventaja de tener siempre fijo el origen en la esquina inferior izquierda de la pantalla, lo cual nos permite hacernos una idea exacta de la situación del gráfico. El programa CREATOR 2D traslada los gráficos basándose en este último sistema.

Basta, pues, con sumar a cada punto de la matriz D los incrementos horizontal y vertical de la traslación. La rutina es bien simple:

```
710 REM RUTINA DE TRASLACION
720 TR=0:O=0:XR=X-D(T,0):YR=Y-D(T,1)
730 FOR I=1 TO T:IF D(I,0)=1000000 THEN 750
740 D(I,0)=D(I,0)+XR:D(Y,1)=D(Y,1)+YR
750 NEXT I:GOTO 90
```

Para empezar, la rutina pone a cero los indicadores TR y o, es decir, cuando pulsamos la tecla «T» el programa (línea 190) emite un sonido y asigna el valor 1 a ambas variables; el programa se encuentra en el modo de traslación. A partir de ahora, podremos mover el cursor para indicar la traslación que deseamos efectuar. En el momento en que pulsemos la barra espaciadora, el ordenador llevará el último punto dibujado a la nueva posición del cursor, y con él todo el gráfico.

Al pulsar la barra espaciadora, el programa pregunta si estamos en el modo de traslación (línea 330). Si TR=1 estamos en dicho modo, e inmediatamente se pondrá en marcha la rutina correspondiente.

Después de poner los indicadores a cero (anula el modo de traslación) calcula el incremento de coordenadas que hemos efectuado al mover el cursor, es decir:

COORDENADA ACTUAL (X,Y) -
ULTIMA COORDENADA (D(T,0),D(T,1))

Tan sólo queda llevar a cabo un bucle que sume estos incrementos a cada coordenada, y vuelva al programa principal.

ROTACIÓN DE GRÁFICOS

Como hemos podido comprobar, trasladar un gráfico es bien sencillo; rotarlo es algo más complicado. En la rotación existen dos datos imprescindibles: el centro de rotación y el ángulo de rotación.

Al pulsar la barra espaciadora, el programa asigna el valor 1 a las variables R y O, que en este caso son los indicadores del modo de rotación. A partir de este momento, deberemos elegir con los cursores el centro de rotación deseado y pulsar de nuevo la barra espaciadora. Entonces entra en funcionamiento la rutina para la rotación del gráfico. Veámosla:

```
660 REM RUTINA DE ROTACION
670 R=0:O=0:LOCATE 1,1:INPUT "Angulo";A
680 FOR I=1 TO T:IF D(I,0)=1000000 THEN 700
690 XR=D(I,0)-X:YR=D(I,1)-Y:C=SQR(XR*XR*YR*YR):
    ANG=A+ATN(YR/XR)+180*(XR<0):D(I,0)=X+C*
    COS(ANG):D(I,1)=Y+C*SIN(ANG)
700 NEXT I:GOTO 90
```

Del mismo modo que antes, lo primero que hace esta rutina es anular el modo de rotación (igualando R y O a cero), para después preguntar el ángulo deseado. Dicho ángulo deberá estar expresado en grados sexagesimales, y no en radianes. Si preferimos trabajar en el otro sistema angular, bastará con eliminar el comando DEG de la primera línea y sustituir el 180 de la línea 690 por PI.

Antes de continuar explicando la rutina del programa, veamos la manera de rotar un dibujo en dos dimensiones. Los lectores que no hayan realizado estudios de trigonometría pueden saltarse la próxima explicación, pues ésta no es imprescindible para usar correctamente el programa, aunque sí sería interesante entender lo siguiente:

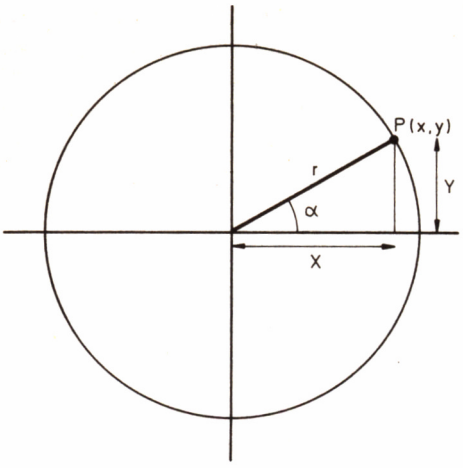
Recordemos que dado un círculo, de centro en el origen, las coordenadas de cualquiera de los puntos que lo forman es

$$X = r * \text{COSENO}(A) \quad , \quad Y = r * \text{SENO}(A)$$

siendo r el radio del círculo y A el ángulo que forman la recta que une el centro y el punto con el eje horizontal, como indica la figura (18.4.2).

¿Qué tiene esto que ver con la rotación de un gráfico? Pues basta con observar que al cambiar el ángulo A, el punto va ROTANDO alrededor del centro de coordenadas.

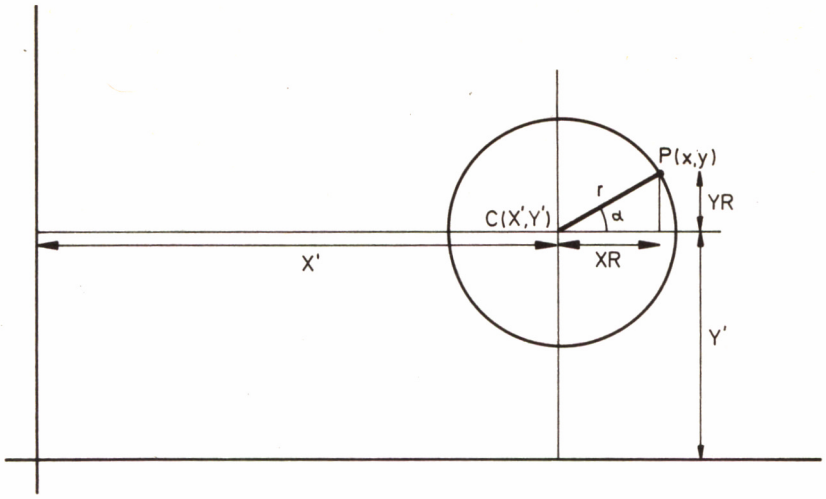
Supongamos ahora que lo que deseamos hacer es girar un punto alrededor de otro centro, que no sea el origen de coordenadas. Para ello, únicamente es necesario sumar a las coordenadas anteriores las nuevas coordenadas del centro de rotación. Todo esto se muestra en la figura siguiente (18.4.3).



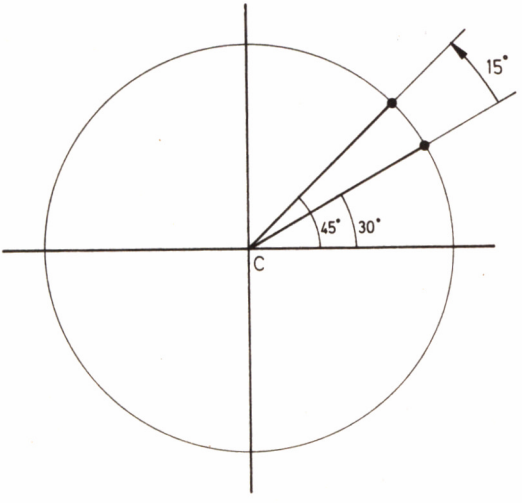
18.4.3.

Dicho esto, parece bastante fácil hacer girar un objeto alrededor de cualquier punto de la pantalla. Basta con aplicar estas fórmulas a cada uno de los puntos de la matriz D, y habremos terminado. Pero todavía nos queda un pequeño detalle. El ángulo que nosotros pretendemos girar el dibujo no es el verdadero ángulo de rotación, que debemos incluir en estas fórmulas, puesto que el ángulo de dichas fórmulas está expresado en relación a la horizontal y el ángulo deseado por nosotros es relativo al ángulo que ya posee el punto. Dicho de otra forma, si queremos rotar el punto 15 grados, y este forma un ángulo de 30 grados con la horizontal, debemos hacer girar el punto un total de 45 grados. Este es el ángulo que tendremos que incluir en las fórmulas (figura 18.4.4).

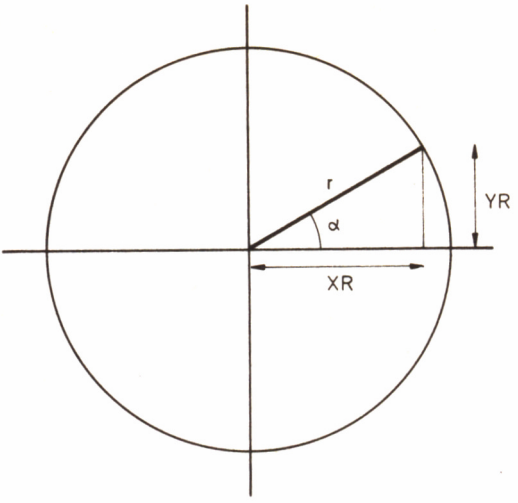
Desgraciadamente, este ángulo nos es desconocido. Para hallarlo debemos hacer uso de otra función llamada arco tangente, la cual nos dice cuál es el ángulo cuya tangente es el valor indicado entre paréntesis. Como la tangente de un ángulo es el seno dividido por el coseno, nuestro ángulo será el arco tangente de YR/XR, siendo XR e YR las coordenadas del punto relativas al centro de rotación (figura 18.4.5).



18.4.4.



18.4.5.



18.4.6.

El único problema de este método es que la función arco tangente falla cuando la coordenada XR es negativa. Pero esto es fácilmente solucionable sumándole al ángulo 180 grados (PI en radianes) cuando se dé este caso.

Volviendo a la rutina del programa, ésta pone en marcha un bucle para rotar cada punto por separado. En primer lugar, el bucle calcula las distancias horizontal y vertical entre el centro de rotación (X,Y), y el punto que vayamos a rotar (D(I,0),D(I,1)). Dichas distancias se almacenan en las variables XR e YR respectivamente. Una vez hecho esto, calcula la distancia total (C) entre el centro de rotación y el punto a rotar por el Teorema de Pitágoras.

Con estos datos, ya estamos en disposición de utilizar las fórmulas matemáticas que calculan el valor del ángulo total que debemos girar el punto, es decir, el ángulo introducido por nosotros, más el ángulo que tuviese el punto por su situación en la pantalla, con respecto al centro de rotación señalado.

AUMENTO Y DISMINUCIÓN DE GRÁFICOS

Pulsando la tecla «A» se canaliza la rutina de aumento y disminu-

ción. En este caso, no hace falta tener en cuenta ningún punto en especial, basta con indicar el número de veces que queremos aumentar o disminuir el gráfico en cuestión.

Por ejemplo, si queremos aumentar el dibujo al doble de su tamaño actual, deberemos introducir el valor 2 ante la pregunta «¿Fracción de aumento?», que nos formulará el ordenador. Si deseamos que el dibujo retorne a su tamaño primitivo, no habrá más que volver a pulsar «A» e introducir esta vez el valor 0.5, es decir, 1/2. Como se puede comprobar, el valor que indique la fracción de aumento o disminución depende única y exclusivamente del tamaño actual del gráfico.

Dicho esto, la rutina es muy fácil de entender:

```
760  REM RUTINA DE AUMENTO
770  LOCATE 1,1:INPUT"Fraccion de aumento";AU
780  FOR I=1 TO T:IF D(I,0)=1000000 THEN 800
790  D(I,0)=D(I,0)*AU:D(I,1)=D(I,1)*AU
800  NEXT I:GOTO 90
```

Una vez introducida la fracción de aumento (línea 770) se pone en marcha un bucle que multiplica cada coordenada por dicho valor. Haciendo esto, se consigue una especie de «efecto lupa», ya que cada coordenada se reduce o se aumenta, con respecto al origen de coordenadas, tantas veces como se halla indicado en la variable AU. Como es lógico, el centro de coordenadas permanece constante al ser sus coordenadas (0,0).

Hay que tener en cuenta que si aumentamos el gráfico, este podría salirse de los límites de la pantalla. No es problema. Utilizando la rutina de traslación, podremos desplazar el dibujo como queramos por toda la pantalla, y colocarlo de nuevo dentro de sus márgenes.

Listado Del Programa

El programa CREATOR 2D sólo puede ser utilizado si previamente se ha tecleado y grabado el programa CREATOR del capítulo anterior. Después de teclear CREATOR 2D, grabémoslo, para posteriormente cargar CREATOR. Una vez llevada a cabo la operación, bastará con la orden MERGE"CREATOR2D, para obtener la unión de ambos programas.

EXTENSIÓN DEL PROGRAMA CREATOR 2D

Otra técnica algo más compleja para aumentar o disminuir los gráficos es el sistema de ventanas. El BASIC de los CPC posee una instrucción llamada WINDOW, cuya traducción al castellano es ventana. No nos llevemos a engaño, dicha instrucción no tiene nada que ver con este sistema.

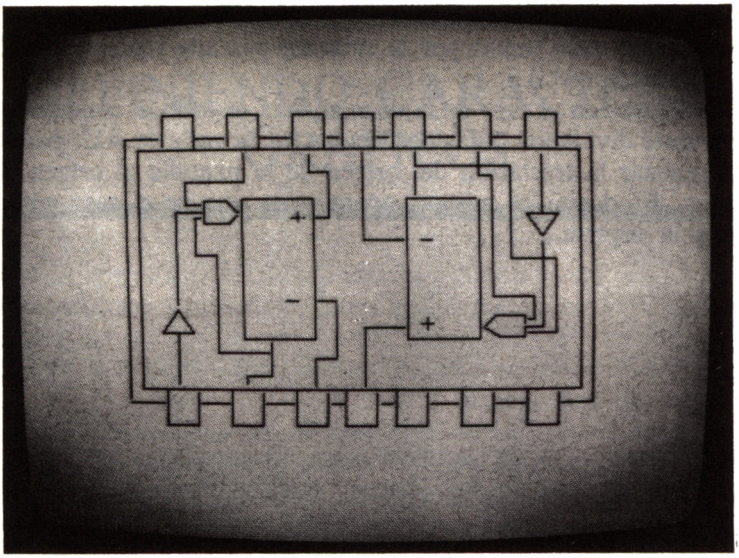
El programa CREATOR 3 es una rutina que amplía las posibilidades de aumento y disminución de gráficos ya existentes en el programa CREATOR 2D. El programa está preparado para ser «MERGEado» con CREATOR 2D, con lo cual obtendremos una versión más completa de este programa.

```
CREATOR3

5 'EXTENSION DEL PROGRAMA CREATOR 2D
6 'POR DANIEL LOZANO VALVERDE
7 '                &
8 'RAFAEL DE LA OSSA VILLACAÑAS
760 REM RUTINA DE AUMENTO
770 x=208:y=130:xr=223:yr=138:GOSUB 880
780 a$=INKEY$:IF a$="" THEN 780 ELSE a=ASC(a$)
790 IF a=240 THEN GOSUB 880:y=y+2:GOSUB 880
800 IF a=241 THEN GOSUB 880:y=y-2:GOSUB 880
810 IF a=242 THEN GOSUB 880:x=x-2:GOSUB 880
820 IF a=243 THEN GOSUB 880:x=x+2:GOSUB 880
830 IF a=48 THEN GOSUB 880:xr=xr*0.9:yr=yr*0.9:GOSUB 880
840 IF a=49 THEN GOSUB 880:xr=xr/0.9:yr=yr/0.9:GOSUB 880
850 IF a=13 THEN 90
860 IF a=32 THEN 890
870 GOTO 780
880 MOVE x,y:DRAWR xr,0,1,1:DRAWR 0,yr,1,1:DRAWR -xr,0,1,1:DRAWR 0,-yr
,1,1:RETURN
890 LOCATE 1,1:INPUT "Aumento o disminucion (a/d)";a$
900 au=640/xr:IF a$="d" OR a$="D" THEN 940
910 FOR i=1 TO t:IF d(i,0)=1000000 THEN 930
920 d(i,0)=(d(i,0)-x)*au:d(i,1)=(d(i,1)-y)*au
930 NEXT i:GOTO 90
940 FOR i=1 TO t:IF d(i,0)=1000000 THEN 960
950 d(i,0)=d(i,0)/au+x:d(i,1)=d(i,1)/au+y
960 NEXT i:GOTO 90
```

Una vez pulsada la tecla «A» aparecerá en la pantalla una ventana, que podremos mover haciendo uso de los cursores. Cuando tengamos colocada dicha ventana en el lugar deseado, podremos también hacerla variar de tamaño, esta vez utilizando las teclas «1» y «0». El «1» sirve para aumentar la ventana y el «0» para disminuirla.

Pulsando la barra espaciadora aparecerá en la pantalla la opción: «Aumento o disminución (a/d)?» Contestando «a» a esta pregunta, el ordenador ampliará la parte del gráfico comprendida dentro de los márgenes de la ventana, a toda la pantalla. Haciendo uso de la otra posibilidad, pulsando «d» y RETURN, el gráfico contenido en la pantalla se reducirá al tamaño de la ventana y se colocará en el lugar ocupado por ésta.



En el programa CREATOR 3, X e Y son las coordenadas de la esquina inferior izquierda de la ventana, y XR e YR son las dimensiones horizontal y vertical, respectivamente, de la misma. Todo esto se inicializa en la línea 770.

Las líneas 780 a 870 son el bucle principal que se encarga de averiguar qué tecla es pulsada en cada momento. Pulsando RETURN se vuelve al programa principal y se anula la orden de aumento o disminución.

Para conseguir el aumento de la ventana, basta con dividir las variables XR e YR por 0.9. En el caso de la disminución, ocurre todo lo contrario; hay que multiplicar ambas variables por 0.9. Si se desea, se puede cambiar este valor. Cuanto más próximo a cero sea más rápidamente aumentará o disminuirá la ventana. No hay que olvidar que dicho valor no debe ser nunca superior a 1, ni inferior o igual a cero.

En la línea 880 se encuentra la subrutina que dibuja la ventana. En la línea 900, una vez pulsada la barra espaciadora y elegida la opción, el ordenador calcula la relación entre el tamaño de la pantalla y el de la ventana. Sólo es necesario hacerlo con las coordenadas horizontales (640 y XR), pues las verticales tienen la misma proporción.

En el caso de que se haya elegido la opción de aumentar, en las líneas 910-930 se encuentra el bucle que realiza este aumento. Para ello, se lleva a cabo, en primer lugar, una traslación del origen de la

ventana al origen de coordenadas de la pantalla y, en segundo lugar, se multiplica cada punto por la relación pantalla-ventana calculada anteriormente (AU).

De nuevo, en el caso de la disminución, es todo lo contrario. Primeramente dividimos cada coordenada por la variable AU, para después realizar la traslación del origen de la pantalla al origen de la ventana. Es decir, las mismas operaciones que en el caso anterior, pero a la inversa.

DEFINIENDO CARACTERES GRÁFICOS

Qué son los caracteres gráficos? Buena pregunta, y muy fácil de contestar: encendamos el ordenador y pulsemos una tecla cualquiera de las que tienen serigrafiada una letra; en la pantalla habrá aparecido, seguramente, alguna letra. Si no es así será mejor que llevemos el ordenador al taller, o probemos a conectar el aparato a la red...

En todo caso, cualquier letra o cualquier signo que haya aparecido en la pantalla es un carácter gráfico.

¿Cuál es la utilidad de estos caracteres? La respuesta a esta pregunta, si cabe, es todavía más sencilla. Si no existieran dichos caracteres gráficos no podríamos leer ni escribir nada en el monitor, debido a lo cual no podríamos entendernos con la máquina.

El ordenador tiene en su memoria ROM una zona de números denominados JUEGO de CARACTERES. Cada vez que se le ordena presentar un mensaje en la pantalla acude a ese juego de caracteres y en él LEE los códigos correspondientes a cada signo o a cada letra, para después visualizarlos en el monitor.

Como se puede comprobar, dichos caracteres son absolutamente indispensables a la hora de ejecutar cualquier programa en el ordena-

dor. Todas las letras, los números y signos de puntuación están almacenados en la memoria ROM, en el juego de caracteres.

Pero esto no es lo más interesante de ellos. Lo que hace verdaderamente importante este juego de caracteres es que se puede modificar, es decir, el ordenador nos brinda la posibilidad de cambiar dichos códigos para conseguir un nuevo vocabulario a nuestro gusto.

De esta manera, podemos crear nuestras propias letras y números, así como signos especiales que no tenga incorporados el ordenador por sí mismo. Por ejemplo, tratemos de escribir la palabra «Cigüeña» en un Amstrad de la serie CPC y nos encontraremos con gran sorpresa que es imposible colocar una diéresis sobre la u. La razón es bien simple: el ordenador no tiene diéresis en su juego de caracteres, por tanto, no nos esforcemos.

Sin ir más lejos, ¿hemos visto acentos por algún rincón de nuestro Amstrad? Podremos pasar largas tardes de frío invierno buscándolos a lo largo y a lo ancho del teclado del ordenador y tampoco los encontraremos. Y esto sólo son un par de ejemplos...

Por tanto, como acabamos de ver, lo más importante del juego de caracteres de un ordenador es que se pueda cambiar a nuestro gusto, y muchos de ellos, posiblemente inútiles para muchos de nosotros, pueden ser transformados en otros de uso frecuente.

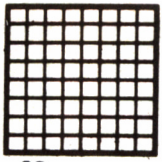
Organización Del Juego De Caracteres

Antes de estudiar la técnica para transformar los caracteres gráficos del ordenador veamos cómo están constituidos cada uno de ellos.

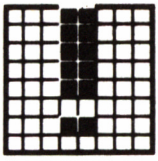
Para empezar, tomemos rápidamente una hoja de papel cuadriculado y dibujemos en ella un cuadrado que tenga ocho casillas de ancho por ocho de alto. Esta es la forma en que el ordenador VE dichos caracteres (aunque esto no sea muy exacto).

La realidad es que la máquina está dispuesta para representar los gráficos que se pueden construir con una serie de ocho por ocho puntos. Para entender esto con mayor facilidad, podemos observar la configuración inicial de todos y cada uno de los caracteres de nuestro Amstrad. Como vemos, todas y cada una de las letras, números y demás signos se pueden representar dentro de una cuadrícula de ocho por ocho casillas, denominada matriz.

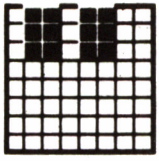
Así, pues, elijamos nuestro propio carácter y dibujémoslo dentro de la cuadrícula confeccionada en la hoja, teniendo en cuenta que cada celdilla corresponde a un punto encendido o apagado en la pantalla.



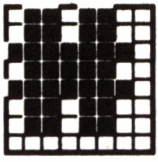
32



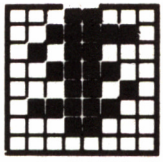
33



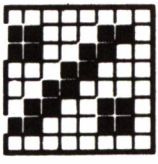
34



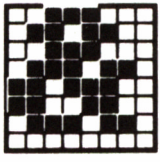
35



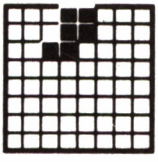
36



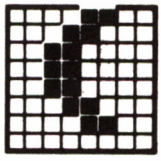
37



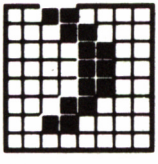
38



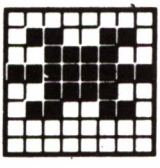
39



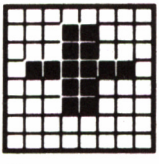
40



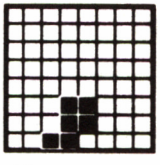
41



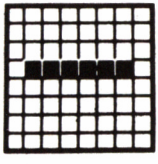
42



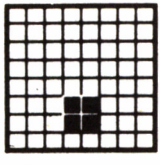
43



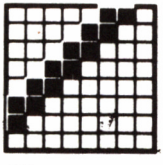
44



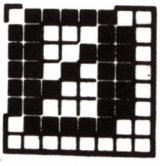
45



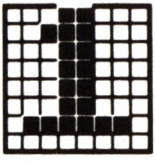
46



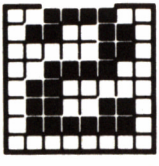
47



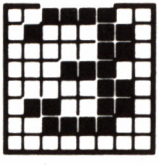
48



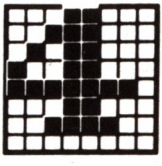
49



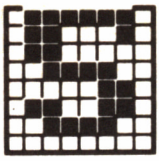
50



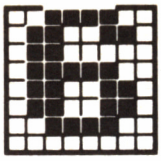
51



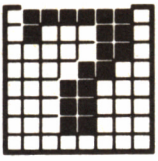
52



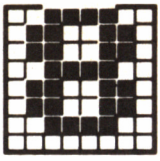
53



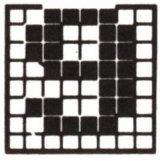
54



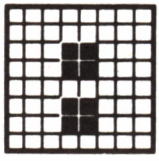
55



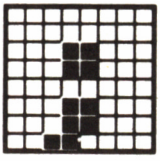
56



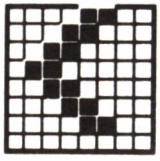
57



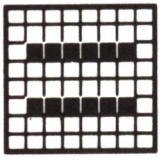
58



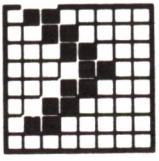
59



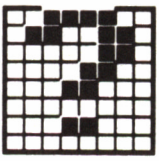
60



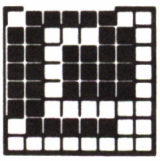
61



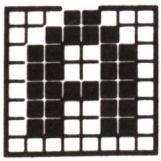
62



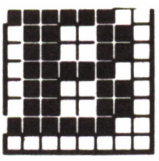
63



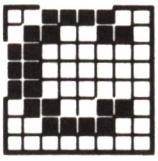
64



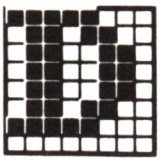
65



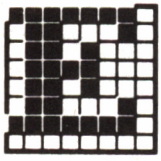
66



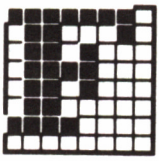
67



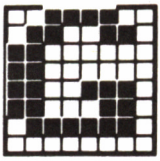
68



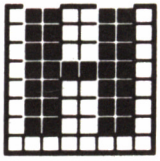
69



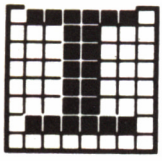
70



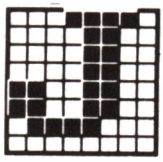
71



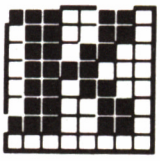
72



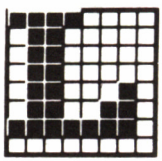
73



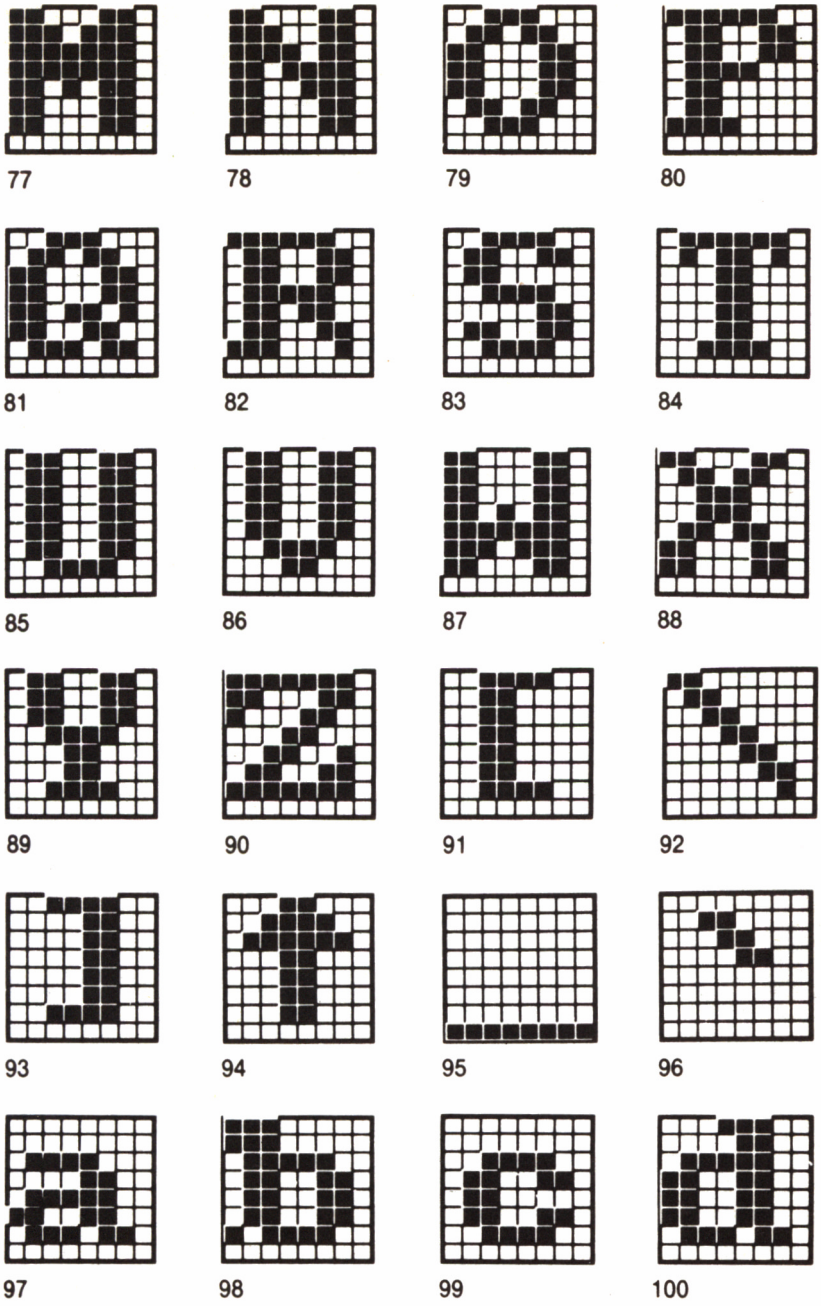
74

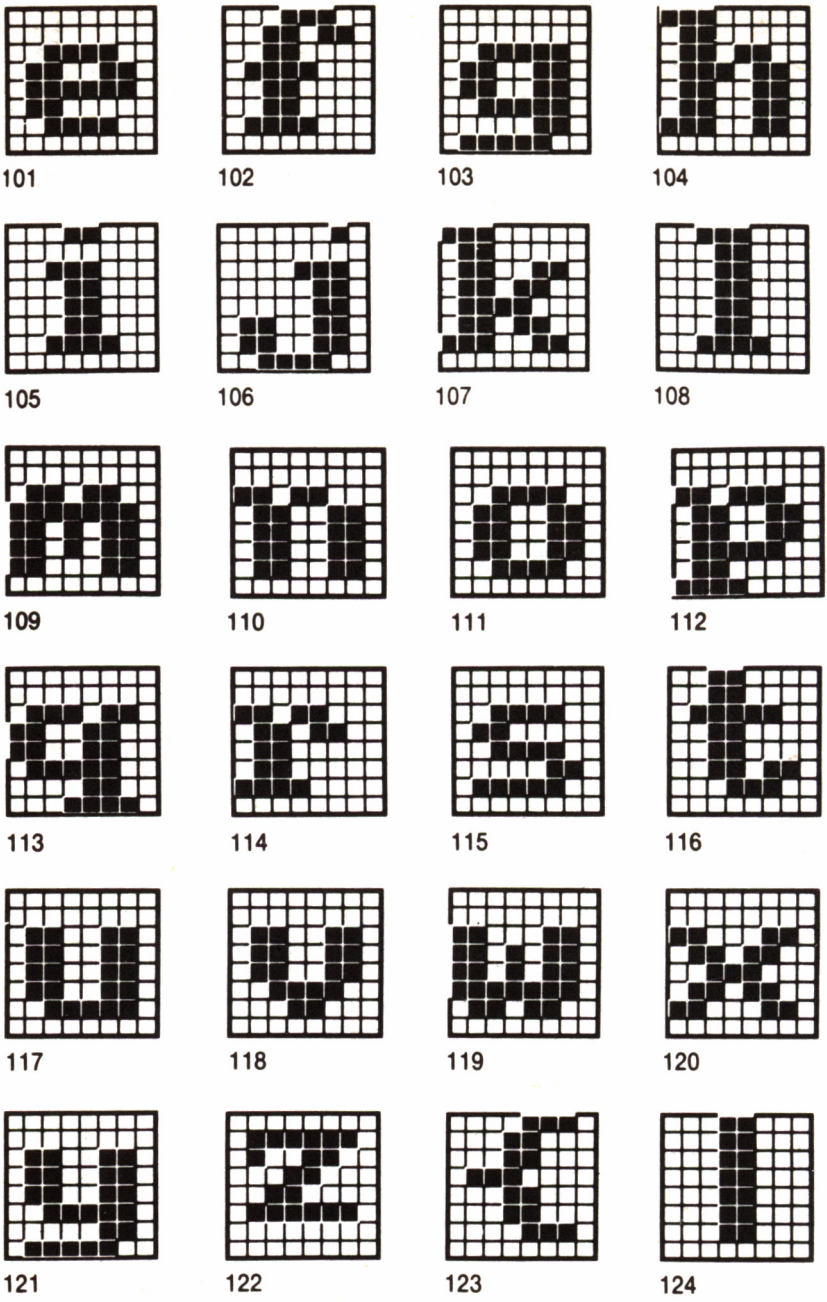


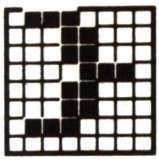
75



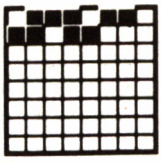
76


77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

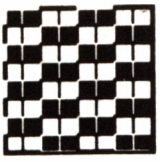

101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124



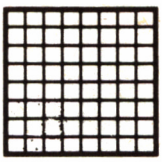
125



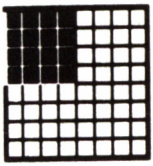
126



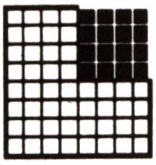
127



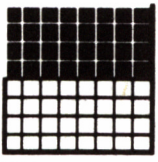
128



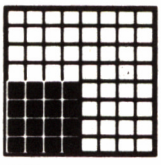
129



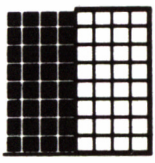
130



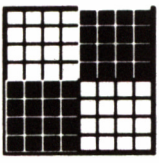
131



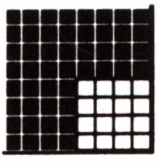
132



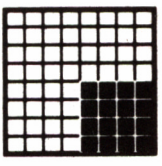
133



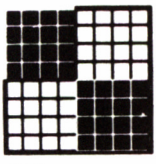
134



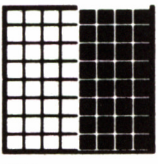
135



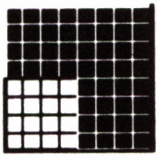
136



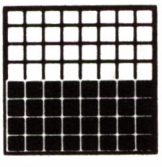
137



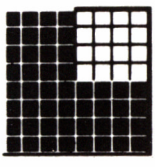
138



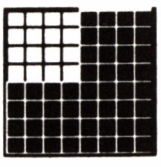
139



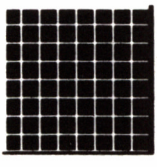
140



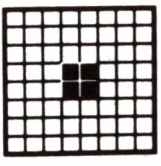
141



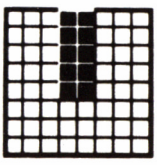
142



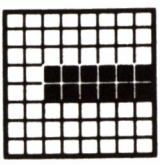
143



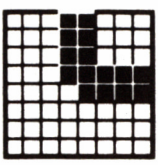
144



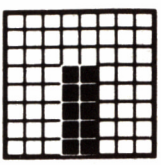
145



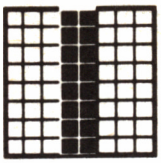
146

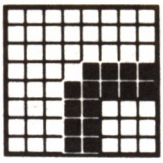


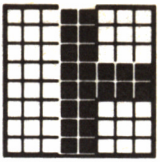
147

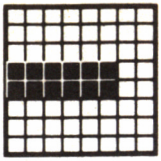


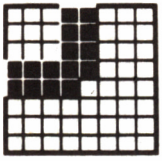
148

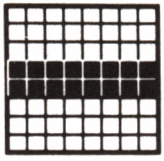

149

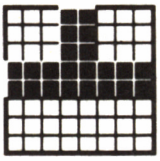

150

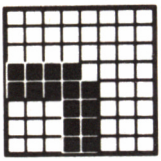

151

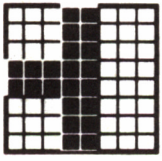

152

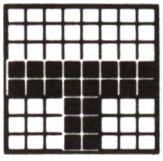

153

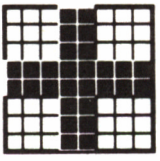

154

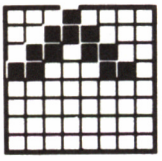

155

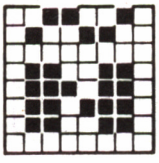

156

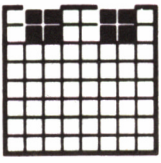

157

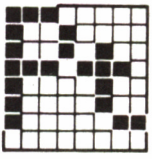

158

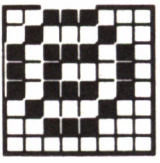

159

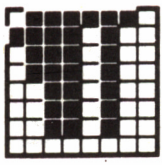

160

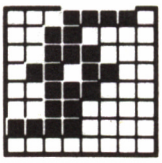

161

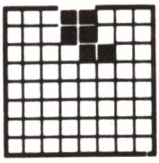

162

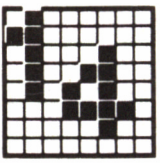

163

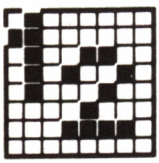

164

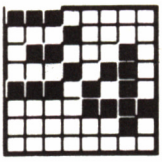

165

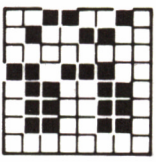

166

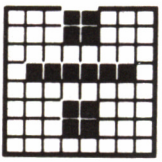

167

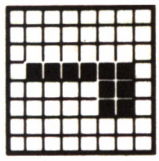

168


169

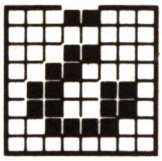

170


171

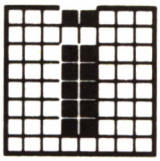

172



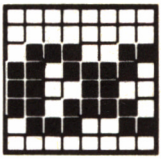
173



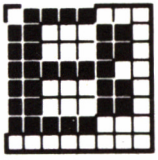
174



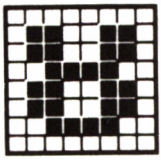
175



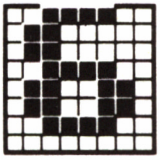
176



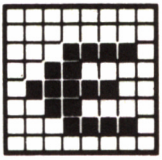
177



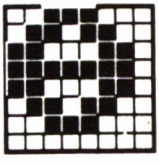
178



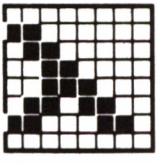
179



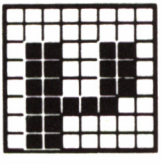
180



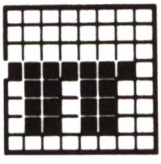
181



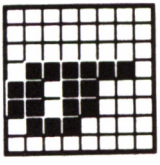
182



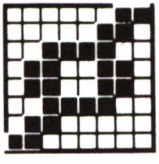
183



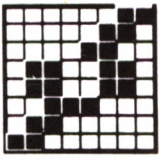
184



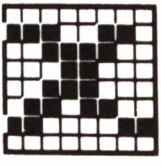
185



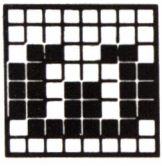
186



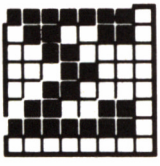
187



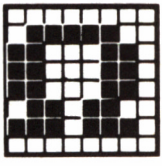
188



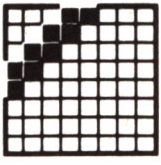
189



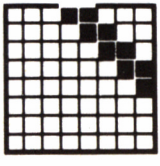
190



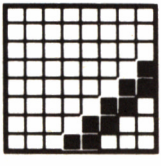
191



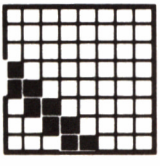
192



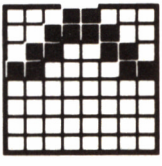
193



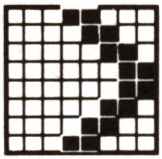
194

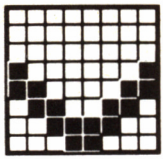


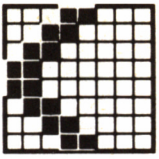
195

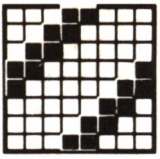


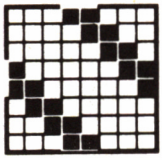
196

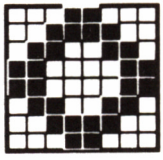

197

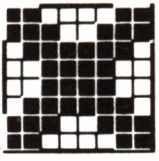

198

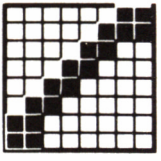

199

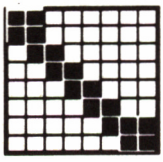

200

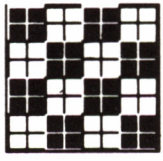

201


202

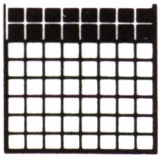

203

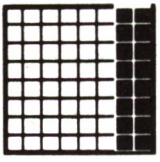

204

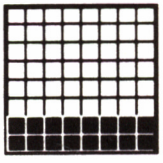

205

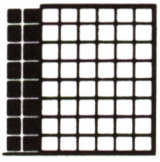

206

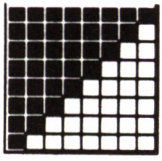

207

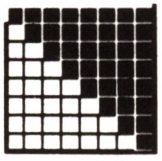

208

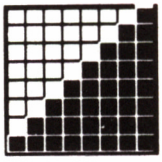

209

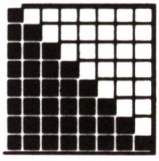

210

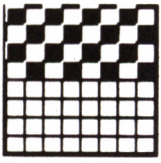

211

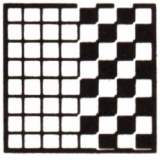

212

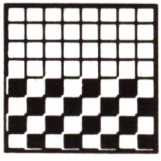

213

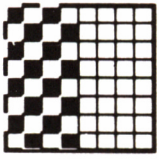

214

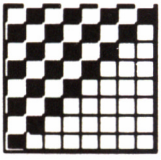

215

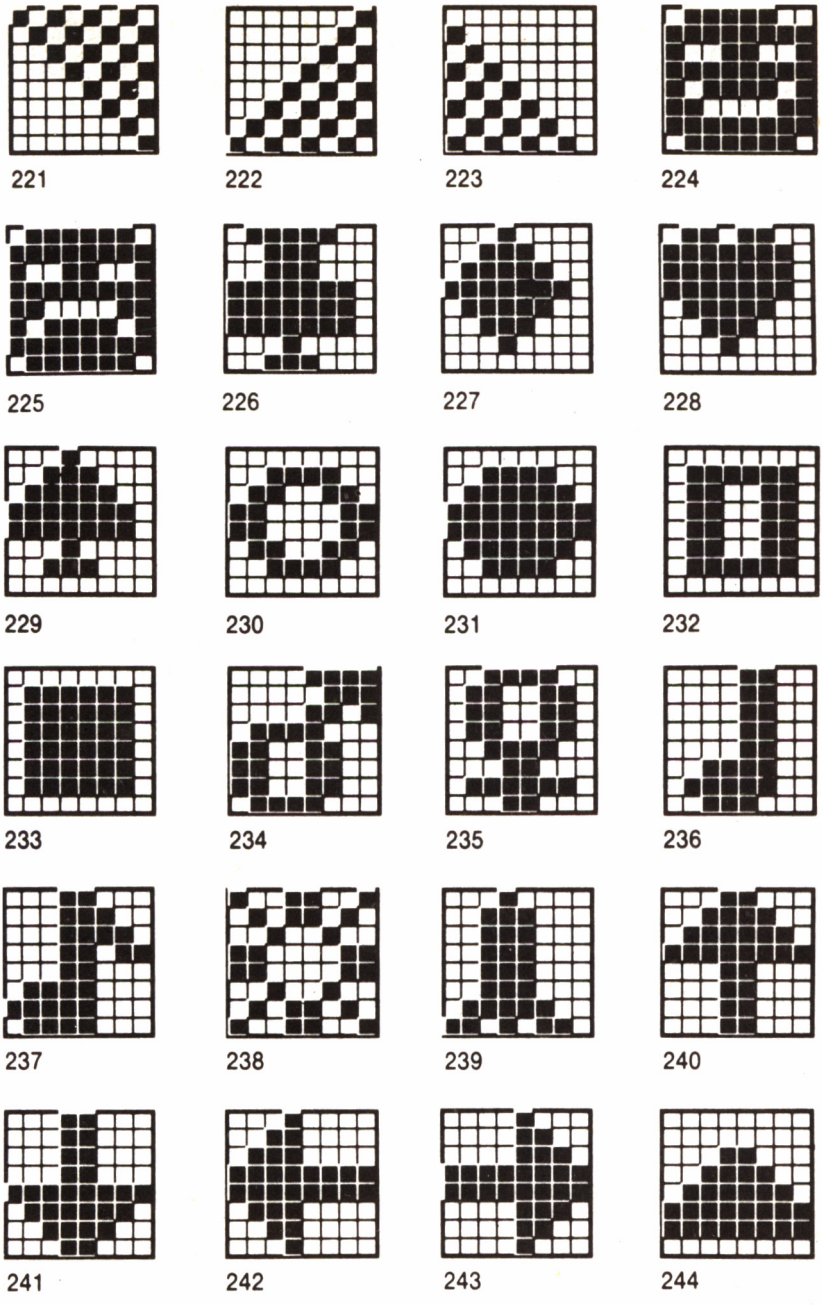

216

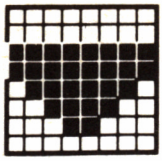

217


218

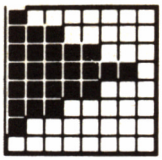

219


220

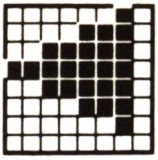

221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244



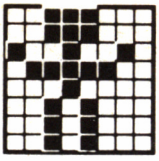
245



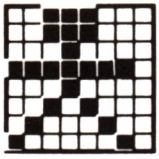
246



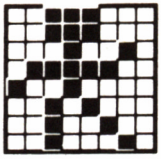
247



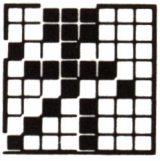
248



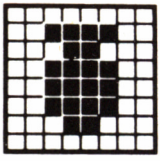
249



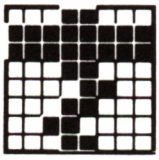
250



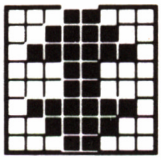
251



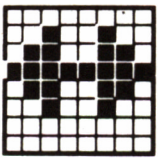
252



253



254



255

No obstante, un ordenador no es capaz de memorizar cuadrículas en su interior. Tan sólo puede almacenar números en su memoria y, por tanto, tendremos que transformar en números dicho gráfico. Para ello, la máquina utiliza un sistema de numeración denominado BINARIO.

Desde luego, no es necesario que sepamos nada sobre el código binario, aunque sería conveniente que tuviéramos algunas nociones sobre ello.

Tomemos nuestro gráfico y dividámoslo en ocho líneas horizontales. Cada una de estas líneas se va a convertir en un número comprendido entre 0 y 255. Va a ser esta serie de ocho números la que el ordenador almacenará en su memoria, para después poder representar su gráfico en la pantalla.

Para convertir cada una de estas líneas en un número, imaginemos que cada una de las ocho celdas que componen una línea es un cero o un uno. Si la celdilla está rellena, es un uno; si por el contrario está en blanco, es un cero. De esta manera, transformamos ocho líneas de puntos en ocho números binarios.

Ya tenemos el gráfico. Con esta serie de ocho números tenemos los suficientes datos como para poder definir nuestro propio carácter

gráfico. Para ello, haremos uso del comando BASIC SYMBOL. Esta instrucción redefine la forma de un carácter.

SYMBOL <número del carácter>,<serie de números>

El primer parámetro indica el número ASCII del carácter que deseamos reemplazar, el cual deberá ser mayor o igual a 240, y menor o igual a 255. A continuación, irá la lista de los ocho números binarios obtenidos en la operación anterior, separados por comas, anteponiendo a cada uno de ellos los símbolos &X. Estos símbolos se utilizan para indicar al ordenador que el número está en código binario, y él mismo se encargará de transformarlo al sistema decimal. Si lo deseamos, podemos transformar nosotros mismos estos números y teclearlos en el ordenador directamente en este sistema. Para ello, sólo deberemos pedirle al ordenador que nos indique en decimal el número binario. Así, por ejemplo, para saber el valor decimal del número binario 10101010 sólo tendremos que efectuar:

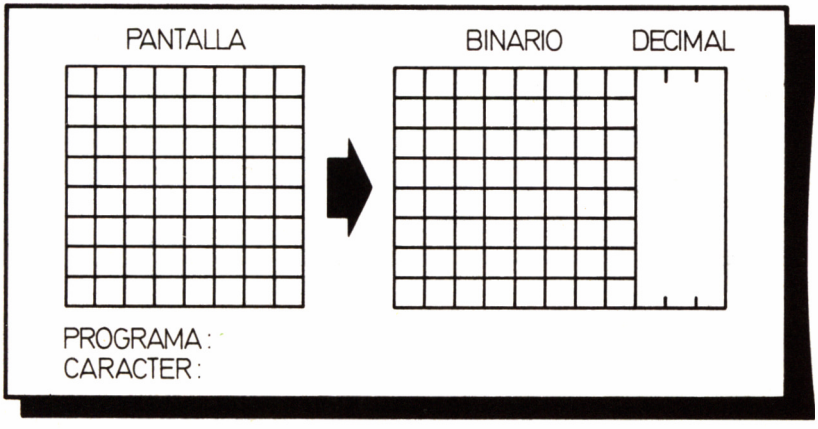
PRINT &X10101010

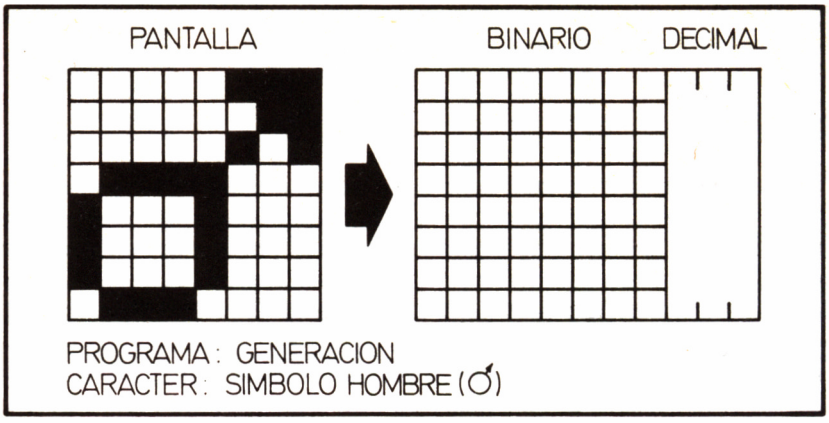
a lo cual el ordenador responderá 170.

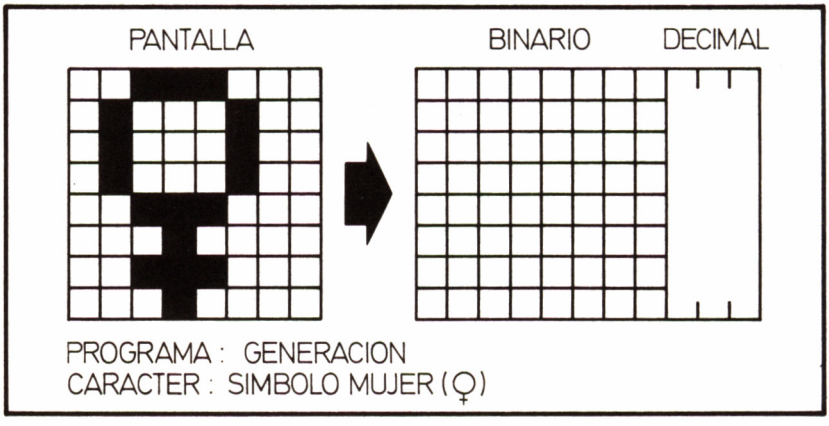
Para ver el resultado de la transformación llevada a cabo con todas estas operaciones, tecleemos:

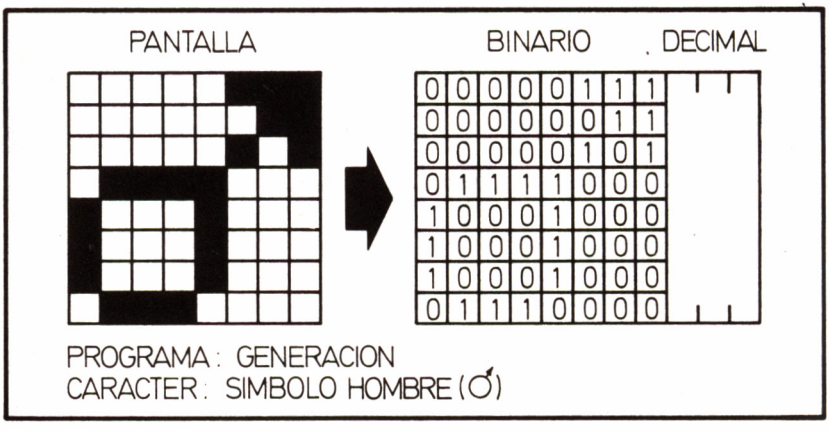
PRINT CHR$(<número del carácter>)

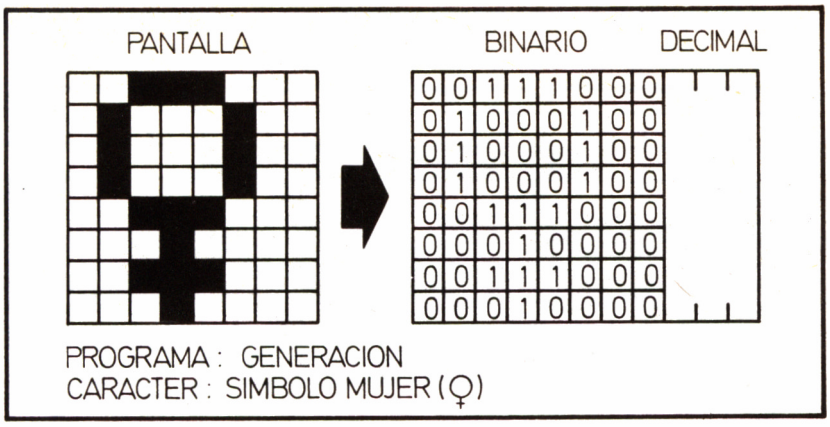
y nuestro propio dibujo aparecerá en la pantalla.

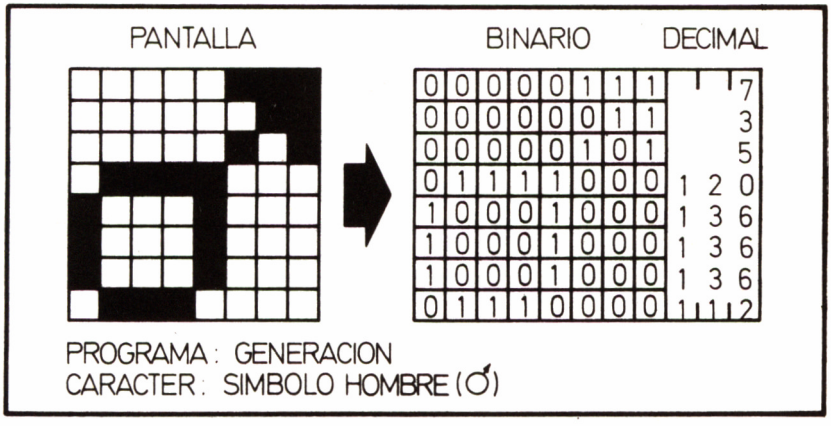


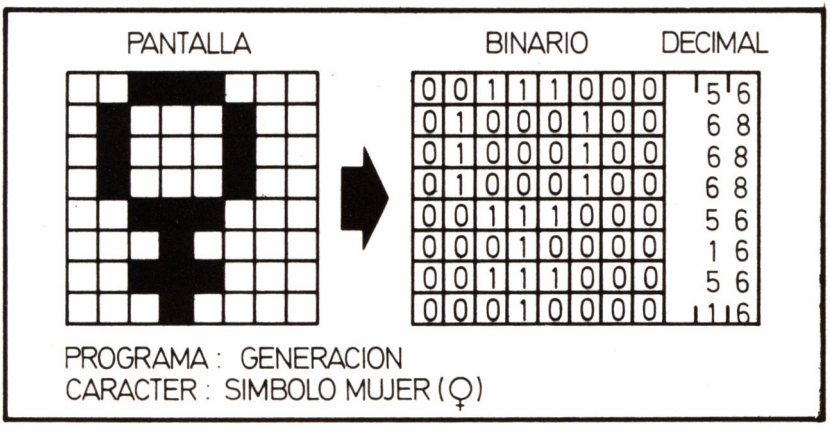

PANTALLA
BINARIO
DECIMAL
PROGRAMA : GENERACION
CARACTER : SIMBOLO HOMBRE (♂)

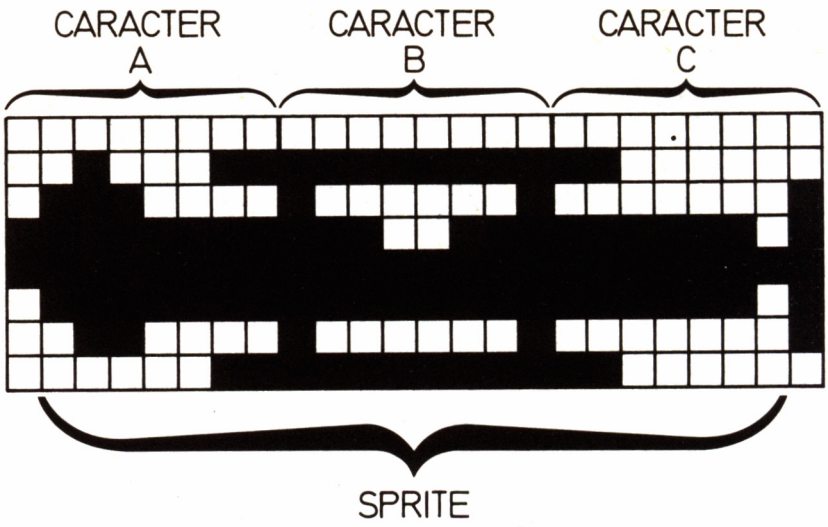

PANTALLA
BINARIO
DECIMAL
PROGRAMA : GENERACION
CARACTER : SIMBOLO MUJER (♀)


PANTALLA
BINARIO
DECIMAL
0 0 0 0 0 1 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 1 0 1
0 1 1 1 1 0 0 0
1 0 0 0 1 0 0 0
1 0 0 0 1 0 0 0
1 0 0 0 1 0 0 0
0 1 1 1 0 0 0 0
PROGRAMA : GENERACION
CARACTER : SIMBOLO HOMBRE (♂)


PANTALLA
BINARIO
DECIMAL
0 0 1 1 1 0 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 1 0 0
0 0 1 1 1 0 0 0
0 0 0 1 0 0 0 0
0 0 1 1 1 0 0 0
0 0 0 1 0 0 0 0
PROGRAMA : GENERACION
CARACTER : SIMBOLO MUJER (♀)


PANTALLA
BINARIO
DECIMAL
0 0 0 0 0 1 1 1 7
0 0 0 0 0 0 1 1 3
0 0 0 0 0 1 0 1 5
0 1 1 1 1 0 0 0 1 2 0
1 0 0 0 1 0 0 0 1 3 6
1 0 0 0 1 0 0 0 1 3 6
1 0 0 0 1 0 0 0 1 3 6
0 1 1 1 0 0 0 0 1 1 2
PROGRAMA : GENERACION
CARACTER : SIMBOLO HOMBRE (♂)


PANTALLA
BINARIO
DECIMAL
0 0 1 1 1 0 0 0 5 6
0 1 0 0 0 1 0 0 6 8
0 1 0 0 0 1 0 0 6 8
0 1 0 0 0 1 0 0 6 8
0 0 1 1 1 0 0 0 5 6
0 0 0 1 0 0 0 0 1 6
0 0 1 1 1 0 0 0 5 6
0 0 0 1 0 0 0 0 1 6
PROGRAMA : GENERACION
CARACTER : SIMBOLO MUJER (♀)



Si deseamos transformar algún carácter cuyo código sea menor que 240, bastará con que antes hayamos tecleado la instrucción SYMBOL AFTER, la cual le indica al ordenador a partir de qué número podemos transformar los gráficos de caracteres. Por ejemplo, escribiendo:

SYMBOL AFTER 32

el ordenador nos permitirá cambiar cualquier carácter cuyo código esté comprendido entre los números 32 y 255. SYMBOL AFTER 256 impide la redefinición de cualquier carácter.

Definiendo Nuestros Propios Caracteres

Como podemos comprobar, este sistema para redefinir caracteres gráficos es muy útil, pero también muy pesado. Puede resultar rentable cuando sólo deseamos transformar uno o dos gráficos, pero ¿y cuando se trata de muchos?

El ordenador pone a prueba nuestra paciencia y resistencia en hacer dibujos en un papel cuadriculado y teclear cientos de números, para después comprobar que nos hemos equivocado en unos cuantos y empezar de nuevo todo el proceso.

Para evitar todo este engorroso trabajo, existen programas como el que presentamos a continuación: KARACTEKA.

CÓMO SE UTILIZA KARACTEKA

KARACTEKA es un programa que ejecuta todos y cada uno de los pasos mencionados anteriormente, incluyendo el de representar en la pantalla una cuadrícula de ocho por ocho celdillas, en la cual podremos hacer nuestros gráficos sin necesidad de rellenar páginas y páginas de interminables pruebas.

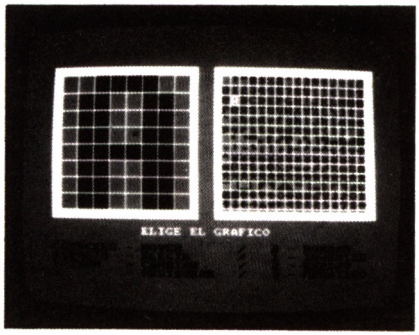
Además, aparecen también en el monitor todos los caracteres gráficos comprendidos entre el 32 y el 255, es decir, todos los que la máquina permite redefinir.

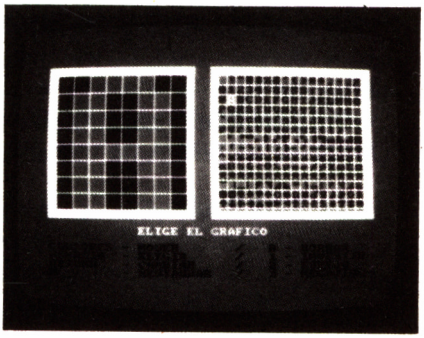
KARACTEK

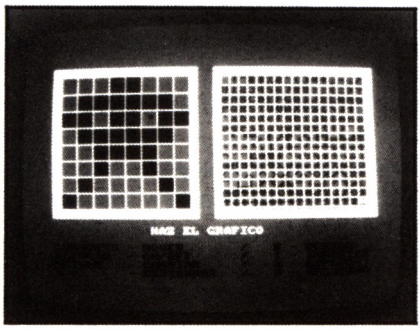
```
1 'PROGRAMA KARACTEKA DE CREACION DE CARACTERES GRAFICOS
2 'DANIEL LOZANO VALVERDE
3 'RAFAEL DE LA OSSA VILLACA\AS
5 'Inicializa programa
10 MODE 1:BORDER 13:INK 0,13:INK 1,0:INK 3,24:PAPER 0:PEN 2:CLS
20 ORIGIN 16,128:SYMBOL AFTER 32
30 DIM g(8,8):x=1:y=1:ret=0
40 WINDOW #1,2,17,2,18:WINDOW #2,22,39,2,17:WINDOW #3,1,40,20,25
45 'Dibuja pantalla
50 LOCATE 1,1:PRINT STRING$ (18,143);STRING$(2,32);STRING$(20,143):LOC
ATE 1,18:PRINT STRING$(18,143);STRING$(2,32);STRING$(20,143)
60 RESTORE 60:FOR i=1 TO 4:READ a:FOR j=1 TO 18:LOCATE a,j:PRINT CHR$(
143):NEXT j:NEXT i:DATA 1,18,21,40
70 LOCATE #3,12,1:PEN #3,3:PRINT#3," HAZ EL GRAFICO":LOCATE #3,1,3:PEN
 #3,1
80 PRINT #3,"CURSORES - MOVER       /   B - BORRAR   ESPACIO  - ON/OFF
      /   1 - INVERTIR RETURN   - CAMBIAR     /   2 - ESPEJO   R
  - RESTAURAR   /   3 - NEGATIVO"
90 GOSUB 740:GOSUB 750:PEN 1:GOTO 690
95 'Investiga teclado
100 A$=INKEY$:IF A$="" THEN GOTO 100 ELSE A=ASC(A$)
110 IF A=32 THEN 210
120 IF A=49 THEN 370
130 IF A=50 THEN 400
140 IF A=51 THEN 430
150 IF A=13 THEN 330
160 IF A=66 OR A=98 THEN 360
165 IF A=67 OR A=99 THEN GOTO 850
170 IF A=82 OR A=114 THEN 280
175 IF A=71 OR A=103 THEN 780
180 IF A=240 THEN 460
185 IF A=241 THEN 490
190 IF A=242 THEN 520
195 IF A=243 THEN 550
200 GOTO 100
205 'Rutina del espacio
210 IF RET=1 THEN GOTO 250
220 IF G(X,Y)=1 THEN G(X,Y)=0:H$="  ":GOTO 240
230 G(X,Y)=1:H$=STRING$(2,CHR$ (143))
240 GOSUB 730:MOVE (X-1)*32,(8-Y)*32:DRAWR 0,32,2:MOVE (X-1)*32,(8-Y)*
```

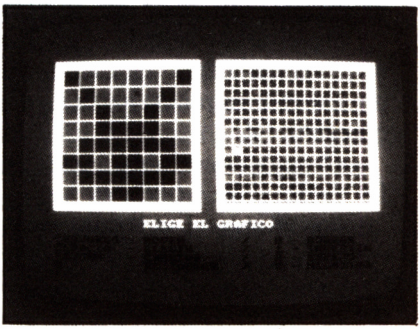
```
32:DRAWR 32,0,2:GOTO 690
250 FOR I=1 TO 8:B(I)=128*G(1,I)+64*G(2,I)+32*G(3,I)+16*G(4,I)+8*G(5,I
)+4*G(6,I)+2*G(7,I)+G(8,I):NEXT I
260 SYMBOL NUM,B(1),B(2),B(3),B(4),B(5),B(6),B(7),B(8)
270 ORIGIN 336,383:TAG:MOVE xr,yr,3:PRINT CHR$(NUM);:TAGOFF:ORIGIN 16,
128:GOTO 100
275 'Rutina de restaurar
280 LOCATE #3,12,1:PEN #3,3:PRINT #3," ESTAS SEGURO ? "
290 A$=INKEY$:IF A$="" THEN GOTO 290
300 IF A$="S" OR A$="s" THEN SYMBOL AFTER 32:CLS #2:GOSUB 750
310 LOCATE #3,12,1:PEN #3,3:IF RET=0 THEN PRINT #3," HAZ EL GRAFICO "
ELSE PRINT #3,"ELIGE EL GRAFICO"
320 GOTO 100
325 'Rutina del cambio de pantalla(enter)
330 IF RET=0 THEN GOTO 350
340 RET=0:LOCATE #3,12,4:PEN #3,1:PRINT #3,"ON/OFF":GOTO 310
350 RET=1:LOCATE #3,12,4:PEN #3,1:PRINT #3,"ELEGIR":GOTO 310
355 'Rutina de borrar
360 ERASE G:DIM G(8,8):CLS #1:LOCATE 1,18:PEN 2:PRINT STRING$(18,143):
PEN 1:X=1:Y=1:GOSUB 740:GOTO 690
365 'Rutina de invertir
370 FOR I=1 TO 8:FOR J=1 TO 4
380 IN=G(I,J):G(I,J)=G(I,9-J):G(I,9-J)=IN
390 NEXT J:NEXT I:GOTO 700
395 'Rutina del espejo
400 FOR J=1 TO 8:FOR I=1 TO 4
410 IN=G(I,J):G(I,J)=G(9-I,J):G(9-I,J)=IN
420 NEXT I:NEXT J:GOTO 700
425 'Rutina del negativo
430 FOR I=1 TO 8:FOR J=1 TO 8
440 IF G(I,J)=1 THEN G(I,J)=0 ELSE G(I,J)=1
450 NEXT J:NEXT I:GOTO 700
455 'Cursor 1 arriba
460 IF RET=1 THEN GOTO 580
470 IF Y=1 THEN GOTO 100
480 GOSUB 680:Y=Y-1:GOTO 690
485 'Cursor 1 abajo
490 IF RET=1 THEN GOTO 600
500 IF Y=8 THEN GOTO 100
510 GOSUB 680:Y=Y+1:GOTO 690
515 'Cursor 1 izquierda
520 IF RET=1 THEN GOTO 620
530 IF X=1 THEN GOTO 100
540 GOSUB 680:X=X-1:GOTO 690
545 'Cursor 1 derecha
550 IF RET=1 THEN GOTO 640
560 IF X=8 THEN GOTO 100
570 GOSUB 680:X=X+1:GOTO 690
575 'Cursor 2 arriba
580 IF YR=0 THEN GOTO 100
590 GOSUB 660:YR=YR+18:GOSUB 670:GOTO 100
595 'Cursor 2 abajo
600 IF YR=-234 THEN GOTO 100
610 GOSUB 660:YR=YR-18:GOSUB 670:GOTO 100
615 'Cursor 2 izquierda
620 IF XR=0 THEN GOTO 100
630 GOSUB 660:XR=XR-18:GOSUB 670:GOTO 100
635 'Cursor 2 derecha
640 IF XR=270 THEN GOTO 100
650 GOSUB 660:XR=XR+18:GOSUB 670:GOTO 100
655 'Coloca el caracter en amarillo
660 ORIGIN 336,383:TAG:MOVE XR,YR,1:PRINT CHR$(NUM);:RETURN
665 'Coloca el caracter en amarillo y halla NUM
670 NUM=(YR/-18)*16+XR/18+32:MOVE XR,YR,3:PRINT CHR$(NUM);:TAGOFF:ORIG
IN 16,128:RETURN
675 'Borra el cursor
680 LOCATE #1,X*2,Y*2-1:PEN #1,1:PRINT #1,CHR$(128+G(X,Y)*15):RETURN
685 'Pone el cursor
690 LOCATE #1,X*2,Y*2-1:PEN #1,1:PRINT #1,CHR$(132+g(x,y)*7):GOTO 100
695 'Subrutina de dibujar matriz
700 FOR Y=1 TO 8:FOR X=1 TO 8
710 IF G(X,Y)=1 THEN H$=STRING$(2,CHR$(143)) ELSE H$="  "
720 GOSUB 730:NEXT X:NEXT Y:GOSUB 740:X=1:Y=1:GOTO 690
```

```
730 LOCATE #1,X*2-1,Y*2-1:PEN #1,1:PRINT#1,H$:LOCATE #1,X*2-1,Y*2:PRIN
T #1,H$:RETURN
735 'Cuadricula 1 pantalla
740 FOR I=0 TO 224 STEP 32:MOVE I,0,2:DRAWR 0,256:NEXT I:FOR I=0 TO 22
4 STEP 32:MOVE 0,I,2:DRAWR 256,0:NEXT I:RETURN
745 'Cuadricula 2 pantalla, pone los caracteres e inicializa el cursor
 2
750 ORIGIN 336,383:FOR I=16 TO 288 STEP 18:MOVE I,0,2:DRAWR 0,-256:NEX
T I:FOR I=-16 TO -256 STEP -18:MOVE 0,I,2:DRAWR 288,0:NEXT I
760 E=32:TAG:FOR I=0 TO -238 STEP -18:FOR J=0 TO 272 STEP 18:MOVE J,I,
1:PRINT CHR$ (E);:E=E+1:NEXT J:NEXT I
770 XR=270:YR=-234:GOTO 670
780 LOCATE #3,12,1:PEN #3,3:INPUT #3,"NOMBRE DEL FICHERO";N$
785 OPENOUT N$:FOR E=32 TO 255
790 WRITE #9, CHR$(E)
800 NEXT E:CLOSEOUT
810 GOSUB 750:GOTO 100
850 LOCATE #3,12,1:PEN #3,3:INPUT #3,"NOMBRE DEL FICHERO";N$
860 OPENIN N$:FOR E=32 TO 255:A$=CHR$(E)
870 INPUT #9, A$
880 NEXT E:CLOSEIN
890 GOSUB 750:GOTO 100
```









Una vez que haya sido tecleado, grabado en cinta o disco, y ejecutado mediante RUN, aparecerán en la pantalla dos ventanas. En la derecha observaremos la lista completa de caracteres gráficos antes mencionada. La ventana de la izquierda sirve para hacer el gráfico.

Como podemos comprobar, esta última ventana es una cuadrícula de ocho por ocho casillas, por las cuales nos podremos mover haciendo uso de los cursores. Pulsando la barra espaciadora en una celdilla vacía, ésta quedará inmediatamente rellena. Pulsándola en una rellena, se vaciará automáticamente.

De esta forma, y con un poco de imaginación, se puede construir cualquier gráfico que deseemos. Una vez terminado el carácter, pulsando la tecla RETURN, el cursor pasará a la ventana de la derecha. Allí, manejando de nuevo los cursores, podremos elegir el gráfico que queremos sustituir por el nuestro. Podemos pulsar la barra espaciadora tantas veces como queramos, y conseguir tantos gráficos iguales como deseemos.

Tanto si ya hemos elegido el gráfico sustituido, como si no, pulsando de nuevo RETURN apareceremos otra vez en la ventana de definición.

Para facilitar la construcción de nuestros propios caracteres, se incluyen en el programa cuatro funciones especiales:

— BORRAR (B).
— INVERTIR (1).
— ESPEJO (2).
— NEGATIVO (3).

Pulsando B, el ordenador borrará la ventana de la izquierda, perdiéndose lo dibujado hasta ese momento, a no ser que ya hayamos definido el carácter en la ventana de la derecha.

La función invertir, la cual se ejecuta pulsando 1, pone boca abajo el carácter dibujado en la ventana.

Si pulsamos la tecla 2, el ordenador le dará la vuelta al gráfico que estemos realizando en ese momento. Es decir, si el gráfico visualiza a un hombre mirando hacia la derecha, pulsando 2 mirará hacia la izquierda.

El negativo cambiará cada celdilla de la cuadrícula por su inversa. Es decir, si la celdilla es negra, la convertirá en blanca y viceversa.

Por último, si pulsamos la tecla R, el ordenador nos preguntará si estamos seguros de nuestra decisión. Tengamos mucho cuidado al utilizar esta función, pues si contestamos «s» o «S» la máquina volverá a definir por sí misma el juego completo de caracteres original, debido

a lo cual se habrán perdido todos los gráficos confeccionados hasta el momento.

FUNCIONAMIENTO DEL PROGRAMA

Estudiemos ahora el funcionamiento del programa.

5-40: Inicializa el programa eligiendo el modo de pantalla y los colores de las plumas. Fija el centro de coordenadas en (16,128) y define el campo de posibilidades de transformación de caracteres gráficos.

Dimensiona la variable G, en la cual se almacenará el gráfico que vayamos construyendo. X e Y son las coordenadas del cursor dentro de la propia matriz G, es decir, indican en qué celda de la ventana de la izquierda nos encontramos. Inicializa el indicador RET, que sólo toma los valores 0 y 1, estando a cero cuando nos encontremos en la ventana izquierda y estando a uno cuando nos encontremos en la ventana derecha.

. Define tres ventanas (WINDOW). La primera corresponde a la ventana del gráfico, la segunda a la ventana de los caracteres gráficos y la tercera es la ventana inferior, donde aparece el menú de funciones.

45-90: Dibuja los marcos de las ventanas superiores, coloca los mensajes del menú y traza los entramados de ambas ventanas, colocando posteriormente la lista de los caracteres que podemos redefinir.

95-200: Investiga el teclado, poniendo en marcha la rutina correspondiente a cada tecla pulsada. Este es el bucle principal.

205-270: Rutina del espacio. Pregunta en qué ventana está el usuario en ese momento. En caso de encontrarse en la ventana del gráfico, apaga el punto si está encendido y lo enciende si está apagado, tanto en la matriz como en la pantalla. Cuando una casilla se vacía o rellena, se borra parte del cuadriculado existente, el cual es restituido por esta rutina (línea 240).

En caso de encontrarnos en la ventana derecha, con un bucle calcula los números decimales correspondientes al gráfico definido anteriormente en la otra ventana y lo almacena en la memoria del ordenador. Cambia el origen de coordenadas, colocando el nuevo carácter en su lugar correspondiente en la ventana de caracteres gráficos.

275-320: Pregunta al usuario si está seguro de restaurar el juego completo de caracteres, perdiendo lo definido hasta el momento. En caso afirmativo, restaura el juego de caracteres y regresa al bucle principal.

325-350: Cambia de ventana. Cuando el cursor se encuentra en la ventana izquierda, aparece la orden «HAZ EL GRAFICO», y en el menú el espacio toma la función ON/OFF. En caso de encontrarnos en la ventana derecha, aparece la orden «ELIGE EL GRAFICO» y el espacio toma la función ELEGIR.

355-360: Borra la matriz G e inicializa la ventana de gráficos, perdiéndose el gráfico de dicha ventana y volviendo al bucle principal.

365-390: Coloca el gráfico boca abajo, intercambiando los valores de la matriz de arriba a abajo, para después ir a la rutina de dibujar matriz.

395-420: Le da la vuelta al carácter intercambiando los valores de la matriz de izquierda a derecha, para ir después a la rutina de dibujar matriz.

425-450: Coloca el carácter en negativo, poniendo a cero los valores de la matriz que sean iguales a uno y viceversa. Posteriormente, va a la rutina de dibujar matriz.

455-560: Se encargan de mover el cursor cuando nos encontremos en la ventana izquierda.

575-650: Se encargan de mover el cursor cuando nos encontremos en la ventana derecha.

655-670: Estando en la pantalla de caracteres gráficos (ventana derecha), coloca un carácter sobre el que está el cursor en amarillo y calcula el código decimal de ese carácter (NUM).

675-690: Se encargan de borrar o de poner el cursor de la ventana izquierda, para posteriormente volver al bucle principal.

695-730: Dibujan el carácter en la pantalla de dibujo, tomando los datos almacenados en la matriz G. Va comprobando dato por dato de dicha matriz, acudiendo a las líneas 680 ó 690, según esté el punto encendido (valor 1 en la matriz) o apagado (valor 0 en la matriz).

735-745: Dibujan las cuadrículas interiores de las dos ventanas. La primera subrutina dibuja el cuadriculado de la ventana izquierda. La segunda subrutina ejecuta lo mismo en la ventana derecha.

760-770: Subrutina que representa en la ventana izquierda el estado o apariencia actual de todos los caracteres gráficos desde el 32 hasta el 255.

GRÁFICOS, ESTADÍSTICA Y NEGOCIOS

os gráficos de ordenador se utilizan ampliamente en el campo de los negocios, como ayuda a los estudios de márketing, análisis financieros y planificación. Los gráficos se emplean para mostrar tablas y cuadros de datos, tales como ventas, pagos y fluctuaciones.

ANTE TODO, LA PRESENTACIÓN

Para una persona, la información es más rápidamente asimilable de forma gráfica (una imagen vale más que mil palabras) y, además, un dibujo es siempre más grato de observar que una tira interminable de datos ininteligibles y aparentemente absurdos.

Existen tres tipos de técnicas de gráficas en el campo de los negocios y la estadística. La primera supone la exposición de datos en forma de círculo seccionado. Otra es la típica línea que solemos ver en todas las empresas saliéndose por la parte baja del cuadro y, por último, están los gráficos de barras.

Son muchos los programas de este tipo que podemos adquirir en el amplio mercado del software, pero afortunadamente, la técnica de los gráficos de empresa es realmente simple. Además, el usuario de Amstrad está de suerte, pues la pantalla de su ordenador posee un modo de ochenta columnas, el cual admite gráficos de negocios profesionales.

GRÁFICOS DE PASTELERO

Ya sabemos cómo dibujar un círculo en la pantalla, lo cual es muy importante en el programa PASTEL. Los datos en un gráfico de pastel son expresados como una proporción de la suma de todos ellos.

Veamos un ejemplo para entenderlo mejor. Tenemos los datos de las ventas de una hipotética empresa exportadora de coles de bruselas a varias ciudades europeas. Veamos las ventas en los puntos donde se distribuye el producto:

LUGAR	N.º TONELADAS
PARIS	1.800
LONDRES	800
MADRID	1.550
BRUSELAS	50
TOTAL	4.200

Estos datos nos indican que el pastel estará dividido en cuatro trozos o sectores, una por cada ciudad. Las ventas en cada capital serán representadas en el pastel como una proporción de las ventas totales. Por ejemplo, Londres ha comprado el

$$\frac{800}{4.200} \times 100 = 19{,}04\ \%$$

La primera misión del programa PASTEL es introducir los datos, sumarlos y totalizarlos, para trabajar con las proporciones con respecto al total. Pero ¿cómo hallaremos qué parte de curva del círculo corresponde a cada dato?

Como seguramente sabremos, un gráfico de pastel consiste en un número de líneas que nacen del centro de la circunferencia y mueren en alguna parte de su periferia. Cada línea forma un determinado

ángulo con la otra. Este ángulo entre líneas es crucial; así pues, tomemos ahora los porcentajes y pasémoslos a cantidad de grados de la circunferencia:

$$\frac{19{,}04\ \%}{100} \times 360 \text{ grados} = 68 \text{ grados}$$

Esta es una de las equivalencias porcentaje-ángulo que el programa deberá calcular. Ahora necesitamos un método para calcular las líneas radiales, para lo cual utilizaremos las mismas ecuaciones que para construir una circunferencia.

Cada punto de la circunferencia posee dos coordenadas X e Y, definidas por:

```
X=X1 +(RADIO*COS(ANGULO))
Y=Y1 +(RADIO*SIN(ANGULO))
```

donde X1 e Y1 son las coordenadas del centro del círculo. Las líneas que definen el primer trozo del pastel son:

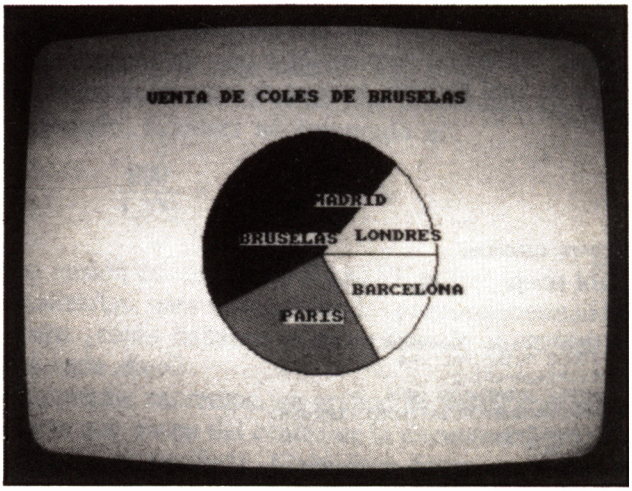
```
MOVE X1,Y1
DRAW X,Y
MOVE X1,Y1
DRAW X2,Y2
```

El pastel resulta mucho más vistoso y útil con cuatro colores que con los dos a los que nos deberíamos limitar empleando el MODO 2 de pantalla. Utilizamos, pues, el modo 1, gracias a lo cual podremos escribir mensajes en la pantalla no mayores de cuarenta caracteres. Podemos, sin embargo, emplear el modo de pantalla que más nos plazca, pues es fácil hacerlo, aunque es muy recomendable el elegido por nosotros.

PASTEL

```
10 REM DIAGRAMA CIRCULAR
20 REM DATOS
30 CLS: INK 0,13: INK 1,0:INK 2,3:INK 3,7
40 BORDER 13
50 MODE 2
60 PRINT"BIENVENIDO AL PROGRAMA PASTEL"
70 INPUT "TITULO PRINCIPAL";P$
80 P1=LEN(P$):P1=20-(P1/2)
90 INPUT"CUANTOS SEGMENTOS";NUM
100 DIM S(NUM),H$(NUM),POINT(NUM),CUM(NUM)
110 TOTAL=0:CANGLE=0
120 FOR I=1 TO NUM
130 INPUT"INTRODUCE EL TITULO DE SEGMENTO";H$(I)
```

```
140 INPUT"VALOR DEL SEGMENTO";S(I)
150 TOTAL=TOTAL+S(I)
160 NEXT I
170 MODE 1
180 FOR I=1 TO NUM:REM ANGULOS DE CADA SEGMENTO
190 CANGLE=CANGLE+((S(I)/TOTAL)*(2*PI))
200 POINT(I)=CANGLE-(((S(I)/2)/TOTAL)*(2*PI))
210 CUM(I)=CANGLE
220 NEXT I
230 CLS
240 LOCATE P1,1:PRINT P$
250 TAG
260 REM DIBUJA CIRCULO
270 RADIUS=150
280 XC=320:YC=200
290 A=(2*PI)/ 100
300 ANGLE=0
310 X2=XC+RADIUS:Y2=YC
320 FOR I=1 TO 100
330 ANGLE=ANGLE+A
340 X1=X2:Y1=Y2
350 X2=XC+RADIUS*COS(ANGLE)
360 Y2=YC+RADIUS*SIN(ANGLE)
370 MOVE X1,Y1
380 DRAW X2,Y2
390 NEXT I
400 REM DIBUJA LOS SEGMENTOS
410 N=-1
420 FOR I=1 TO NUM
430 N=N+1:IF N=4 THEN N=0
440 MOVE XC,YC
450 X1=XC+RADIUS*COS(CUM(I))
460 Y1=YC+RADIUS*SIN(CUM(I))
470 DRAW X1,Y1
480 X2=XC+(RADIUS/2)*COS(POINT(I))
490 Y2=YC+(RADIUS/2)*SIN(POINT(I))
500 DISP=LEN(H$(I))+15
510 MOVE X2,Y2
520 FILL N
530 IF X2<XC THEN DISP=DISP+10
540 MOVE X2-DISP,Y2
550 PRINT H$(I);
560 NEXT I
```



Observando el listado del programa, lo encontraremos muy fácil de comprender, pero destaquemos dos puntos: primeramente, hemos utilizado un comando FILL para rellenar cada sección, instrucción no implementada en el 464, y muy difícil de llevarla a cabo, si no es con una rutina en código máquina.

El segundo punto concierne al emplazamiento del texto. Este se coloca sobre el pastel utilizando el comando TAG. La posición de comienzo de cada texto está en una supuesta línea radial bisectriz del ángulo dado, a medio camino entre el centro del círculo y la circunferencia.

Observemos los datos del programa: son introducidos en la matriz S, junto con los nombres de cada segmento que entran a formar parte de la matriz H$. Los ángulos son calculados e introducidos en una nueva matriz llamada CUM. Esta contiene los valores del ángulo de cada segmento; si este empezara en la línea de cero grados, es decir, si por ejemplo tuviéramos que los tres primeros ángulos fueran 15, 40 y 20, en CUM quedarían almacenados como 15, 55 y 75, respectivamente. La matriz POINT contiene los ángulos bisectrices de cada segmento para la ubicación de textos.

¡MÁS DIFÍCIL TODAVÍA!

Exacto, podemos cambiar el programa PASTEL para obtener un gráfico todavía más complejo e informativo. Así crearemos el programa PASTEL 2, que se emplea para remarcar uno de los segmentos, «arrancándolo» de la circunferencia.

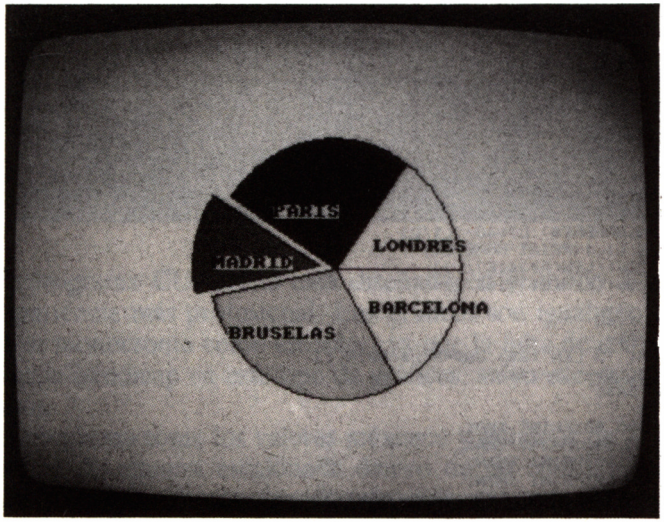
Imaginemos que el Sr. Rodríguez, empleado de la empresa de exportación de coles de bruselas antes reseñada, informa a su jefe con un gráfico de pastel de las ventas realizadas por cada uno de los empleados de la sección de ventas. Rodríguez remarcará su segmento y posiblemente haría que emitiera destellos si supiera algo más de programación.

Cada cual puede encontrar una utilidad a este tipo de gráfico, cuya misión es informar con mayor efectividad que el programa anterior. Más o menos, el programa se construye de la misma forma que PASTEL, pero necesitamos encontrar la línea bisectriz del segmento remarcado, y el centro del círculo ha de ser temporalmente cambiado a lo largo de esta línea.

PASTEL2

```
10 REM PROGRAMA PASTEL 2
20 REM DATOS
30 CLS:INK 0,13:INK 1,0:INK 2,6:INK 3,12
40 MODE 2
50 PRINT"****BIENVENIDO AL PROGRAMA PASTEL II****"
60 INPUT"NUMERO DE SEGMENTOS?";NUM
70 INPUT "SEGMENTO A SER SEPARADO";EX
80 IF EX=NUM THEN PRINT"ESE NO SE PUEDE SEPARAR":GOTO 70
90 DIM D(NUM),G$(NUM),PUN(NUM),C(NUM)
100 TOT=0:CA=0
110 FOR N=1 TO NUM
120 PRINT"TITULO DEL SEGMENTO";N;:INPUT G$(N)
130 INPUT"VALOR?";D(N)
140 TOT=TOT+D(N)
150 NEXT N
160 MODE 1
170 TAG
180 FOR N=1 TO NUM
190 CA=CA+((D(N)/TOT)*(2*PI))
200 PUN(N)=CA-(((D(N)/2)/TOT)*(2*PI))
210 C(N)=CA
220 NEXT N
230 CLS
240 REM DIBUJAMOS CIRCULO
250 RA=150
260 XR=320:YR=200
270 ANG=(2*PI)/300
280 ANGULO=0
290 X2=XR+RA:Y2=YR
300 NO1=0:NO2=0
310 FOR N=1 TO 300
320 ANGULO=ANGULO+ANG
330 X1=X2:Y1=Y2
340 IF ANGULO>=C(EX-1) AND NO1=0 THEN GOSUB 630
350 IF ANGULO>=C(EX) AND NO2=0 THEN MOVE XR,YR:DRAW X1,Y1
360 IF ANGULO>=C(EX) AND NO2=0 THEN XR=EQ1:YR=EQ2
370 X2=XR+RA*COS(ANGULO)
380 Y2=YR+RA*SIN(ANGULO)
390 IF ANGULO>=C(EX-1) AND NO1=0 THEN X1=X2:Y1=Y2
400 IF ANGULO>=C(EX-1) AND NO1=0 THEN MOVE XR,YR:DRAW X2,Y2:NO1=1
410 IF ANGULO>=C(EX) AND NO2=0 THEN ANGULO=ANGULO-ANG:X1=X2:Y1=Y2:NO2=
1:GOTO 330
420 MOVE X1,Y1
430 DRAW X2,Y2
440 NEXT N
450 REM DIBUJAMOS SEGMENTOS
460 P=-1
470 FOR N=1 TO NUM
480 P=P+1:IF P=4 THEN P=0
490 MOVE XR,YR
500 X1=XR+RA*COS(C(N))
510 Y1=YR+RA*SIN(C(N))
520 DRAW X1,Y1
530 X2=XR+(RA/2)*COS(PUN(N))
540 Y2=YR+(RA/2)*SIN(PUN(N))
550 SUT=LEN(G$(N))*3
560 MOVE X2,Y2
570 FILL P
580 IF X2<XR THEN SUT=SUT*4
590 MOVE X2-SUT,Y2
600 NEXT N
610 GOSUB 680
620 END
630 DOS=(((C(EX)-C(EX-1))/2)+C(EX-1)
640 EQ1=XR:EQ2=YR
650 XR=EQ1+20*COS(DOS)
660 YR=EQ2+20*SIN(DOS)
670 RETURN
680 P=-1
690 FOR N=1 TO NUM
```

```
700 P=P+1:IF P=4 THEN P=0
710 MOVE XR,YR
720 X1=XR+RA*COS(C(N))
730 Y1=YR+RA*SIN(C(N))
740 DRAW X1,Y1
750 X2=XR+(RA/2)*COS(PUN(N))
760 Y2=YR+(RA/2)*SIN(PUN(N))
770 SUT=LEN(G$(N))*3
780 MOVE X2,Y2
790 IF X2<XR THEN SUT=SUT*4
800 MOVE X2-SUT,Y2
810 PRINT G$(N);
820 NEXT N
```



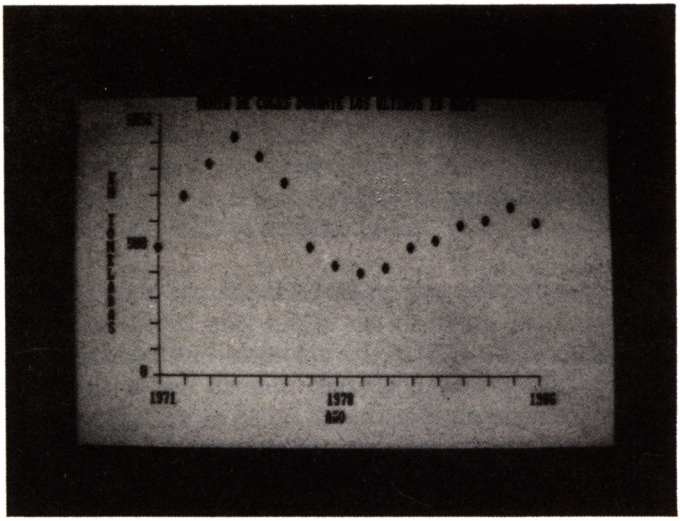
El Gráfico Comercial Más Antiguo Del Mundo

Es sabido por todos que la técnica más antigua para dibujar en un ordenador fue la línea. Los gráficos comerciales de línea convertían una gran cantidad de abstractos números en una información comprensible hasta para un niño. Las gráficas más complicadas poseen más de tres o cuatro variables, que trazadas en conjunto, empiezan a ser menos comprensibles para el niño anterior.

Tras estudiar los dos primeros capítulos, sabremos perfectamente cómo trazar una línea en nuestra pantalla. Empezaremos, de todas formas, con el programa más simple de este tipo que vamos a presentar.

CUADRO

```
10 INK 0,13
20 INK 1,0
30 READ NOMBRE$
40 OP=1:
50 MODE 2
60 READ PUN
70 DIM X(PUN),Y(PUN)
80 FOR N=1 TO PUN
90 READ X(N)
100 READ Y(N)
110 NEXT N
120 READ XMIN,XMAX,YMIN,YMAX
130 READ A$
140 READ B$
150 CLS
160 MOVE 100,380
170 DRAW 100,80
180 DRAW 550,80
190 FOR N=1 TO 11
200 MOVE 90,(N*30)+50
210 DRAW 100,(N*30)+50
220 NEXT N
230 FOR N=1 TO 16
240 MOVE (N*30)+70,70
250 DRAW(N*30)+70,80
260 NEXT N
270 L1=LEN(NOMBRE$):L1=40-(L1/2)
280 LOCATE L1,1:PRINT NOMBRE$;
290 DX=(320-((LEN(A$)*16)/2))
300 DY=(220+((LEN(B$)*16)/2))
310 TAG
320 MOVE DX,40
330 PRINT A$;
340 FOR N=1 TO LEN (B$):DA$=MID$(B$,N,1)
350 MOVE 40,DY-((N-1)*16)
360 PRINT DA$;
370 NEXT N
380 MOVE 530,60:PRINT XMAX;
390 MOVE 50,382:PRINT YMAX;
400 MOVE 70,90:PRINT YMIN;
410 MOVE 80,60:PRINT XMIN;
420 MOVE 55,240:PRINT INT((YMAX+YMIN)/2);
430 MOVE 290,60:PRINT INT((XMAX+XMIN)/2);
440 IF OP=2 THEN GOTO 520
450 FOR N=1 TO PUN
460 XUP=XMAX-XMIN:YUP=YMAX-YMIN
470 XVER=XUP-(XMAX-X(N)):YVER=YUP-(YMAX-Y(N))
480 MOVE 96+(450*(XVER/XUP)),86+(300*(YVER/YUP))
490 PRINT CHR$(231);
500 NEXT N
510 END
520 REM
530 FOR N=1 TO POINTS
540 XUP=XMAX-XMIN:YR=YMAX-YMIN
550 XVER=XUP-(XMAX-X(N)):YVER=YUP-(YMAX-Y(N))
560 IF N=1 THEN MOVE 96+(450*(XVER/XUP)),86+(300*(YVER/YUP))
570 DRAW 96+(450*(XVER/XUP)),86+300*(YVER/YUP)
580 NEXT N
590 END
600 REM DATA
610 DATA"VENTA DE COLES DURANTE LOS ULTIMOS 16 A!OS"
620 DATA 16,1971,500,1972,700,1973,820,1974,920,1975,850,1976,750,1977
,500,1978
630 DATA 430,1979,400,1980,420,1981,500,1982,530,1983,590,1984,610,198
5,660,1986,600
640 DATA 1971,1986,0,1000
650 DATA "A\O","EN TONELADAS"
```



El programa CUADRO está diseñado en el modo 2 para darle una apariencia más profesional. Los gráficos son igualmente buenos, tanto en monocromo como en color, ya que la falta de color no supone ningún hándicap en este tipo de gráficos, salvo excepciones, como veremos.

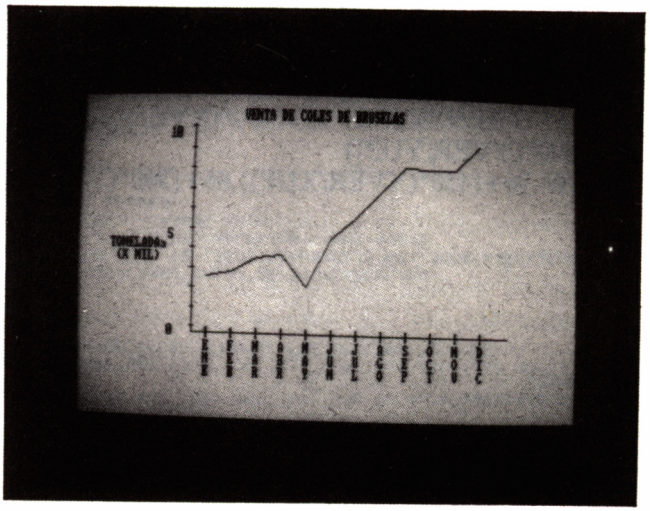
El programa toma los valores máximos y mínimos de X e Y, y los distribuye de manera escalonada dentro de las coordenadas. CUADRO coloca en cada coordenada calculada un punto, pero si desease que apareciera una línea fluctuante no tendría más que introducir las siguientes instrucciones:

```
600  IF N=1 THEN MOVE 96+(450*(XVER/XUP)),
     86+(300*(YVER/YUP))
610  DRAW 96+(450*(XVER/XUP)),86+(300*(YVER/YUP))
```

CUADRO2

```
10 REM
20 REM
30 REM
40 REM
50 DIM XR(12),YR(12)
60 CLS:MODE 2:INK 0,13:INK 1,0
70 REM
80 INPUT"TITULO PRINCIPAL (MAX 80 CARAC.)";N$
90 INPUT"TITULO SECUNDARIO (MAX 20 CARAC.)";S$
95 INPUT"TITULO SECUNDARIO (SUB)";S1$
100 REM
110 R1=LEN (N$)
```

```
120 R2=LEN (S$)
125 R3=LEN (S1$)
130 XC=40-(R1/2)
140 AS=10-(R2/2)
145 XR1=10-(R3/2)
150 CLS
160 LOCATE XC,2:PRINT N$
170 LOCATE AS,12:PRINT S$
175 LOCATE XR1,13:PRINT S1$
180 GOSUB 450
190 REM
200 MOVE 145,365:DRAW 145,105
210 DRAW 550,105
220 REM
230 FOR N=362 TO 112 STEP -25
240 MOVE 142,N:DRAW 147,N
250 NEXT N
260 REM
270 READ M
280 F=131
290 TAG:MOVE 105,360:PRINT M;
300 MOVE 105,235:PRINT M/2;
310 MOVE 105,112:PRINT 0;
320 FOR N=1 TO 12
330 READ VALOR
340 VALOR=((VALOR/M)*250)+112
350 F=F+32
360 IF N=1 THEN X2=F:Y2=VALOR
370 X1=X2:Y1=Y2
380 X2=F:Y2=VALOR
390 MOVE X1,Y1
400 DRAW X2,Y2
420 NEXT N
440 GOTO 440
450 REM
460 LOCATE 1,19
470 PRINT TAB (21);"¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦"
480 PRINT TAB (21);"E   F   M   A   M   J   J   A   S   O   N   D"
490 PRINT TAB (21);"N   E   A   B   A   U   U   G   E   C   O   I"
500 PRINT TAB (21);"E   B   R   R   Y   N   L   O   P   T   V   C"
510 RETURN
520 DATA 10
530 DATA 2.6,2.8,3.5,3.7,2.1,4.6,5.6,6.9,8.2,8.1,8.1,9.3
```



Las fluctuaciones de los datos pueden también estar referidas a un corto período de tiempo. La técnica es en esta ocasión más o menos parecida a la de CUADRO. El siguiente programa, llamado CUADRO 2 (tendremos hasta cuatro), se utiliza con este propósito.

En las líneas finales se encuentran los datos, los cuales son recogidos y dibujados por las líneas 320-420. El programa ilustra las variaciones durante un período de doce meses. Estos se hallan situados en el eje X.

A veces puede resultar interesante una comparación entre dos líneas de datos. En el más simple de los casos, las líneas pueden ser trazadas alterando sensiblemente CUADRO o CUADRO 2 con diferentes colores o con distinto tipo de puntos.

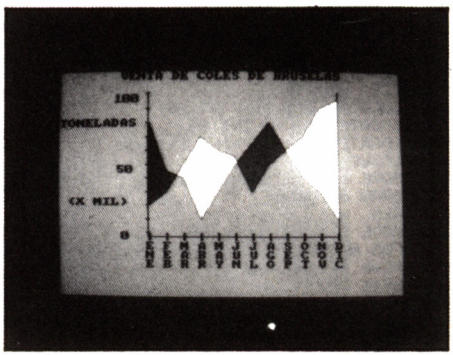
Si la comparación se desea hacer desde un punto de vista competitivo, puede ser útil enfatizar los períodos en los que uno de los dos productos, por ejemplo, fue dominante sobre el otro. Esto se consigue, naturalmente, utilizando el comando FILL.

CUADRO 3, además de emplear el comando FILL, inconveniente para los poseedores de un 464, está programado en modo 1, por lo cual es algo más basto, gráficamente hablando, que los programas anteriores. CUADRO 3 ilumina con claridad la diferencia entre las dos filas de datos que deseamos comparar.

CUADRO3

```
10 DIM XR(12),YR(12),COM (2,12)
20 CLS:MODE 2:INK 0,13:INK 1,0:INK 2,24:INK 3,6
30 INPUT"TITULO PRINCIPAL (MAX 40 CARAC.)";N$
40 INPUT"TITULO SECUNDARIO SUPERIOR (MAX.10 CARAC.)";S$
50 INPUT"TITULO SECUNDARIO INFERIOR (MAX.10 CARAC.)";S1$
55 MODE 1
60 R1=LEN (N$)
70 R2=LEN (S$)
80 R3=LEN (S1$)
90 XC=20-(R1/2)
100 AS=5-(R2/2)
110 XR1=5-(R3/2)
120 CLS
130 LOCATE XC+1,1:PRINT N$
140 LOCATE AS,6:PRINT S$
150 LOCATE XR1,15:PRINT S1$
160 GOSUB 570
170 MOVE 164,365:DRAW 164,105
180 DRAW 520,105
190 REM
200 MOVE 520,105:DRAW 520,365
210 FOR N=362 TO 112 STEP -25
220 MOVE 161,N:DRAW 167,N
230 NEXT N
240 REM
250 READ M
260 F=135
270 TAG:MOVE 85,360:PRINT M;
```

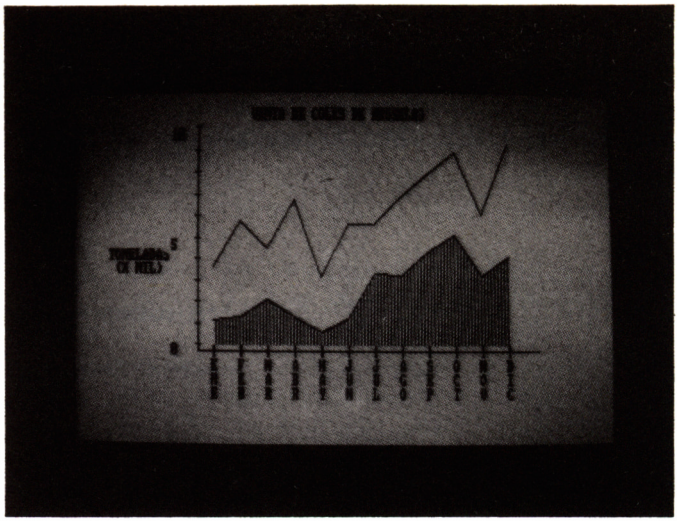
```
280 MOVE 90,235:PRINT M/2;
290 MOVE 98,112:PRINT 0;
300 FOR N=1 TO 12
310 READ VALOR
320 VALOR=((VALOR/M)*250)+112
330 COM(1,N)=VALOR
340 YMIN=115
350 F=F+32
360 IF N=1 THEN X2=F:Y2=VALOR
370 X1=X2:Y1=Y2
380 X2=F:Y2=VALOR
390 MOVE X1,Y1
400 DRAW X2,Y2
410 NEXT N
420 F=135
430 FOR N=1 TO 12
440 READ VALOR
450 VALOR=((VALOR/M)*250)+112
460 COM(2,N)=VALOR
470 F=F+32
480 IF N=1 THEN X2=F:Y2=VALOR
490 X1=X2:Y1=Y2
500 X2=F:Y2=VALOR
510 MOVE X1,Y1
520 DRAW X2,Y2
530 NEXT N
540 GOSUB 660
550 END
560 REM
570 LOCATE 1,19
580 PRINT TAB (11);"¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
590 PRINT TAB (11);"E F M A M J J A S O N D"
600 PRINT TAB (11);"N E A B A U U G E C O I"
610 PRINT TAB (11);"E B R R Y N L O P T V C"
620 RETURN
630 DATA 100
640 DATA 80,45,40,10,30,50,65,80,60,40,30,10
650 DATA 20,30,50,70,60,55,30,50,60,70,90,95
660 ALV=135:FOR N=1 TO 12
670 IF COM(1,N) < COM(2,N) THEN LOC=2
680 IF COM(1,N) > COM(2,N) THEN LOC=3
690 IF COM(1,N) = COM(2,N) THEN ALV=ALV+32:GOTO 740
700 YY=(COM(1,N)+COM(2,N))/2
710 ALV=ALV+32
720 MOVE ALV,YY
730 FILL LOC
740 NEXT N
750 RETURN
```



Ahora imaginemos que una de las líneas de datos es notablemente superior a la otra. Podemos emplear otro método para realzar la diferencia existente entre la venta de dos productos, la producción de dos empresas, los gastos caseros en dos alimentos, etc., utilizando el programa CUADRO 4. Este ofrece el resultado que podemos ver en las figuras.

```
CUADRO4

10 REM
12 REM
15 YMIN=400
45 REM
46 REM
260 REM
270 RFAD M
280 F=131
290 TAG:MOVE 105,360:PRINT M;
300 MOVE 105,235:PRINT M/2;
310 MOVE 105,112:PRINT 0;
320 FOR N=1 TO 12
330 READ VALOR
340 VALOR=((VALOR/M)*250)+112
345 YMIN=115
350 F=F+32
360 IF N=1 THEN X2=F:Y2=VALOR
370 X1=X2:Y1=Y2
380 X2=F:Y2=VALOR
390 MOVE X1,Y1
400 DRAW X2,Y2
420 NEXT N
422 REM
424 F=131
426 FOR N=1 TO 12
428 READ VALOR
430 VALOR=((VALOR/M)*250)+112
432 F=F+32
434 IF N=1 THEN X2=F:Y2=VALOR
436 X1=X2:Y1=Y2
438 X2=F:Y2=VALOR
440 MOVE X1,Y1
442 DRAW X2,Y2
444 NEXT N
446 GOSUB 1000
448 GOTO 448
520 DATA 10
530 DATA 3.6,5.8,4.5,6.7,3.1,5.6,5.6,6.9,7.9,8.9,6.1,9.3
540 DATA 1.1,1.3,2.1,1.3,0.6,1.2,3.3,3.2,4.3,5.1,3.2,4.1
1000 REM
1005 AU=4
1010 YV=YMIN-2:REM
1015 ALV=250+AU
1020 FOR YY=YV TO 400 STEP 2
1030 IF TEST (ALV,YY)<>0 THEN 1100
1040 PLOT ALV,YY
1065 NEXT YY
1100 FOR YY=YV TO 0 STEP -2
1110 IF TEST (ALV,YY)<>0 THEN 1200
1120 PLOT ALV,YY
1130 NEXT YY
1200 ALV=ALV+AU
1210 IF ALV>515 THEN AU=-AU:ALV=250
1220 IF ALV<165 THEN RETURN
1230 GOTO 1040
```



Esta vez el 464 no se verá afectado por la influencia de algún comando FILL. La parte sombreada se lleva a cabo gracias a una rutina que se mueve a lo largo del eje X, de INC en INC puntos (línea 1005), y dibuja una línea vertical desde Y=0 hasta que se encuentra con la línea de datos.

CUADRO 4 es una mejora del programa CUADRO 2. Ejecutando el comando MERGE de estas líneas encima de CUADRO 2 ahorraremos gran cantidad de trabajo.

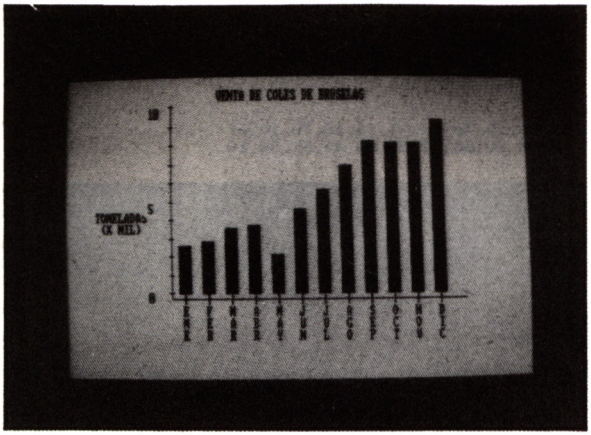
GRÁFICOS DE BARRAS

Una técnica muy útil para la interpretación de datos en forma de gráficos son los cuadros o tablas de barras. La manera más simple de crear este tipo de gráficos la encontraremos en el programa BARRAS, el cual deberá ser «MERGEado» con CUADRO 2 para su buen funcionamiento.

BARRAS

```
10 REM
20 REM
260 REM
270 READ m
280 F=131
290 TAG:MOVE 105,360:PRINT M;
300 MOVE 105,235:PRINT M/2;
310 MOVE 105,112:PRINT 0;
```

```
320 FOR N=1 TO 12
330 READ VALOR
340 VALOR=((VALOR/M)*250)+112
350 F=F+32
360 REM
370 MOVE F-8,VALOR:DRAW F+8,VALOR
380 DRAW F+8,112
390 DRAW F-8,112
400 DRAW F-8,VALOR
410 MOVE F,115:FILL 1
420 NEXT N
```



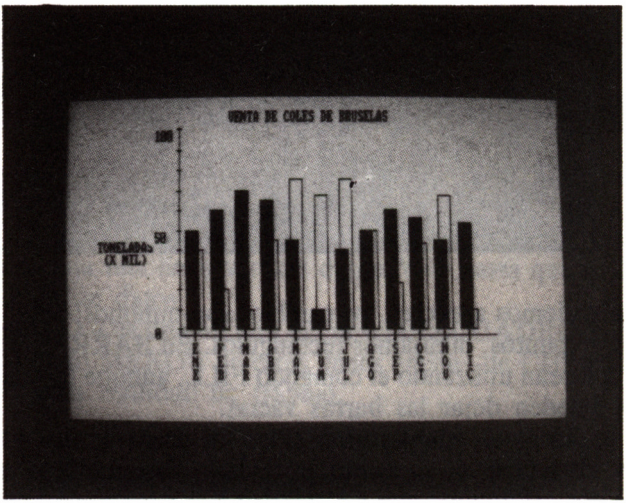
Como podemos apreciar, los cambios son mínimos: en vez de trazar series de puntos, dibujamos rectángulos. En BARRAS los rectángulos se colorean utilizando el comando FILL, aunque los usuarios de CPC 464 pueden dejar las barras vacías.

Para realizar una comparación entre los datos de dos productos, podemos introducir el siguiente programa, fusionándolo mediante MERGE posteriormente con CUADRO 2. El resultado será francamente sorprendente.

BARRAS2

```
10 REM
20 REM
25 REM
260 REM
265 READ M
270 F=129
275 TAG:MOVE 105,360:PRINT M;
280 MOVE 105,235:PRINT M/2;
285 MOVE 105,112:PRINT 0;
290 FOR N=1 TO 12
295 READ VALOR
300 VALOR=((VALOR/M)*250)+112
305 F=F+32
310 REM
```

```
315 MOVE F-8,VALOR:DRAW F+8,VALOR
320 DRAW F+8,112
325 DRAW F-8,112
330 DRAW F-8,VALOR
335 MOVE F,115:FILL 1
340 REM
345 READ VALOR
350 VALOR=((VALOR/M)*250)+112
355 F=F+6
360 REM
365 MOVE F-8,VALOR:DRAW F+8,VALOR
370 DRAW F+8,112
375 DRAW F-8,112
380 DRAW F-8,VALOR
385 F=F-6
390 NEXT N
400 REM
420 REM
520 DATA 100
530 DATA 50,40,60,20,70,10,65,45,45,75,10,67
540 DATA 40,75,50,50,60,23,56,43,45,67,54,10
```



GRÁFICOS DE BARRA EN TRES DIMENSIONES

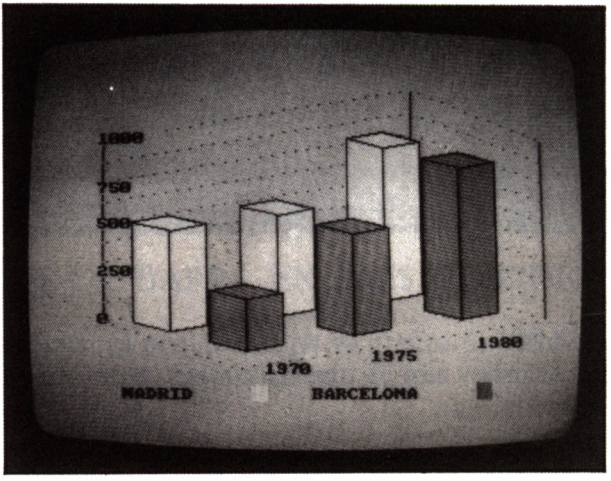
Naturalmente, es posible dibujar un gráfico de barras en tres dimensiones para ver el comportamiento de tres variables simultáneamente.

H3D

```
10 REM
20 CLS
30 MODE 1:INK 0,13:INK 1,0:INK 2,9:INK 3,15:BORDER 13
40 REM
50 GRAPHICS PEN 1
60 X=460:XL=40:XR=639
70 MOVE X,400
80 DRAW X,160
```

```
90 MOVE XL,340
100 DRAW XL,100
110 MOVE XR,340
120 DRAW XR,100
130 A=1000
140 RW=4
150 C$="E"
160 FOR Y=400 TO 160 STEP -(160/6)
170 MOVE X,Y
180 MASK 16
190 DRAW XL,Y-60
200 IF C$="O" THEN 270
210 LOCATE 2,RW
220 PRINT A;
230 C$="O"
240 A=A-250
250 RW=RW+3.7
260 GOTO 280
270 C$="E"
280 MOVE X,Y
290 DRAW XR,Y-60
300 NEXT Y
310 MOVE 40,100
320 DRAW 220,40
330 DRAW 639,100
340 MASK 255
350 LOCATE 17,23:PRINT "1970";
360 LOCATE 26,22:PRINT "1975";
370 LOCATE 35,21:PRINT "1980";
380 H=40:C=0
390 XL=80
400 YL=100
410 XS=240
420 MR=-50/210
430 ML=15/89
440 F=3:O=2
450 H=100
460 REM
470 GRAPHICS PEN F
480 FOR X=XS TO XS+20
490 MOVE X,20
500 DRAW X,1
510 NEXT X
520 LOCATE 5,25:PRINT "FRANCIA"
530 LOCATE 21,25:PRINT "ESPA\A"
540 FOR J=1 TO 3
550 READ DA
560 TP=(240*DA)/900
570 GOSUB 710
580 GOSUB 1010
590 XL=XL+146
600 YL=YL+20
610 NEXT J
620 C=C+1
630 LOCATE 1,1
640 IF C=2 THEN END
650 XL=182
660 YL=75
670 F=2
680 O=3
690 XS=544
700 GOTO 460
710 REM
720 BL=YL+TP-ML*XL
730 BR=YL-MR*XL
740 FOR X=XL TO XL+H/2
750 Y1=ML*X+BL
760 Y2=MR*X+BR
770 GRAPHICS PEN F
780 MOVE X,Y1
790 DRAW X,Y2
800 GRAPHICS PEN 1
810 PLOT X,Y1
```

```
820 PLOT X,Y2
830 NEXT X
840 YT=Y1
850 YB=Y2
860 BL=YB-ML*(XL+H/2)
870 BR=YT-MR*(XL+H/2)
880 FOR X=XL+H/2 TO XL+H
890 Y1=MR*X+BR
900 Y2=ML*X+BL
910 GRAPHICS PEN F
920 MOVE X,Y1
930 DRAW X,Y2
940 GRAPHICS PEN 1
950 PLOT X,Y1
960 PLOT X,Y2
970 NEXT X
980 YR=Y2
990 RETURN
1000 REM
1010 MOVE XL,YL
1020 GRAPHICS PEN 1
1030 DRAW XL,YL+TP
1040 MOVE XL+H/2,YB
1050 DRAW XL+H/2,YB+TP
1060 MOVE XL+H,YR
1070 DRAW XL+H,YR+TP
1080 MOVE XL+H/2,YB+TP
1090 DRAW XL+H,YR+TP
1100 MOVE XL,YL+TP
1110 DRAW XL+H/2,YR+TP-(6*(H/80))
1120 RETURN
1130 DATA 500,500,750,250,500,750
```



Existen otro tipo de gráficas de negocios que no hemos nombrado, por ejemplo, exhibir un gráfico de pastel y de barras simultáneo. Estos programas, no son más que la base de todos ellos.

GRÁFICOS EN TRES DIMENSIONES

La vista humana capta los objetos en tres dimensiones, por lo cual es lógico pensar que la representación más real de una figura en nuestro ordenador, es aquella que visualiza su altura, su anchura y la tercera dimensión: su profundidad.

Un sistema de coordenadas bidimensional está formado por dos ejes perpendiculares, X e Y. En este sistema podríamos definir una superficie, conociendo las coordenadas de todos sus puntos. Pero a la hora de dar volumen, «consistencia», a esa figura habríamos de añadir a ese sistema un eje más, igualmente perpendicular a cada uno de los otros dos. Se trata del eje Z.

Si dibujamos en un papel los ejes X e Y, formando noventa grados el uno respecto al otro (perpendiculares), el tercer eje (Z) se encontrará apuntando hacia nosotros y, por tanto, como máximo sólo podremos apreciar un punto de él.

Un objeto en tres dimensiones colocado en un sistema tridimensional de este tipo sólo nos permitiría observar la parte frontal de él. Para apreciar la triple dimensión de la figura es necesario cambiar

nuestro punto de vista con respecto al objeto. Gracias a ello, lo podremos observar en perspectiva.

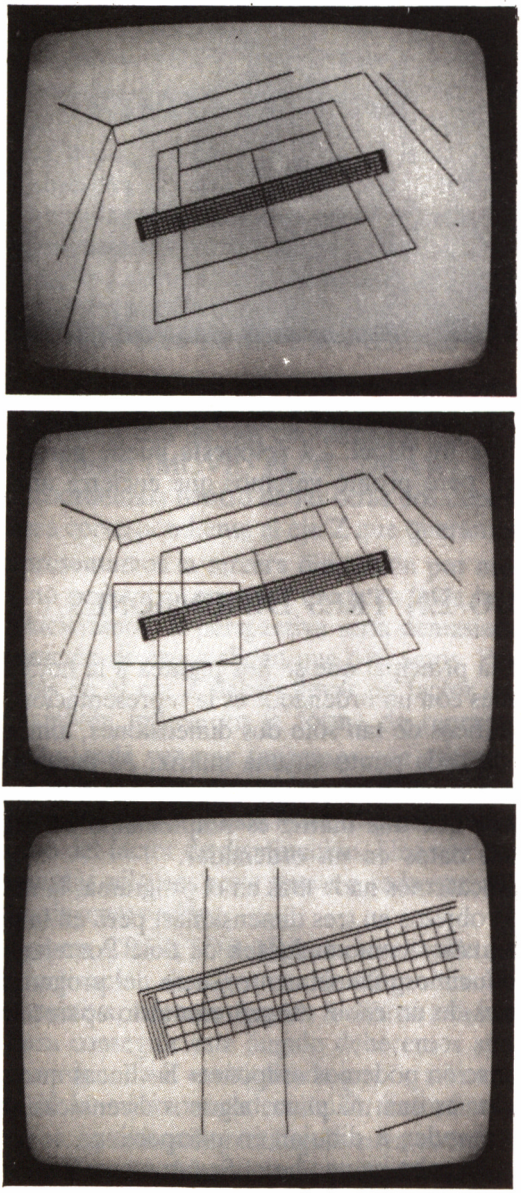
No es necesario que aprendamos enormes y complicadas fórmulas matemáticas y trigonométricas que manejen y transformen, en una palabra, manipulen los datos en tres dimensiones. Para utilizar el programa que le vamos a presentar a continuación, nada de esto es necesario, pero recordemos que al principio todo nos puede parecer más complicado de lo que en realidad es.

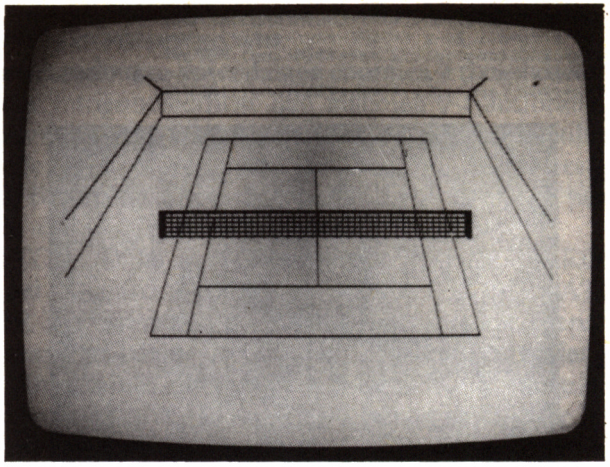
CREATOR 3D ha de ser fusionado mediante MERGE con el conjunto del programa formado por CREATOR, CREATOR 2D y CREATOR 3. Finalmente, poseeremos un programa preparado para dibujar, rotar, trasladar y transformar de tamaño, cualquier gráfico en dos y tres dimensiones, además de la posibilidad de grabarlo y cargarlo del disco o casete. La guinda de este merengue, la pone la compatibilidad para todos los ordenadores de la serie CPC de Amstrad.

CREATOR4

```
5 'EXTENSION DEL PROGRAMA CREATOR 2D
6 'POR DANIEL LOZANO VALVERDE
7 '              &
8 'RAFAEL DE LA OSSA VILLACAÑAS
30 MODE 1:PAPER 0:PEN 1:BORDER 5:LOCATE 15,3:PRINT "CREATOR 3D"
40 RESTORE 50:FOR i=7 TO 21 STEP 2:READ a$,b$:LOCATE 2,i:PRINT a$,b$:N
EXT i
60 DATA "B         - Cls","O         - Origen","C         - Cargar","G
     - Grabar","]         - 3D(+)","\         - 3D(-)","","","","Return
- Menu/Dibujo"
115 IF a=93 THEN GOSUB 290:Z=Z+2:GOSUB 290
135 IF a=92 THEN GOSUB 290:Z=Z-2:GOSUB 290
270 DIM d(1000,2):R=0:TR=0:O=0:t=1:x=320:y=200:Z=0:d(0,0)=1000000:d(1,
0)=x:d(1,1)=y:D(1,2)=Z:RETURN
290 IF O=1 THEN PLOT x+Z/2,y+Z/3,1,1:RETURN ELSE MOVE d(t,0)+D(T,2)/2,
d(t,1)+D(T,2)/3:DRAW x+Z/2,y+Z/3,1,1:RETURN
360 t=t+1:d(t,0)=x:d(t,1)=y:D(T,2)=Z:GOTO 100
390 GOSUB 290::x=d(t,0):y=d(t,1):Z=d(t,2):t=t-1:PLOT x+z/2,y+z/3:GOTO
100
410 CLS:FOR i=0 TO t:IF d(i,0)=1000000 THEN i=i+1:PLOT d(i,0)+d(i,2)/2
,d(i,1)+d(i,2)/3:GOTO 430
420 PLOT D(I-1,0)+D(I-1,2)/2,D(I-1,1)+D(I-1,2)/3:DRAW d(i,0)+D(I,2)/2,
d(i,1)+d(i,2)/3,1,1
430 NEXT i:x=d(t,0):y=d(t,1):z=d(t,2):GOTO 100
515 D(T,0)=1000000:T=T+1:D(T,0)=X:D(T,1)=Y:d(t,2)=z:O=0:PLOT X+z/2,Y+z
/3:GOTO 100
550 WRITE #9,d(n,0):WRITE #9,d(n,1):WRITE #9,d(n,2)
630 INPUT #9,D(n,0):INPUT #9,D(n,1):INPUT #9,D(n,2)
670 r=0:o=0:LOCATE 1,1:INPUT "Eje de giro (X=1 Y=2 Z=3)";e:INPUT "Angu
lo";a
690 ON e GOSUB 970,980,990
720 tr=0:o=0:xr=x-d(t,0):yr=y-d(t,1):zr=z-d(t,2)
740 d(i,0)=d(i,0)+xr:d(i,1)=d(i,1)+yr:d(i,2)=d(i,2)+zr
770 x=208:y=130:z=0:xr=223:yr=138:GOSUB 880
795 IF a=93 THEN GOSUB 880:z=z+2:GOSUB 880
815 IF a=92 THEN GOSUB 880:z=z-2:GOSUB 880
920 d(i,0)=(d(i,0)-x)*au:d(i,1)=(d(i,1)-y)*au:d(i,2)=(d(i,2)-z)*au
950 d(i,0)=d(i,0)/au+x:d(i,1)=d(i,1)/au+y:d(i,2)=d(i,2)/au+z
```

```
970 zr=d(i,2)-z:yr=d(i,1)-y:c=SQR(zr*zr+yr*yr):ang=a+ATN(yr/zr)+180*(z
r<0):d(i,2)=z+c*COS(ang):d(i,1)=y+c*SIN(ang):RETURN
980 xr=d(i,0)-x:zr=d(i,2)-z:c=SQR(xr*xr+zr*zr):ang=a+ATN(zr/xr)+180*(x
r<0):d(i,0)=x+c*COS(ang):d(i,2)=z+c*SIN(ang):RETURN
990 xr=d(i,0)-x:yr=d(i,1)-y:c=SQR(xr*xr+yr*yr):ang=a+ATN(yr/xr)+180*(x
r<0):d(i,0)=x+c*COS(ang):d(i,1)=y+c*SIN(ang)::RETURN
```





¿Cómo representar una figura en tres dimensiones en una pantalla de tan sólo dos? De la misma manera que dibujamos un objeto tridimensional sobre un papel. La respuesta puede parecer obvia o evidente a priori, pero pronto veremos que encierra una cierta dificultad.

Dibujando En Tres Dimensiones

El problema principal que se nos plantea a la hora de trabajar en tres dimensiones con un ordenador es la representación. Como hemos visto en los gráficos de tan sólo dos dimensiones, almacenábamos las coordenadas de cada punto en una matriz. Al pasar a la tercera dimensión, no necesitamos más que ampliar un poco la misma matriz anterior, puesto que una matriz es simplemente un lugar preparado para almacenar datos en un ordenador.

Si no modificásemos nada más en el programa, la máquina tendría en su memoria objetos en tres dimensiones, pero en la pantalla aparecerían representados como si fuesen de dos. Por ejemplo, supongamos que introducimos manualmente (fuera del programa) las coordenadas espaciales de un cubo. Al representarlo aparecería en pantalla un cuadrado.

Con imaginación podemos «suponer» las líneas que no vemos, pero un ordenador es una máquina, algunos dicen «casi» inteligente, y le será fácil aprender a dibujar en perspectiva.

Dibujemos un cubo en una hoja. Seguramente para hacerlo habre-

mos trazado dos cuadrados y unido sus vértices con líneas. La tercera dimensión, que es perpendicular al papel, no debería aparecer y, sin embargo, la hemos dibujado. Para ello, hemos «girado» el eje Z de forma que aparezca como una línea ligeramente inclinada. A esto se le llama dibujar en perspectiva.

No es esta la única forma de perspectiva que existe, pero sí es la más fácil para nosotros, y como es lógico, la más sencilla para el ordenador. Si enseñamos al ordenador a dibujar la tercera dimensión como una línea inclinada, los gráficos seguirán teniendo dos dimensiones, pero podremos apreciar la perspectiva, la cual nos dará la impresión de estar observando una figura en el espacio.

Los cambios necesarios para conseguir construir los dibujos en la pantalla, trasladarlos, aumentarlos, disminuirlos y rotarlos, se logran manipulando los datos almacenados en la matriz, en la cual poseemos tres coordenadas para cada punto. Estos cambios los veremos algo más adelante.

Pero cuando se trata de dibujar cualquier punto o línea en perspectiva, sólo contamos con las dos coordenadas que nos permiten las instrucciones de dibujo (MOVE, PLOT, DRAW). La forma de incluir la tercera coordenada dentro de las otras dos depende de la inclinación de la línea correspondiente al eje Z que queramos representar.

Cuando representamos la tercera dimensión por medio de una línea inclinada, no podemos conservar en dicha línea las distancias habituales. Si lo hiciéramos, la perspectiva sería demasiado exagerada y no daría la sensación de espacialidad que deseamos. Por ello, las unidades expresadas en el eje Z deben llevar un factor de disminución. Dicho factor también influye en la forma de representar este eje.

Para todo ello, cada vez que tengamos que utilizar alguna de las instrucciones de dibujo habrá que incluir en ellas un factor de conversión. Como acabamos de decir, este factor depende tanto de la inclinación del eje como del factor de disminución. Una vez decidida la perspectiva a representar sólo nos resta sumar a cada una de las coordenadas este factor de conversión.

Para el programa CREATOR 3D hemos utilizado como eje Z una línea con una inclinación de algo más de 30 grados. Ello nos ofrece una buena perspectiva, además de gran facilidad a la hora de efectuar los cálculos. Para conseguir este ángulo deberemos añadir a la coordenada X, la coordenada Z multiplicada por 3, a la vez que sumar a la coordenada Y la misma coordenada Z multiplicada por 2.

Si construimos un triángulo rectángulo cuya coordenada X sea 3 y cuya coordenada Y sea 2, el ángulo más pequeño tendrá 33 grados, 41

minutos y 24.24 segundos, que es precisamente el ángulo de la perspectiva conseguida.

Pero como ya hemos dicho, esta perspectiva es bastante falsa, ya que las coordenadas del eje Z serán muy exageradas. Si nosotros dibujáramos un cubo con este sistema, en la matriz tendríamos un cubo, pero en la pantalla veríamos un paralelepípedo un tanto alargado. Esto se debe a que no hemos incluido aún el factor de disminución.

Para hacerlo sin necesidad de cambiar el ángulo del eje Z bastará, en vez de multiplicar, dividir la coordenada Z por los números 2 y 3, esta vez intercambiados. Es decir, sumar a la coordenada X la coordenada Z dividida por 2 y añadir a la coordenada Y dicha coordenada Z dividida por 3. Con esto, la perspectiva obtenida es bastante real y nuestro cubo alargado volverá a ser normal.

CREAT3D

```
5 '*** PROGRAMA CREATOR ***
6 '*** GRAN BIBLIOTECA AMSTRAD ***
7 '*** PROGRAMADO POR DANIEL LOZANO VALVERDE ***
10 ZONE 10:DEG:GOSUB 270
20 REM RUTINA DE PRESENTACION Y MENU
30 MODE 1:PAPER 0:PEN 1:BORDER 5:LOCATE 15,3:PRINT "CREATOR 3D"
40 RESTORE 50:FOR i=7 TO 21 STEP 2:READ a$,b$:LOCATE 2,i:PRINT a$,b$:N
EXT i
50 DATA "Cursores - Mover","Espacio  - Fijar","Delete   - Borrar","R
     - Rotar","T        - Trasl.","A        - Aumentar"
60 DATA "B        - Cls","O        - Origen","C        - Cargar","G
     - Grabar","]        - 3D(+)","\        - 3D(-)","","","","Return
- Menu/Dibujo"
70 a$=INKEY$:IF a$=CHR$(13) THEN 90 ELSE 70
80 REM PROGRAMA PRINCIPAL
90 MODE 2:PLOT x,y:IF t>1 THEN 410
100 a$=INKEY$:IF a$="" THEN 100 ELSE a=ASC(a$)
110 IF a=240 THEN GOSUB 290:y=y+2:GOSUB 290
115 IF a=93 THEN GOSUB 290:Z=Z+2:GOSUB 290
120 IF a=241 THEN GOSUB 290:y=y-2:GOSUB 290
130 IF a=242  THEN GOSUB 290:x=x-2:GOSUB 290
135 IF a=92 THEN GOSUB 290:Z=Z-2:GOSUB 290
140 IF a=243 THEN GOSUB 290:x=x+2:GOSUB 290
150 IF a=32 THEN 310
160 IF a=127 THEN 380
170 IF a=82 OR a=114 THEN SOUND 1,200:o=1:r=1
180 IF a=66 OR a=98 THEN 480
190 IF a=84 OR a=116 THEN SOUND 1,200:o=1:tr=1
200 IF a=79 OR a=111 THEN SOUND 1,200:O=1
210 IF a=65 OR a=97 THEN 770
220 IF a=13 THEN 30
230 IF a=67 OR a=99 THEN 600
240 IF a=71 OR a=103 THEN GOSUB 520
250 GOTO 100
260 REM INICIALIZACION DE VARIABLES
270 DIM d(1000,2):R=0:TR=0:O=0:t=1:x=320:y=200:Z=0:d(0,0)=1000000:d(1,
0)=x:d(1,1)=y:D(1,2)=Z:RETURN
280 REM MOVIMIENTO DEL CURSOR
290 IF O=1 THEN PLOT x+Z/2,y+Z/3,1,1:RETURN ELSE MOVE d(t,0)+D(T,2)/2,
d(t,1)+D(T,2)/3:DRAW x+Z/2,y+Z/3,1,1:RETURN
300 REM RUTINA FIJAR LINEA U ORIGEN
310 IF t=1000 THEN 450
```

```
320 IF r=1 THEN 670
330 IF tr=1 THEN 720
340 IF O=1 THEN 510
350 REM FIJAR LINEA
360 t=t+1:d(t,0)=x:d(t,1)=y:D(T,2)=Z:GOTO 100
370 REM RUTINA DE BORRAR ULTIMA LINEA
380 IF d(t-1,0)=1000000 THEN 100
390 GOSUB 290::x=d(t,0):y=d(t,1):Z=d(t,2):t=t-1:PLOT x+z/2,y+z/3:GOTO
100
400 REM RUTINA DE DIBUJAR
410 CLS:FOR i=0 TO t:IF d(i,0)=1000000 THEN i=i+1:PLOT d(i,0)+d(i,2)/2
,d(i,1)+d(i,2)/3:GOTO 430
420 PLOT D(I-1,0)+D(I-1,2)/2,D(I-1,1)+D(I-1,2)/3:DRAW d(i,0)+D(I,2)/2,
d(i,1)+d(i,2)/3,1,1
430 NEXT i:x=d(t,0):y=d(t,1):z=d(t,2):GOTO 100
440 REM RUTINA DE MEMORIA AGOTADA
450 LOCATE 1,1:PRINT "MEMORIA  AGOTADA"
460 a$=INKEY$:IF a$="" THEN 460 ELSE 90
470 REM RUTINA DE BORRADO TOTAL
480 LOCATE 1,1:INPUT "Esta seguro de que quiere borrarlo todo";a$:IF a
$="s" OR A$="S" THEN ERASE d:GOSUB 270
490 GOTO 90
500 REM RUTINA DE FIJAR ORIGEN
510 GOSUB 290:IF T=1 THEN T=0 ELSE T=T+1
515 D(T,0)=1000000:T=T+1:D(T,0)=X:D(T,1)=Y:d(t,2)=z:O=0:PLOT X+z/2,Y+z
/3:GOTO 100
520 REM RUTINA DE GRABAR
530 LOCATE 1,1:INPUT "Nombre del fichero";n$
540 OPENOUT n$:WRITE #9,T: FOR n=0 TO t
550 WRITE #9,d(n,0):WRITE #9,d(n,1):WRITE #9,d(n,2)
560 NEXT n
570 CLOSEOUT
580 GOTO 90
590 REM RUTINA DE CARGA
600 LOCATE 1,1:INPUT"Nombre de fichero";N$
610 OPENIN N$:INPUT #9,T
620 FOR n=0 TO T
630 INPUT #9,D(n,0):INPUT #9,D(n,1):INPUT #9,D(n,2)
640 NEXT n:CLOSEIN
650 GOTO 90
660 REM RUTINA DE ROTACION
670 r=0:o=0:LOCATE 1,1:INPUT "Eje de giro (X=1 Y=2 Z=3)";e:INPUT "Angu
lo";a
680 FOR i=1 TO t:IF d(i,0)=1000000 THEN 700
690 ON e GOSUB 970,980,990
700 NEXT i:GOTO 90
710 REM RUTINA DE TRASLACION
720 tr=0:o=0:xr=x-d(t,0):yr=y-d(t,1):zr=z-d(t,2)
730 FOR i=1 TO t:IF d(i,0)=1000000 THEN 750
740 d(i,0)=d(i,0)+xr:d(i,1)=d(i,1)+yr:d(i,2)=d(i,2)+zr
750 NEXT i:GOTO 90
760 REM RUTINA DE AUMENTO
770 x=208:y=130:z=0:xr=223:yr=138:GOSUB 880
780 a$=INKEY$:IF a$="" THEN 780 ELSE a=ASC(a$)
790 IF a=240 THEN GOSUB 880:y=y+2:GOSUB 880
795 IF a=93 THEN GOSUB 880:z=z+2:GOSUB 880
800 IF a=241 THEN GOSUB 880:y=y-2:GOSUB 880
810 IF a=242 THEN GOSUB 880:x=x-2:GOSUB 880
815 IF a=92 THEN GOSUB 880:z=z-2:GOSUB 880
820 IF a=243 THEN GOSUB 880:x=x+2:GOSUB 880
830 IF a=48 THEN GOSUB 880:xr=xr*0.9:yr=yr*0.9:GOSUB 880
840 IF a=49 THEN GOSUB 880:xr=xr/0.9:yr=yr/0.9:GOSUB 880
850 IF a=13 THEN 90
860 IF a=32 THEN 890
870 GOTO 780
880 MOVE x,y:DRAWR xr,0,1,1:DRAWR 0,yr,1,1:DRAWR -xr,0,1,1:DRAWR 0,-yr
,1,1:RETURN
890 LOCATE 1,1:INPUT "Aumento o disminucion (a/d)";a$
900 au=640/xr:IF a$="d" OR a$="D" THEN 940
910 FOR i=1 TO t:IF d(i,0)=1000000 THEN 930
920 d(i,0)=(d(i,0)-x)*au:d(i,1)=(d(i,1)-y)*au:d(i,2)=(d(i,2)-z)*au
930 NEXT i:GOTO 90
940 FOR i=1 TO t:IF d(i,0)=1000000 THEN 960
```

```
950 d(i,0)=d(i,0)/au+x:d(i,1)=d(i,1)/au+y:d(i,2)=d(i,2)/au+z
960 NEXT i:GOTO 90
970 zr=d(i,2)-z:yr=d(i,1)-y:c=SQR(zr*zr+yr*yr):ang=a+ATN(yr/zr)+180*(z
r<0):d(i,2)=z+c*COS(ang):d(i,1)=y+c*SIN(ang):RETURN
980 xr=d(i,0)-x:zr=d(i,2)-z:c=SQR(xr*xr+zr*zr):ang=a+ATN(zr/xr)+180*(x
r<0):d(i,0)=x+c*COS(ang):d(i,2)=z+c*SIN(ang):RETURN
990 xr=d(i,0)-x:yr=d(i,1)-y:c=SQR(xr*xr+yr*yr):ang=a+ATN(yr/xr)+180*(x
r<0):d(i,0)=x+c*COS(ang):d(i,1)=y+c*SIN(ang):RETURN
```

MANIPULANDO GRÁFICOS EN TRES DIMENSIONES

Cuando se trata de transformar gráficos en tres dimensiones, cualquier operación que deseemos efectuar con ellos se limita a alterar la matriz. En esto no se diferencian en absoluto de los gráficos bidimensionales, con los que como vimos en el capítulo cuarto, añadiendo unas pequeñas subrutinas al programa principal, conseguíamos traslaciones, rotaciones y cambios de tamaño. Las tres dimensiones se comportan de la misma manera, sólo que esta vez las rutinas serán diferentes.

Por tanto, olvidémonos ahora de lo que hemos hablado anteriormente sobre la representación de tres dimensiones, sobre dos, de los factores de conversión, de la inclinación del eje Z, y de todo lo demás, y tengamos en cuenta que el ordenador trabajará internamente con tres dimensiones tan fácilmente como lo hizo con dos.

TRASLACIONES Y CAMBIOS DE TAMAÑO 3D

En lo que a traslaciones se refiere, la tercera dimensión no es en absoluto importante. Incluso podríamos eliminarla, ya que la tercera dimensión sólo se muestra en la pantalla como una perspectiva para dar sensación de espacialidad. Si utilizásemos la rutina de traslación del programa CREATOR 2, podríamos mover igualmente el gráfico por la pantalla. Esto se debe a que la coordenada Z de la matriz no se modifica y, por tanto, no influye en la representación del dibujo. Sin embargo, hemos incluido la traslación a través de la tercera dimensión para completar el programa y darle más vistosidad, aunque sea inútil.

A la hora de cambiar el tamaño del dibujo hemos conservado casi en su totalidad la rutina del programa CREATOR 2D, de manera que su manejo es el mismo. Unicamente hemos modificado, al igual

que en la traslación, el hecho de que podamos mover nuestra «ventana» por el eje Z. Aunque en este caso la coordenada de la tercera dimensión sí que es modificada y multiplicada por la fracción de aumento elegida por nosotros.

Por supuesto, hemos incluido también las teclas] y \, que producen el movimiento a través del eje Z.

Las complicaciones comienzan cuando empezamos a estudiar la rotación. Rotar un gráfico en tres dimensiones es algo más difícil que hacerlo en dos. Para empezar, en dos dimensiones giramos superficies con relación a un punto. En el espacio giramos volúmenes con relación a una recta. En el primer caso dicho punto se llamaba centro de rotación, ahora dicha recta se denomina eje de rotación.

Sin embargo, la realidad es menos complicada de lo que en principio podría parecer.

Para verlo de una manera más clara, imaginemos un objeto tridimensional como infinitas superficies una detrás de otra. Así, un cubo tendría el aspecto de millones de cuadrados colocados en fila. O dicho de otra forma, cojamos una butifarra y cortémosla en miles de finísimas lonchas.

Haciendo esto, teóricamente, habremos dividido el espacio tridimensional en infinitos planos de dos dimensiones. Como ya sabemos rotar objetos bidimensionales, girar objetos espaciales se reduce a girar infinitos gráficos de dos dimensiones. El eje de rotación se habrá convertido, por medio de estos cortes, en millones de puntos o centros de rotación.

Si observamos la rutina, ésta nos pregunta en primer lugar el eje de rotación. Sin embargo, lo que nos está preguntando, en realidad, es la dirección de dicho eje. Una vez hayamos desplazado el punto por la pantalla, deberemos indicarle a la máquina dicha dirección. El eje de rotación será aquel que pase por el punto elegido por nosotros y tenga la dirección indicada.

Aunque el ordenador lo ejecuta de otra manera, podemos imaginar que dividimos el espacio en infinitos planos perpendiculares a dicho eje, para rotar cada plano por separado. Sin embargo, no es necesario girar todos y cada uno de esos planos, sino sólo aquellos que contengan algún punto.

Al girar el gráfico en una dirección paralela al eje Y las coordenadas verticales no serán modificadas, es decir, no necesitamos hacer ningún cambio en ellas y podremos limitarnos a modificar únicamente las coordenadas X y Z.

Esto simplifica mucho los cálculos, pues la rotación de un punto

tridimensional con relación al eje Y es lo mismo que la rotación de las coordenadas X y Z de dicho punto, con relación al punto de corte del eje Y con el plano en el que se encuentre dicho punto. Este punto de corte actuará como centro de rotación y tendrá las coordenadas X y Z que tenga el punto elegido por nosotros con los cursores.

Si nos quedó clara la línea 690 del programa CREATOR 2, observaremos que lo que hace esta subrutina es canalizarse a una de otras tres subrutinas (líneas 970,980,990), según hayamos elegido la dirección del eje de rotación. Los cálculos y las fórmulas trigonométricas utilizadas en el programa CREATOR 3D son las mismas que en el anterior, intercambiando los nombres de las variables X, Y, Z, XR, YR y ZR, según estemos girando a través de un eje o de otro.

EL PROGRAMA

Acometamos, pues, sin más dilación el estudio del programa.

30-60: Pertenecen a la rutina de presentación y menú. En éste hemos introducido la posibilidad de poder mover el cursor a través de la tercera dimensión con las teclas «]» y «\».

115 y 135: Insertadas en el bucle principal para que las teclas «]» y «\» puedan ser utilizadas durante la ejecución del programa.

270: Esta es la línea a la cual se dirige el programa al comienzo, lugar de inicialización de variables y la propia matriz, la cual es ampliada para albergar la tercera coordenada de la tercera dimensión. La variable Z, profundidad, es igual a cero, pues al principio no existe volumen.

290: Se añade a X y a Y, Z/2 y Z/3, respectivamente, para que el cursor, cuando circule por el eje Z, recorra una imaginaria línea en perspectiva. Z/2 y Z/3 producen esa perspectiva.

360: Cada vez que fijamos un punto, las coordenadas de éste son almacenadas en la matriz D, siendo D(T,2) quien alberga la coordenada Z de cada punto.

340: Incluida en la rutina de borrar.

410-430: Parte de la rutina de dibujo de la figura cuyos datos contiene la matriz.

550 y 630: Adecuación de la rutina de grabación y carga de la matriz para que la nueva matriz sea grabada o cargada según proceda.

670-690: Rutina de rotación. Según el eje elegido se dirigirá a la línea 970, 980 ó 990, pues en todos los casos puede prescindir de las

coordenadas paralelas del eje elegido, dado que éstas no se transforman.

720-740: De la rutina de traslación. El único «problema» reside en que también debemos trasladar las coordenadas Z, y esto, evidentemente, no es ningún problema.

970-990: Producen la rotación del dibujo considerando tres dimensiones.

COMANDOS GRÁFICOS AMSTRAD

iempre puede resultar interesante estar provisto de una guía general de los comandos gráficos disponibles para los ordenadores CPC 6128, 664, 472 y 464. Cada uno de los comandos viene acompañado de una pequeña explicación de su funcionamiento y uso.

COMANDOS GRÁFICOS DE ACCIÓN

Los comandos incluidos dentro de esta categoría son las instrucciones BASIC de dibujo, cuyo efecto es la aparición de gráficos sobre la pantalla o el movimiento del cursor de gráficos. Se trata éste de un punto imaginario e invisible que existe sobre la pantalla, que será el principio de una línea. Dicho punto se mueve por toda la pantalla, incluso fuera de ella por medio del comando MOVE.

MOVE coordenada X, coordenada Y, tinta, modo de tinta

Desplaza el cursor de gráficos a la posición X,Y de la pantalla

elegida, siendo X la horizontal e Y la vertical. La tinta y el modo de tinta son opcionales y podrán ser utilizados o no según nuestras necesidades.

MOVER incremento X, incremento Y, tinta, modo tinta

MOVER convierte momentáneamente la posición del cursor de gráficos en el centro 0,0 de coordenadas y sitúa el cursor en la posición X,Y en relación con ese «origen».

El siguiente comando

DRAW coordenada X, coordenada Y, tinta, modo de tinta

dibuja una línea desde el actual cursor de gráficos hasta la posición especificada en las coordenadas X,Y. La tinta, un parámetro opcional, permite valores entre 0 y 15 en todos los modos, pero tengamos en cuenta que los valores mayores que 1 son ignorados en modo 2 y valores por encima de tres lo son en el modo 1.

El parámetro opcional del modo de tinta especifica cómo debe ser escrito el pixel en la pantalla cuando interaccione con algún otro. Un comando análogo a MOVER, llamado DRAWR, dibuja una línea desde el cursor de gráficos hasta la posición especificada por X,Y, tomando como coordenada 0,0 la actual posición del cursor.

DRAWR coordenada X, coordenada Y, tinta, modo de tinta

El comando PLOT dibuja un punto sobre la pantalla en las coordenadas X,Y especificadas. Tengamos en cuenta que el ancho del punto depende del modo de pantalla elegido. Con MODE 0 el punto será representado en la pantalla como cuatro pixels horizontales juntos y paralelos al eje X, en MODE 1 será representado por dos y, por último, en MODE 2 cada punto será representado en pantalla por un pixel individual.

PLOT coordenada X, coordenada Y, tinta, modo de tinta

Tal y como ocurre con los comandos MOVE y DRAW, PLOT también posee una palabra clave análoga llamada PLOTR, la cual dibuja un punto en pantalla en la posición X,Y de la misma, tomando como centro de coordenadas temporal (0,0) el de la posición actual del cursor gráfico.

CLG tinta

Para borrar gráficos de la pantalla. Si el color de tinta se especifica, el papel de gráficos (fondo) tomará ese color. Nótese que CLG borra tanto gráficos como texto. Para terminar con esta sección, veamos el comando FILL, que rellena el interior de un espacio cerrado con el número de pluma especificado.

FILL pluma

Este comando no trabaja en el CPC 464. Se debe usar con precaución, pues si la superficie a ser rellenada no está totalmente cerrada, toda la pantalla se invadirá del color de relleno.

COMANDOS DE TEXTO

Verdaderamente sólo existe un comando de texto en los gráficos Amstrad, y este es el comando PRINT. No debemos olvidar que cada carácter impreso sobre la pantalla es también un gráfico.

PRINT # número de canal, mensaje

PRINT se utiliza para escribir gráficos de la misma manera que se usa para escribir mensajes. Podemos imprimir, pues, una cadena literal.

PRINT "GRAN BIBLIOTECA AMSTRAD"

o una variable, por ejemplo:

PRINT A$

Para situar un mensaje en un lugar determinado empleamos el comando LOCATE, seguido de las coordenadas (COLUMNA, FILA), donde deseamos escribir la primera letra. Si deseáramos prescindir de LOCATE y dejar el mensaje en cualquier punto de la pantalla, utilizando MOVE para especificar el lugar, deberíamos emplear el comando TAG, abreviatura de *Text At Graphics* (Texto en gráficos). TAG fusiona momentáneamente la pantalla de texto y la de gráficos.

TAG número de canal

El número de canal es un parámetro opcional y sólo necesario en algunos casos determinados, como cuando estamos utilizando venta-

nas en nuestro programa. Ahora simplemente especificaremos la posición exacta del lugar donde deseamos colocar el mensaje

MOVE X,Y: PRINT"Probando el comando TAG"

Sin embargo, cuando terminemos la impresión debemos volver a distinguir entre las dos pantallas, la de gráficos y la de texto, para evitar que próximos mensajes sean situados en la posición del cursor de gráficos. La normalidad se restablece gracias a TAGOFF:

TAGOFF número de canal

Otros Comandos Gráficos

Afectan a los comandos de acción, que antes nombramos. Entre ellos, se encuentran todos aquellos comandos que especifican un color, modo o una relocalización del origen. Empecemos por los comandos de color. Tengamos en cuenta que las plumas por defecto para el fondo y la tinta de gráficos es 0 y 1 respectivamente. Para cambiarlos usaremos la palabra clave INK:

INK número de pluma, número de color

Una sentencia INK 1,16 tornaría al color rosa todos los caracteres de texto de la pantalla nada más encender el ordenador, así como de todos los gráficos dibujados hasta el momento. Del marco de la pantalla se ocupa BORDER:

BORDER color, color de intermitencia

Comando totalmente independiente de los colores y modo de pantalla, incluso en MODE 2. Podemos emplear directamente cualquiera de los 27 colores de nuestro Amstrad para colorearlo. El parámetro opcional indica el número de color que deberá presentarse intermitentemente con el color anteriormente introducido.

BORDER 0,13

El marco pasará del blanco al negro continuamente. La velocidad de intermitencia se manipula con el comando SPEED INK.

Por otra parte, al comando PAPER no podemos asignarle un número de color, por ejemplo, PAPER 25, sino que sólo podemos asig-

narle un número de pluma, cuyo color ha sido establecido a su vez mediante un comando INK.

```
INK 1,25
GRAPHICS PAPER 1
```

Si utilizamos los comandos GRAPHICS PAPER y GRAPHICS PEN, sólo influiremos en el color de los gráficos dibujados o por dibujar de la pantalla, afectando respectivamente al fondo y primer término. Estos comandos pueden ser muy útiles en algunas ocasiones.

MODE número (0-1-2)

Afecta a la apariencia del texto o gráficos en pantalla. Las características de los tres modos permitidos son las siguientes:

Modo	Pantalla gráficos	Pantalla texto	Plumas
0	160 × 200	20 × 25	16
1	320 × 200	40 × 25	4
2	640 × 200	80 × 25	2

Sin embargo, si tecleamos en cualquier modo

```
PLOT 320,200
```

aparece un punto en el centro de la pantalla. Si las dimensiones verticales, en todos los modos, son de 0 a 200, ¿por qué no aparece el punto en medio del margen superior de la pantalla? Esto es debido a que nuestro ordenador cuenta un punto por cada dos pixels; tecleemos:

```
PLOT 0,0
```

y un punto aparecerá en la esquina inferior izquierda. Ahora probemos:

```
PLOT 0,1
```

y no veremos ningún nuevo punto en la pantalla. Esto es debido a que las coordenadas (0,0) y (0,1) son las mismas. Pero, atención, porque algo también muy parecido ocurre con los puntos horizontales. En el MODE 0, cada punto queda representado en pantalla por cuatro puntos correlativos paralelos al eje X. Así pues, las coordenadas

(0,0), (1,0), (2,0) y (3,0) servirán para dibujar un punto en la esquina inferior izquierda de la pantalla. Por tanto, tenemos ocho coordenadas diferentes para llamar a un mismo punto en el modo 0.

En MODE 1 sólo (0,0) y (1,0) son el mismo punto en horizontal y en MODE 2, a cada punto le corresponde una sola coordenada en el eje X.

MASK número 0-255, modo de comienzo

Ejecutando la orden MASK delante de una sentencia de dibujo de una línea conseguimos que ésta se trace de manera discontinua. Para entender con facilidad su funcionamiento, dos ejemplos:

MASK 170 - Dibuja un punto sí, otro no, de la línea.
MASK 255 - Dibuja la línea completa, sin saltos.

El número 170 transcrito al código binario, que tan sólo usa ceros y unos, quedaría como 10101010. Cada uno supondrá un punto encendido de la línea y cada cero uno apagado. En el caso del 255, su representación binaria es 11111111, con lo cual la línea será dibujada por completo. Este comando no se haya implementado en los CPC 464.

ORIGIN coor.X, coor.Y, izquierda, derecha, arriba, abajo

Este comando se utiliza cuando deseamos cambiar el origen de la pantalla. Si tecleamos ORIGIN 100,100 obtendremos que las coordenadas del antiguo origen son ahora −100,−100. Las coordenadas de la esquina superior derecha, por ejemplo, serían 540,300. Al mismo tiempo, ORIGIN puede servirnos para crear una ventana gráfica. Por ejemplo:

```
10 MODE 1
20 ORIGIN 320,200,260,380,230,170
30 GRAPHICS PAPER 3
```

Tras efectuar RUN en este miniprograma, una ventana habrá quedado definida de 120 × 60 pixels en el centro de la pantalla. Quizás pensemos que esta afirmación no es cierta, pues, en realidad, parece no haber sucedido nada. Sin embargo, tecleemos como comando directo CLG y tras esto comprobaremos que, efectivamente, un cuadrado de color rojo aparece en el centro de la pantalla.

MÁS COMANDOS GRÁFICOS DE TEXTO

Estos afectan a la apariencia de un texto en la pantalla. Los comandos más relevantes en el BASIC de Amstrad son aquellos que cambian su color y su localización.

PAPER [# número de canal], <tinta>

Esta palabra clave indica el número de pluma, cuyo color será el fondo de las celdillas de texto. Excepto en BORDER, el número de tintas está limitado según el modo que se use.

PEN [# número de canal], número de pluma, modo de impresión

Este comando combina el color del texto mientras que los gráficos permanecen inalterables. Se pueden crear ventanas de texto utilizando el comando WINDOW. En este libro hemos entendido que esta instrucción apenas tiene relación con los gráficos de nuestro ordenador; aún así, empleamos ventanas en el programa de creación de caracteres definidos del capítulo cinco.

RAM DE VÍDEO

Se denomina RAM de Vídeo a aquella parte de la memoria de nuestro ordenador que se ocupa de controlar el estado de cada uno de los 128.000 puntos de los que se compone la pantalla. La RAM de vídeo ocupa 16384 bytes de memoria, que se sitúan entre la dirección 49152 y la 65535.

Los bytes de esta parte de la memoria representan los puntos de la pantalla pero, ¿cómo se disponen los puntos en la memoria? Para responder a esta pregunta no hay nada mejor que un ejemplo. Para empezar, desconectemos el ordenador y volvámoslo a enchufar. Ahora tecleemos directamente este pequeño programa:

```
MODE 2:FOR N=49152 TO 65535:POKE N,255:NEXT
```

La línea consigue encender todos los puntos de los que consta la pantalla, es decir, inicializa con 1 todos los bits de la RAM de vídeo. Los pixels se van encendiendo por líneas de texto: primero los puntos superiores y bajando poco a poco, línea a línea, hasta encender la totalidad de la superficie de la pantalla.

Como sabemos, cada línea de texto está formada por ocho líneas de puntos. Si poseemos 200 puntos de alto, son 200/8=25 líneas de texto.

Imaginemos ahora que colocamos una detrás de otra todas las líneas de texto, formando una grande compuesta por ocho finas líneas de pixels. La capacidad de la memoria gráfica es de 16 Kb, por lo que tocarán a 2 Kb por línea de pixels. Si ahora convertimos estas líneas en tan sólo una, colocando una detrás de la otra, obtendremos la configuración interna de la RAM de vídeo.

Ahora bien, activado el MODE 2 sólo pueden ser visualizados en la pantalla dos colores, fondo y tinta, o lo que es igual, punto activado o desactivado; al ordenador le basta con un bit para saber si el punto está encendido (un 1) o apagado (un 0). Para toda la pantalla se necesitarán tantos bits como puntos haya en ésta y como nuestro Amstrad posee una definición de 640*200=128.000 puntos, será la misma cantidad de bits los que se encargarán de el estado de esos puntos en la memoria.

Pero 128.000 bits son exactamente 16.000 bytes entonces, ¿por qué usamos 16.384 bytes, 384 más de los necesarios? Existen dos respuestas. En primer lugar, el procesador y el chip de vídeo prefieren calcular con kilobytes enteros, pues, según parece resulta más cómodo. Opinión que, naturalmente, nosotros respetamos. Pero además, el procesador necesita algunos bytes de reserva para realizar operaciones extra como algunos scrolls. Por todo ello, al final de cada bloque existen 48 bytes sin utilizar.

Todo parece aclarado pero, ¿en MODO 1 qué ocurre? Para empezar, pensemos que en esta modalidad se pueden usar hasta cuatro colores, por lo que el ordenador necesitará dos bits por cada punto para identificarlo. Las combinaciones posibles serán las siguientes: 00, 01, 10 y 11. De esta manera, estamos doblando el número de bits precisados, por lo que la cantidad de puntos que podemos representar se dividen por dos, aunque, eso sí, tienen el doble de ancho.

Ya hace tiempo que dejó de ser rumor para pasar a conocimiento de todos que, en modo 0, podemos utilizar hasta 16 colores diferentes, para lo cual el ordenador necesitará la mitad de un byte por cada punto, para poder reconocer el estado de cada uno de ellos. Medio byte son, evidentemente, cuatro bits, con lo que nuevamente el número de puntos representados se divide por dos. De la misma forma, poseen cuádruple tamaño que los del modo 2.

Pero volvamos a este modo donde quizás podamos ver todo más claro. Si cada bit en modo 2 determina el estado de un punto, y un byte está compuesto por ocho bits, estaremos de acuerdo en que el primer byte de la RAM de vídeo determinará el aspecto de los primeros ocho puntos.

Pensemos en la representación binaria del número 255:

1 1 1 1 1 1 1 1

Introduciendo POKE 49152,255 aparecerán, partiendo de la esquina superior izquierda de la pantalla, ocho puntos correspondientes cada uno a los valores 1 del número 255 en binario. Si el ejemplo hubiera sido POKE 49152, 170, cuya representación binaria es

1 0 1 0 1 0 1 0

obviamente aparecerían los primeros ocho puntos de la pantalla, uno encendido y otro apagado sucesivamente, tal y como lo indica su configuración binaria.

En la modalidad 1 cada punto necesita dos bits. El chip que produce la señal de televisión, diferencia entre la parte derecha del bit y la parte izquierda, es decir, divide el bit en dos medios byte (o NIBBLE). Tengamos el byte de número 170, que en sistema binario, como ya hemos visto, es 10101010. Ahora, asignaremos un número a cada bit:

```
1 0 1 0 1 0 1 0
| | | | | | | |
7 6 5 4 3 2 1 0
```

para pasar a continuación a dividir el byte en dos medios bytes, colocando el derecho debajo del izquierdo:

```
7 6 5 4        1 0 1 0
3 2 1 0        1 0 1 0
```

Cada par de bits encolumnados nos indican el color de cada uno de los cuatro puntos. En el ejemplo del 170 el primer punto tendría el color 3 (binario 11), el segundo 0 (binario 00), el tercero 3 y el cuarto 0.

Si el número hubiera sido el 11010011 los colores de los cuatro primeros puntos serían:

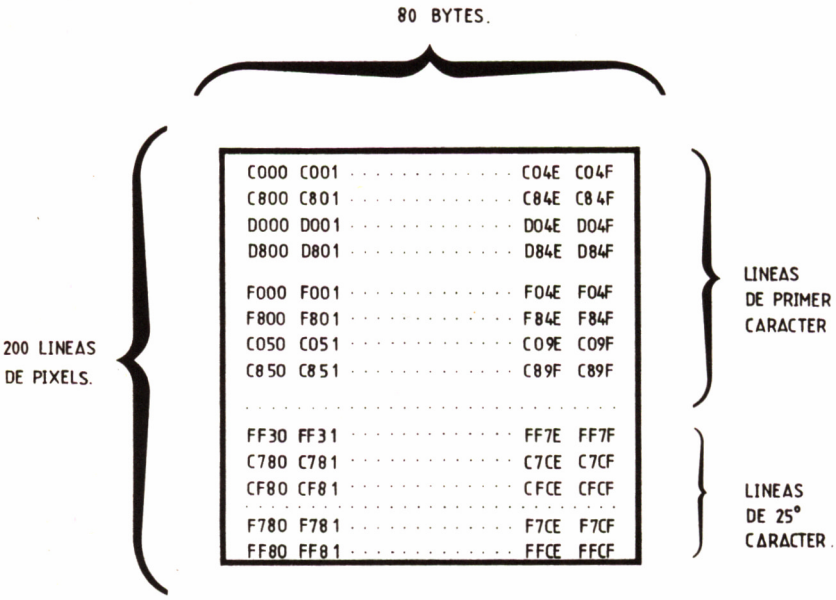
```
1 1 0 1
| | | |
0 0 1 1
```

el 2 para el primero (binario 10), 2 para el segundo, 1 para el tercero (binario 01) y 3 para el cuarto (binario 11).

En la modalidad 0 ocurre algo parecido. Coloquemos en otra disposición los números de orden de los bits de un byte; así:

7 6
3 2
5 4
1 0

Como sabemos, cada bit sólo puede definir dos puntos, cuyos colores quedan determinados por el valor de los números binarios de las posiciones 7632 y 5410, respectivamente. Este es todo el misterio que podía esconder la RAM de vídeo; así de fácil.



NOTAS

En este volumen de la GRAN BIBLIOTECA AMSTRAD emprenderemos un viaje alucinante por el mundo de los gráficos. Una imagen vale más que mil palabras, y la capacidad de los Amstrad para la creación de imágenes es una de sus características más fascinantes. Gracias a una gran cantidad de programas de ejemplo, que ponen en práctica los conocimientos teóricos expuestos en el texto, pasaremos del trazado de simples puntos en la pantalla, hasta el manejo de figuras tridimensionales.

GRAN BIBLIOTECA AMSTRAD

450 ptas.
(incluido IVA)
Precio en Canarias, Ceuta y Melilla: 435 ptas.