

N.º 6
Pras. 395
Canarias, Ceuta y Melilla, 375 ptas.
ESPECIAL

MICRO HOBBY

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES SINCLAIR Y COMPATIBLES

CONTROL DE SPRITES

*Una rutina superpotente
para mover bloques de gráficos*

MOVIMIENTO Y TECLADO

*Todo lo que necesitas saber
sobre el teclado y la animación de figuras*

TÉCNICAS DE MAPEADO

*Para construir
todos los decorados de tus juegos*

DETECCIÓN DE CHOQUES

*Controla cualquier tipo
de obstáculos igual
que en un programa comercial*

ESPECIAL GRÁFICOS
Con rutinas completamente
inéditas para crear
tus propios gráficos

Y ADEMÁS: Guía completa de todos los programas de diseño gráfico

¡JACK ATACA DE NUEVO!



1.200 Ptas.
(VERSION CASSETTE)

DISPONIBLE EN

Spectrum
Commodore
Amstrad
Amstrad Disk



ZAFIRO
CHIP

POCO RUIDO, MUCHAS NUECES

Director Editorial
José I. Gómez-Centurión

Director
Gabriel Nieto

Director de Microhobby
Domingo Gómez

Diseño
José María Oreja,

Redactores
Cristina Fernández,
Pedro Pérez

Secretaría Redacción
Carmen Santamaría

Colaboradores
Pablo Ariza, Rafael Márquez,
Enrique López, David López

Fotografía
Carlos Candel,
Chema Sacristán

Dibujos
F. L. Frontán, J. Igual,
M. López Moreno, A. Luis González Romero,
Vital Gariá, José Manuel Barco

Edita
HOBBY PRESS, S.A.

Presidente
María Andrino

Consejero Delegado
José I. Gómez-Centurión

Jefe de Publicidad
Mar Lumbreiras

Publicidad Barcelona
José Galán Cortés
Tels.: 303 10 22 - 313 71 76

Secretaría de Dirección
Pilar Aristizábal

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

**Redacción, Administración
y Publicidad**
Ctra. de Irún
km 12,400 (Fuencarral)
Tel.: 634 70 12
Telex: 49480 HOPR

Dto. Circulación
Carlos Peropadre

Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
Rotedic, S. A. Ctra. de Irún,
km 12,450 (MADRID)

Fotocomposición
Novocomp, S. A.
Nicolás Morales, 38-40

Fotomecánica
Graf
C/ Ezequiel Solana, 16

Depósito Legal:
M-36.598-1984

Representante para Argentina,
Chile, Uruguay y Paraguay, Cia.
Americana de ediciones, S.R.L.
Sud América 1.532. Tel.: 21 24 64.
1209 BUENOS AIRES (Argentina).

MICROHOBBY no se hace
necesariamente solidaria de las
opiniones vertidas por sus
colaboradores en los artículos
firmados. Reservados todos los
derechos.

Solicitado control
OJD

MICRO HOBBY

ESPECIAL MICROHOBBY • AÑO III • N.º 6 JUNIO 1987

ESPECIAL



4

TODO SOBRE LOS UDG. Conocer el modo de almacenamiento gráfico en el Spectrum es imprescindible para empezar a hacer un juego.

10

CONTROL DE SPRITES. Por fin una potente rutina que mueve gráficos, crea animación y te permite desplazar por pantalla simultáneamente distintos tipos de figuras.

32

MOVIMIENTO Y TECLADO. Todo lo que necesitas saber para poder utilizar el joystick o cualquier tipo de combinación de teclas en tus juegos. Y, además, con la posibilidad adicional de crear distintas secuencias de animación para tus personajes.

48

DETECCIÓN DE CHOQUES. Ahora puedes definir de un modo sencillo todos los obstáculos y elementos hostiles de tus juegos, mediante una rutina capaz de detectar cualquier tipo de choque o disparo.

58

TÉCNICAS DE MAPEADO. Crear los mapas y decorados de tus juegos será a partir de ahora una tarea casi rutinaria.

70

GUÍA DE UTILIDADES GRÁFICAS. Una completa guía de herramientas de programación para trabajar con los gráficos del Spectrum.

SUMARIO

TODO SOBRE LOS

Los UDG o gráficos definidos por el usuario, son unos caracteres similares a los del alfabeto al conectar el ordenador. Gracias a unas pequeñas operaciones podemos llegar a crear cualquier carácter o gráfico que nos sea de utilidad. Para conocer esta interesante posibilidad de nuestro ordenador, os explicamos con ejemplos cómo se realiza paso a paso.



CÓMO SE CREA UN UDG

Un carácter del ordenador es una cuadrícula de ocho por ocho pixels o puntos. Para diseñar el gráfico necesitamos unas cuadrículas como la que se muestra en la figura 1.

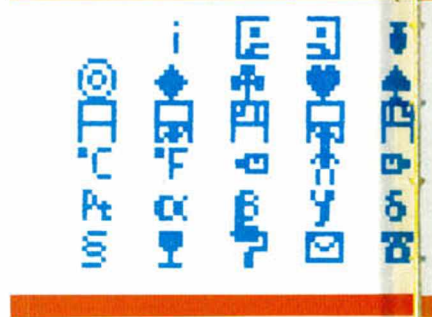
Sobre esta cuadrícula procederemos a rellenar los cuadros con un lápiz hasta obtener la forma deseada. En la figura 2 hemos creado una figura que simula un pequeño muñeco.

Cuando el gráfico esté terminado, podemos introducirlo en el ordenador de varias formas. La más sencilla y aconsejable si se está empezando, es la utilización de la función binaria de nuestro ordenador. Para ello en un papel apuntaremos los datos de la siguiente forma, empezando por el

primer cuadro de arriba a la izquierda. Apuntaremos un 1 cuando el cuadro en cuestión este pintado y con un 0 cuando el cuadro esté en blanco, siguiendo cuadro a cuadro hasta acabar la línea. Cuando hayamos completado la línea empezaremos una nueva serie de números con la siguiente y repetiremos la operación hasta completar las ocho líneas. Como se muestra en el ejemplo 1, los números 1 forman el gráfico que hemos creado.

Una vez realizada esta operación y con los datos que hemos apuntado, utilizaremos el comando POKE para introducir en memoria los distintos datos, teniendo en cuenta lo siguiente; para averiguar cuál es la dirección donde empieza un UDG, lo más sencillo es uti-

lizar PRINT USR "A", obteniendo distintas cantidades ya sea nuestro Spectrum de un 16 ó 48 k. También es importante que podemos pokear utilizando esta misma fórmula, por lo que para introducir nuestro gráfico, utilizaremos POKE USR "A", y después de la coma con el comando BIN colocaremos los ocho números que hemos apuntado en la primera línea; quedará de la forma que sigue POKE USR "A",BIN 0011100. Así hemos procedido a introducir el primer dato del carácter, pero necesitamos introducir el resto; por eso tenemos que sumar después de las comillas de la A y antes de la coma, un 1 (POKE USR "A"+1,BIN 00111000), y en el binario colocar los datos de la segunda línea y repetir la operación sumando luego 2, 3, 4, 5, 6 y 7 y las siguientes líneas; como podéis observar en el listado 1. Esta fórmula es sencilla pero además de lenta, ocupa demasiada cantidad de memoria, por lo que existen





otras fórmulas que realizan la misma operación, como podréis apreciar en el listado 2. En él hemos sustituido los números en binario por sus homólogos en decimal; en el listado 3, donde hemos creado un bucle que realiza la lectura de los datos, se introducen en su dirección correspondiente. Este último listado nos sirve además para introducir cuantos gráficos deseemos de una manera muy sencilla, basta con sustituir el carácter segundo del bucle FOR-NEXT, que es una **a** por el carácter del último gráfico que vayamos a realizar e incluir en DATAS los datos correspondientes al resto de gráficos.

Por último para utilizar dicho gráfico realizamos las mismas operaciones que con PRINT pero al introducirlo pondremos el cursor en modo gráfico y pulsaremos la letra en la que hemos introducido el gráfico, y que en el caso que estamos explicando es la **A**.

ciremos esta cantidad en la dirección 23676, luego con `PRINT 60000 - ((INT 60000/256)*256)` podremos el resultado en la dirección 23675 y cualquiera de los métodos anteriormente explicados para introducir el gráfico.

Para evitar esta limitación en su uso, una de las opciones por las que podemos optar es la de disponer varios bloques de UDG en la memoria, y usarlos según la conveniencia.

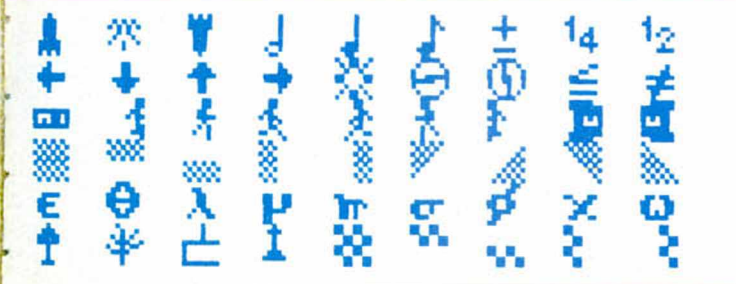
MÁS DE 21 UDG

Además de los UDG, podemos definir los 92 caracteres del ordenador, siguiendo las mismas pautas que para los gráficos definidos y posicionarlos en una dirección determinada de la memoria, y utilizando la variable del sistema llamada CHARS cuya dirección es la 23606 y la 23607, usaremos la misma fórmula que con los UDG pero con estas dos direcciones. En el listado 3 encontraremos un ejemplo práctico de cómo obtener un set de caracteres alternativo; en este programa se utiliza una fórmula distinta a las anteriormente explicadas, que consiste en que con ayuda de la función RANDOMIZE un número, este número se descompone automáticamente en el byte menos y más significativo, y se archiva en la dirección 23670 y 23671 respectivamente, por lo que luego sólo nos queda actualizar la variable del sistema CHARS, con el contenido de estas direcciones. Es importante recordar a la hora de utilizar un nuevo set de caracteres que los primeros 256 bytes no se pueden emplear por lo que debemos restar a la dirección donde hayamos archivado los gráficos esta cantidad. Es también impor-

CUÁNTOS GRÁFICOS PODEMOS CREAR

Es importante saber que los UDG, pueden ser posicionados en cualquier dirección de la memoria RAM, ya que en las variables del sistema se encuentran dos direcciones que a su vez contienen la dirección de memoria donde se encuentran los UDG. Esta variable es la denominada la UDG, y su dirección es la 23675 y la 23676.

Cuando la dirección esté decidida, por ejemplo la 60000, calcularemos las cantidades que hay que pokear, para ello utilizaremos la siguiente fórmula, `PRINT INT 60000/256`, e introdu-



tante recordar cuál es el contenido de la variable CHARS ya que si deseamos volver a utilizarlo deberemos teclear los siguientes pokes: POKE 23606,0 y POKE 23607,60.

Los gráficos que obtenemos con el listado 3, son los que se muestran en la figura.

CÓMO UTILIZAR LOS UDG PARA UN JUEGO

Como habréis podido comprobar, no es nada complicado realizar un UDG. Moverlo por la pantalla entraña algo más de dificultad. En principio debemos tener en cuenta algunas de las características del Spectrum.

La pantalla está compuesta por 32 columnas y 24 líneas, aunque no es fácil mover desde el Basic un gráfico por las dos líneas inferiores de pantalla. Cualquier desplazamiento por pantalla de un gráfico parecerá que va dando saltos, ya que no existe una posición intermedia donde imprimir el gráfico entre la columna 1 y la 2, o la línea 1 y 2. Con los atributos ocurre lo mismo, por lo que lo normal desde el basic es mover los gráficos sin atributos, dejando el color del papel y la tinta como están.

Con el ejemplo 4, obtenemos el gráfico de un muñeco y con las teclas Q, A, O y P desplazaremos el mismo por la pantalla.

Para actualizar el gráfico en la posición correspondiente es necesario, primero, borrar la posición anterior, de cuya misión se encarga la línea 80, borrando el gráfico antes de actualizar. Para ello usamos el comando OVER que imprime invirtiendo los pixels de cada carácter que coinciden

con otro ya en pantalla; así si imprimimos en OVER 1 un gráfico igual que esté será borrado.

Evitar que el gráfico parpadee se consigue colocando una línea que detecte si se ha pulsado una de las teclas correspondientes al movimiento, actuando como si de un filtro se tratase.

Con todas estas técnicas con ayuda del comando OVER conseguimos que lo que anteriormente estaba en la pantalla no se borre totalmente. Con PLOT, DRAW y CIRCLE, dibujamos algo en pantalla y al pasar por encima de las líneas, el dibujo sólo se borra cuando el gráfico está superpuesto pero al desplazarlo el dibujo se restablece.

UTILIZACIÓN DE CÓDIGOS ASC CON PRINT

El uso de los códigos ASC con PRINT nos ayuda a conseguir efectos difíciles como los que vamos a explicaros a continuación.

Por ejemplo, utilizando CHR \$ 8, conseguimos que el puntero de pantalla, se desplace hacia atrás una posición. Para obtener, por ejemplo, el gráfico de una mina en lugar de diseñarla podemos utilizar lo siguiente:

```
PRINT AT 10,10;"0";
OVER1;CHR $ 8;"X"
```

Existen teóricamente otros ASC, que realizan la operación de cursor arriba, abajo y a la derecha, pero por errores del sistema no funcionan correctamente y al utilizarlos aparece en pantalla una interrogación al igual que ocurre con el dedicado al DELETE y EDIT, ASC 12 y 7 respectivamente.

Los ASC al utilizarse des-



de el Código Máquina, desempeñan un papel igual que desde el Basic y los más utilizados son los siguientes:

ASC	CARÁCTER
6	PRINT coma
8	Cursor izquierda
13	ENTER
16	INK control
17	PAPER control
18	FLASH control
19	BRIGHT control
20	INVERSE control
21	OVER control
22	AT control
23	TAB control

AND, OR y XOR

Los operadores lógicos son de gran utilidad al manejar gráficos, pero por desgracia desde el Basic sólo es posible la utilización del AND y el OR para los condicionantes, y el XOR para utilizar en el OVER, las

operaciones que realizan son las siguientes:

AND sirve para comparar variables desde el Basic, pero su utilización desde el Código Máquina es mucho más completa, ya que realiza la operación bit a bit, entre el parámetro S y el registro A, en el cual se almacena el resultado. Para comprobar mejor su funcionamiento observar la operación en la tabla de verdad de la función AND:

A	AND S	= A
0	0	0
0	1	0
1	0	0
1	1	1

XOR, desde el Basic tiene un funcionamiento análogo al OVER 1, y en C/M, realiza la operación lógica entre el operando S y el contenido del registro A; la tabla de Verdad de la función XOR queda así:

A	XOR S	= A
0	0	0

0	1	1
1	0	1
1	1	0

OR, al igual que el AND, en el Basic se dedica exclusivamente a los condicionales, pero en Código Máquina realiza la operación lógica del operando y el contenido del registro A, utilizando las pautas que se muestran en la tabla de verdad:

A	OR S	= A
0	0	0
0	1	1
1	0	1
1	1	1

OTROS COMANDOS ÚTILES PARA LOS GRÁFICOS

ATTR (x,y), esta función devuelve el valor de los atributos de las coordenadas de x e y de pantalla; es imprescindible que los valores



FIG. 3

FIGURA 2

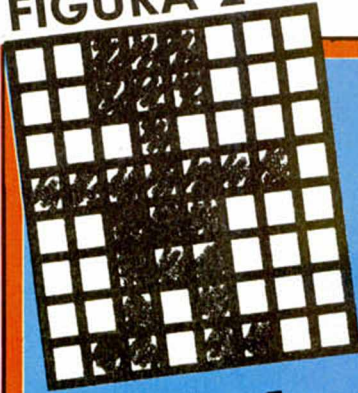
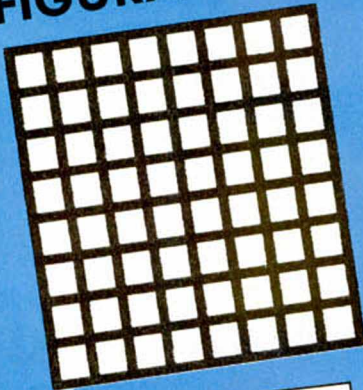


FIGURA 1



EJEMPLO 1

```
00111000
00111000
00010000
11111110
00111000
00111000
00101000
01101100
```

LISTADO 1

```
10 POKE USR "A",BIN 00111000
20 POKE USR "A"+1,BIN 00111000
30 POKE USR "A"+2,BIN 00010000
40 POKE USR "A"+3,BIN 11111110
50 POKE USR "A"+4,BIN 00111000
60 POKE USR "A"+5,BIN 00111000
70 POKE USR "A"+6,BIN 00101000
80 POKE USR "A"+7,BIN 01101100
```

LISTADO 2

```
10 POKE USR "A",56
20 POKE USR "A"+1,56
30 POKE USR "A"+2,16
40 POKE USR "A"+3,254
50 POKE USR "A"+4,56
60 POKE USR "A"+5,56
70 POKE USR "A"+6,40
80 POKE USR "A"+7,108
```

LISTADO 3

```
10 FOR a=USR "a" TO USR "a"+7
20 READ b: POKE a,b
30 NEXT a
40 DATA 56,56,16,254,56,56,40,
108
```

LISTADO 3a

```
10 CLEAR 49999: LOAD ""CODE 50
000
20 RANDOMIZE 49744
30 POKE 23606,PEEK 23670
40 POKE 23607,PEEK 23671
50 FOR a=32 TO 127
60 PRINT CHR$ a;" ";
70 NEXT a
```

de x e y estén cerrados entre paréntesis ya que sino produciríamos un error de sintaxis. Para utilizarlo probar a imprimir en pantalla con papel azul y tinta blanca un carácter (PRINT PAPER 1;INK 7;AT 10,10;"R"), y después usando ATTR preguntar qué colores están en esas coordenadas (PRINT ATTR(10,10)), el valor devuelto por el ordenador será el resultado del color (15). Este número en binario nos dará el dato de color del papel y tinta, y el estado del brillo y flash. El 7 bit por la derecha es el que indica el estado del flash, siendo 0 para desactivado y 1 para activado. El

sexto bit nos expresa el estado del brillo, siguiendo las mismas pautas del flash; del quinto al tercero son los dedicados al color del papel, y del segundo al cero, el color de la tinta. Es aconsejable utilizar la fórmula siguiente para saber qué número corresponde a un estado de atributos.

NÚM. TINTA + NÚM. PAPEL * 8 + NÚM. BRILLO * 64 + NÚM. FLASH * 128

CHR \$n, introduciendo en n un número entre 32 y 255, obtendremos cualquiera de los caracteres ASCII del Spectrum y si el número es el comprendido entre 6 y el 23 son destinados al manejo de pantalla, siendo inuti-

lizables los comprendidos entre 0 y 5 ó 24 y 31.

PEEK n, con este comando podemos averiguar el contenido de una celdilla de memoria, siendo este número entre 0 y 65535.

POINT (x,y), si el resultado de esta operación es 1, es que ese pixel está activado de color de tinta, y si es 0 no lo está. Este comando está limitado a la parte superior de la pantalla, siendo imposible con su uso comprobar el estado de una coordenada dentro de la zona de pantalla destinada a los mensajes.

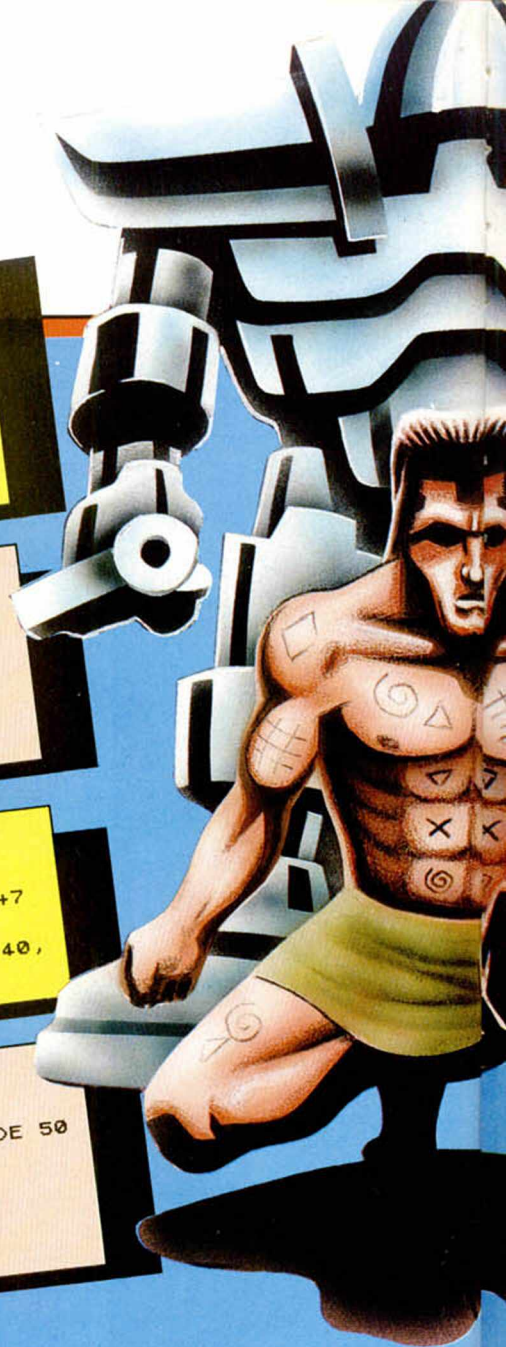
SCREEN \$ (x,y), del mismo modo que podemos averiguar cuál es el color de

una coordenada de la pantalla y el carácter que se encuentra en esta dirección. Al realizar esta operación si el resultado no es 0 no es que no exista en esa posición un carácter, sino que no es reconocido como tal por el ordenador.

USR \$, muy útil a la hora de averiguar la dirección de comienzo de cualquiera de los caracteres definidos por el usuario.

BORDER n, sirve para dar color al borde siendo éste un número comprendido entre 0 y 7.

BRIGHT n, este comando sólo puede ser utilizado activado (1) o desactivado (0), consiguiendo en una y





TODO SOBRE LOS UDG 9

LISTADO 3b

LINEA DATOS CONTROL

1	0000000000000000000000008	8	375
2	00080808080007F404C40		958
3	404F587FFFE02323202F2		568
4	1AFE003C183C3C3C1800		656
5	10383838387C7C540054		828
6	AA2844921000547C7C38		280
7	3838381008080808080838		384
8	4830080808080808080830		302
9	080C0A08083878301010		680
10	7C1010007C0040C0444C		608
11	541E040040C040C524408		612
12	1E00C020442CD41E0408		348
13	1818007E001818003C42		958
14	99A5A599423C0010387C		1244
15	FE7C381000383806FED6		1098
16	0010006CFEFEFE10380010		792
17	0010387CFEFE181818		436
18	307E7E30100000181818		468
19	7E3C18000007E7E0C0800		316
20	180000080C3C245A993C42		1074
21	995A24C3C3C245A993C42		1232
22	819FF981423C3C529199		
23	99894A3C000618607E00	676	
24	7E0000040C7E187E3020	498	
25	006018067E007E000010	394	
26	001020423C00FF818181	816	
27	FF818181FF818181FF98	1694	
28	BDD9FFA5E585FF818181	1830	
29	FF818181FFD9B09BFFA5	1878	
30	A7A1FF8181810000FF80	1366	
31	ADFF00000303060B0203	456	
32	07010C0C181E08142404	154	
33	303060B02830C8040C0C	684	
34	060D140C132030301878	342	
35	10282420C0C0500040C0	1068	
36	E0807C7F3F3F3E0609	1082	
37	3EFEFC8C8CFCE0003844	1463	
38	090E58F060C000003844	763	
39	403C083000CED0101010	642	
40	0E0000DED0101C101000	520	
41	00001F75711F000001818	340	
42	3C5A182424240000F8AE	704	
43	8EF80000AA55AA55AA55	1155	
44	AA55AA55AA5500000000	765	
45	00000000AA55AA55A050	750	
46	A050A050A0500A050A05	756	
47	0A050A05AA54A850A040	756	
48	8000000102050A152A55	294	
49	AA552A150A0502018040	528	
50	A050A854AA5507033D44	886	
51	44443800384444443810	524	
52	3810006050545E444600	580	
53	000072CCCCC72000038	896	
54	6478646478600044C4C38	844	
55	18181830003860384C4C	480	
56	38000003C6078603C00	488	

DUMP: 50000 N.º BYTES 768

57	386CC6FEC66C38000030	1026
58	1818182C460000007C6A68	390
59	667C606000000000C87000	944
60	68000000007EC8C87000	742
61	060C78CCCC7860C00000	954
62	C62C1838660000006CC6	730
63	C6D67C00FEC6603060C6	1426
64	FE00007CC6C6C66CEE00	1318
65	003C403844380478007C	552
66	7C7C3810107C7CFF7D01	965
67	0F18181800FFC3A59981	984
68	FF0000FFDB3C66667E00	1119
69	10387C7C101010382012	474
70	543996781010080808FF	722
71	808080FF103810101038	615
72	7C00CCCC3333CCCC3333	1144
73	CCCC3333000000000000	510
74	0000CCCC3333C03030C0	990
75	C0C0303000C0C03030C0	534
76	0303CCCC3030C0303000	894
77	0000030300C033330000	132

LISTADO 4

```
10 FOR a=USR "a" TO USR "a"+7
20 READ b: POKE a,b
30 NEXT a
40 DATA 56,56,16,254,56,56,40,
108
50 LET x=10: LET y=10
55 GO SUB 9000
60 LET a=CODE INKEY$
70 IF a<>97 AND a<>111 AND a<>
113 AND a<>112 THEN GO TO 60
80 PRINT OVER 1;AT y,x;" "
90 LET x=x+(a=112 AND y<31)-(a=
=111 AND x>0)
100 LET y=y+(a=97 AND y<21)-(a=
113 AND y>0)
110 PRINT OVER 1;AT y,x;" "
120 GO TO 60
9000 BORDER 1: INK 7: PAPER 1: C
LS
9010 DRAW 255,0
9020 PLOT 30,0: DRAW 0,30: DRAW
30,0: DRAW 0,-30
9030 CIRCLE 100,100,30
9040 RETURN
```

otra forma dar brillo a los atributos.

DATA n,n,n,n, todos los datos que aparezcan detrás de esta sentencia, serán leídos como datos, al utilizar la función READ.

FLASH n, su funcionamiento es igual que el del brillo, pero realizando un intercambio de atributos entre el papel y la tinta, consiguiendo así el efecto de parpadeo.

INK n, el color del atributo de tinta es controlado por este comando permitiendo el uso de cualquiera de los ocho colores del Spectrum además del color 8, para transparente y 9 para contraste.

INVERSE n, controla la inversión de caracteres que se van a imprimir. En caso de utilizar con 1 los caracteres se imprimen en video inverso, y si es 0 en video normal.

OVER n, realiza un intercambio al estar activada esta función, entre los datos que estaban en pantalla y los que posteriormente sean utilizados.

PAPER n, su funcionamiento es completamente igual al del comando INK, pero controlando el color de fondo.

POKE n, sirve para introducir valores comprendidos entre 0 y 255 en las direcciones desde la 16384 y

65535, con ayuda de POKE también podemos realizar dibujos en pantalla tanto de gráficos como de atributos.

PRINT, es el comando sin lugar a dudas más utilizado en Basic; con él podemos hacer que aparezca en pantalla cualquier carácter y su funcionamiento puede ser de distintas maneras.

AT x,y; si es utilizado con AT, posicionamos las coordenadas donde van a ser impresos en pantalla uno o varios caracteres.

TAB x; con él sólo podemos indicar el desplaza-

miento desde el puntero de pantalla tantos números como se indiquen en x.

“,”: la coma después de PRINT y sin estar entrecomillada realiza una tabulación de 16 caracteres, desde el puntero.

“;”: el punto y coma sirve para indicar al ordenador que todo lo que está unido por ellas se imprime seguidamente.

: la comilla indica al ordenador que debe saltar de línea para que lo que se exprese a continuación sea impreso en la línea siguiente.

CONTROL

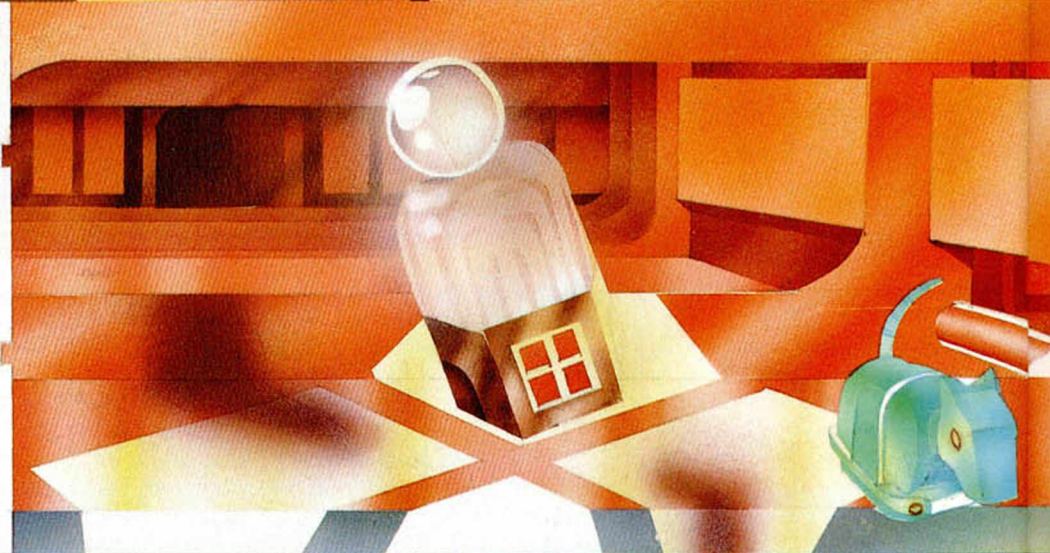
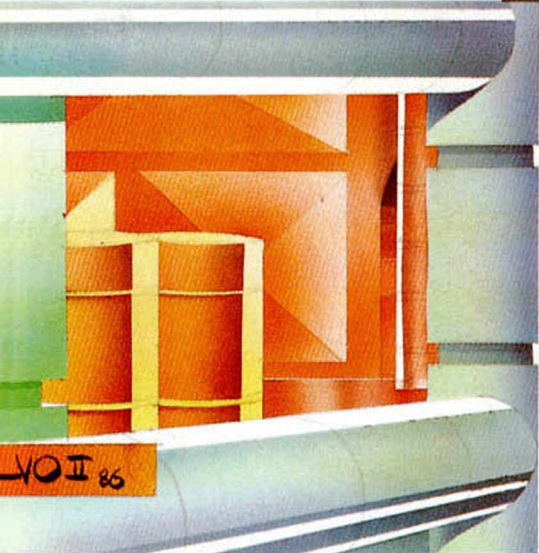
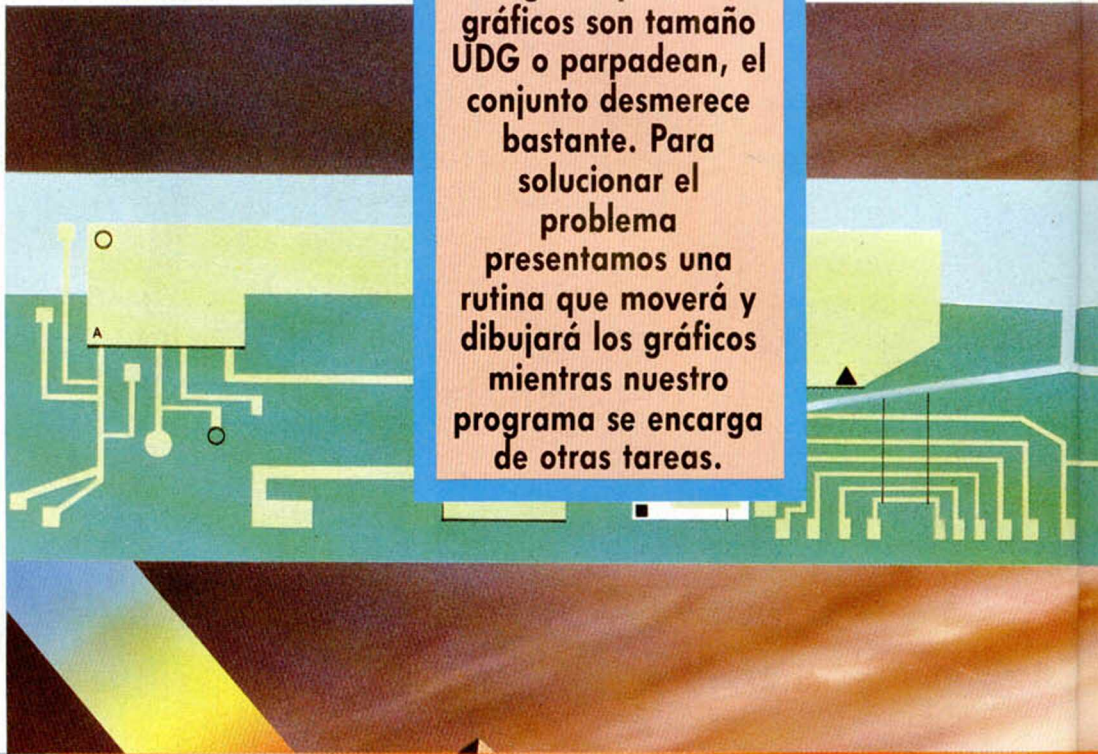
DE

En primer lugar, para los no iniciados, definiremos qué es lo que entendemos por un sprite.

Llamaremos sprite a un gráfico, animado o no, que se mueve por la pantalla en una dirección determinada. Este gráfico deberá pasar por delante de lo que haya dibujado en pantalla sin borrarlo, y si se cruza con otro, uno de los dos pasará por delante de otro. Por esto cada sprite tiene asignada una prioridad, de tal forma que al cruzarse dos sprites, pasará por delante aquél cuya prioridad sea mayor. Evidentemente, no puede haber dos sprites con la misma prioridad.

En otros ordenadores, co-

Sin duda, uno de los factores más decisivos para que un programa tenga éxito es su presentación. Si la idea es muy original, pero los gráficos son tamaño UDG o parpadean, el conjunto desmerece bastante. Para solucionar el problema presentamos una rutina que moverá y dibujará los gráficos mientras nuestro programa se encarga de otras tareas.

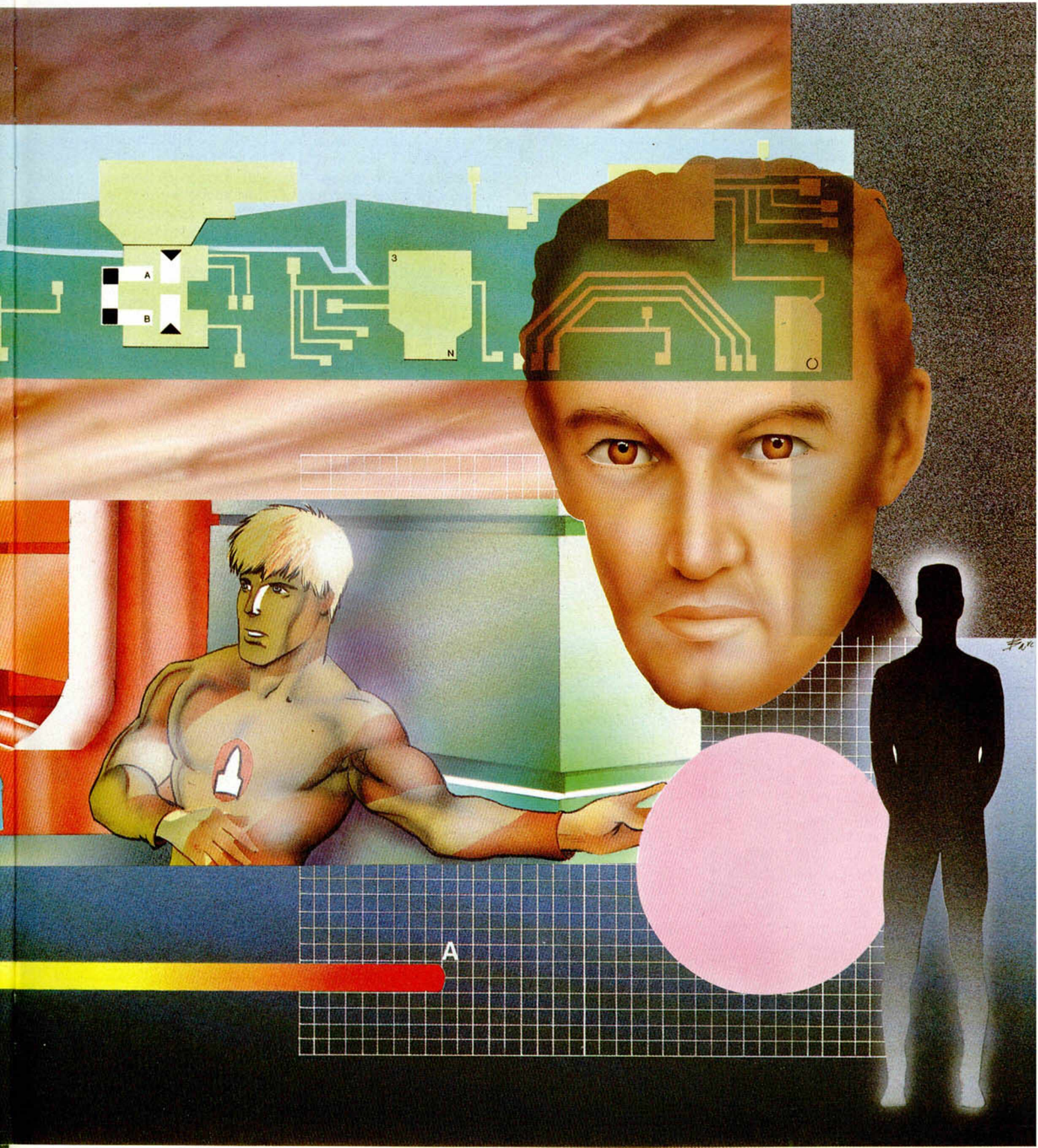


mo el Commodore 64, los sprites se generan por hardware, con lo cual no se consume tiempo de programa. Pero lamentablemente, en el Spectrum no existe hardware para la creación de sprites, así que no tendremos más remedio que crearlos por software. Para ello, utilizaremos un programa en

SPRITES

SPRITES 11

Pablo ARIZA



Código Máquina que se ejecutará automáticamente gracias a las interrupciones. Sobre interrupciones ya se ha hablado bastante en números anteriores de esta revista, así que bastará con que digamos que sirven para que una subrutina escrita en Código Máquina se ejecute automáticamente 50 veces por segundo. Cuando termina, se devuelve el control al programa que estaba ejecutándose antes de producirse el salto a la subrutina. De esta forma, podemos lograr el efecto de que se están efectuando dos tareas a la vez. En nuestro caso, una de las tareas será el dibujo de los sprites, y la otra cualquier programa escrito por nosotros, ya sea en Basic o Código Máquina.

Normalmente, las subrutinas que se ejecutan por interrupciones suelen ser muy cortas, de tal forma que aparentemente la velocidad del programa principal es la misma que tendría si no hubiera interrupciones. Sin embargo, el dibujo de gráficos y el manejo de la pantalla en general, es siempre bastante lento, y cuando pongamos en marcha los sprites, notaremos un descenso enorme en la velocidad de ejecución de nuestros programas. Es el precio que hay que pagar por no tener el hardware adecuado. A pesar de todo, si programamos en Código Máquina, tendremos suficiente velocidad para bastantes cosas, pues no hay que olvidar que nuestro programa ya no se tendrá que preocupar de dibujar los gráficos, que suele ser el proceso que más tiempo consume en la ejecución de un programa, sobre todo en los juegos arcade.

El método que vamos a utilizar para dibujar los gráficos es el de la máscara. En este método, cada gráfico

se compone en realidad de dos gráficos: el gráfico propiamente dicho y su máscara. La máscara es un gráfico adicional, de las mismas dimensiones que el primero, y que nos facilita información acerca de la forma de éste. Esto sirve para que cuando el sprite pase por delante de algo, no se mezcle su imagen con la del fondo ni se borre un rectángulo del fondo alrededor del sprite, como suele ocurrir en los métodos tradicionales de dibujo de sprites. Es el método que se suele usar en los juegos tridimensionales, como «Knight Lore», y en algunos otros, como «Everyone's Wally». Tiene el inconveniente de ocupar más memoria (cada gráfico ocupa el doble, pues necesita de su máscara) y de ser más lento el proceso de dibujo, pero los resultados obtenidos son mucho más espectaculares. Para los que leyeron los artículos sobre el sistema Filmation, publicados en los números 96, 97, 99 y 100 de MICROHOBBY, en los que se utilizaban también máscaras, advertimos que en esta rutina vamos a utilizar un método ligeramente distinto. Para evitar el paso de inversión de la máscara descrito en dichos artículos, nosotros almacenaremos la máscara ya invertida, con lo que ahorramos tiempo a la hora de dibujar el gráfico. Por tanto, si tenéis hechos de antemano gráficos pensados para la rutina presentada en dichos artículos y los queréis usar con ésta, deberéis invertir todos los bytes de la máscara. La forma de hacerlo es bien sencilla. Si por ejemplo, tenéis una máscara en la dirección 400000 que ocupa 32 bytes, bastará con que hagáis:

```
FOR X=400000 TO 40031
:POKE X,255-PEEK X:
NEXT X
```

Si vais a crear gráficos nuevos, específicamente para esta rutina, la forma de crear la máscara, una vez realizado el gráfico, podría ser así:

Observamos la figura 1. En primer lugar, alrededor del gráfico terminado (parte A), dibujamos una línea que marque su contorno, obteniendo la parte B. A continuación, rellenamos desde aquí hacia afuera todo el rectángulo que contiene el gráfico, obteniendo la parte C. Por último, borramos el gráfico original (naturalmente, antes lo habremos grabado), y nos queda la parte D, que es ya la máscara. Si el gráfico tuviera agujeros internos (por ejemplo, si dibujamos una rosquilla), deberemos hacer el mismo proceso con los agujeros. Lo que representa la máscara son las zonas del gráfico a través de las que se puede ver el fondo.

TABLA DE SPRITES

Para conseguir que nuestro sprite se mueva como nosotros queremos, habremos de comunicarle de algún modo a la rutina de control todos los datos necesarios acerca de él. Con este fin, vamos a crear una tabla de sprites, en la que iremos poniendo la posición, dimensión, y demás datos de cada uno de ellos. La ru-

tina está pensada para manejar un máximo de 17 sprites, que numeraremos del 0 al 16 (este número servirá asimismo para indicar la prioridad del sprite). Como veremos enseguida, para especificar todos los datos de un sprite necesitaremos 17 bytes, o lo que es lo mismo, un total de $17 \times 17 = 289$ bytes, que los colocaremos en la dirección 530000. De esta forma, los datos del primer sprite estarán a partir de la dirección 530000, los del segundo en la 53017, etc.

Vamos a ver ahora qué es lo que deberemos meter en cada uno de esos grupos de 17 bytes. Tenemos un resumen en la figura 2. Los dos primeros servirán para indicar la dirección de memoria en la que se encuentra el gráfico. Siempre, a continuación del gráfico propiamente dicho, se debe encontrar su máscara. Si el sprite representa un objeto o un ser animado, deberá constar de varios gráficos parecidos, pero distintos para lograr el efecto de animación. En este caso, todos estos gráficos deben estar uno a continuación del otro, y cada uno con su máscara detrás.

El tercer y cuarto byte indican las dimensiones del sprite. El tercero indicará el ancho, dado en caracteres, y el cuarto el alto, dado en píxeles. Todos los gráficos de

FIGURA 1

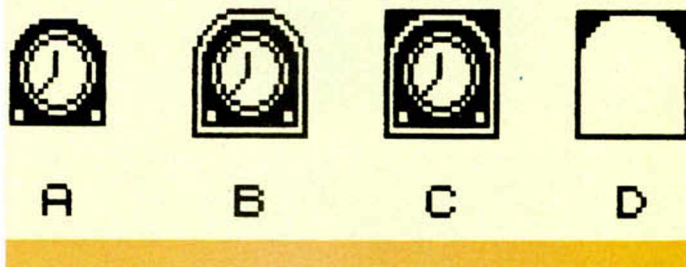


Fig. 2

TABLA DE DATOS DE LOS SPRITES

Dirección relativa	Contenido
DIR+0	Dirección del primer gráfico.
DIR+2	Ancho en caracteres.
DIR+3	Alto en scans (pixels).
DIR+4	Coordenada X.
DIR+6	Coordenada Y.
DIR+8	Número de fases de animación.
DIR+9	Color. Bit 3: Animación. Bit 4: Movimiento.
DIR+10	Incremento X. DIR. Tabla de trayectoria.
DIR+11	Incremento Y.
DIR+12	Contador de animación.
DIR+13	Contador de trayectoria.
DIR+15	Memoria ocupada por cada gráfico.

las distintas fases de un único sprite y sus respectivas máscaras deberán tener las mismas dimensiones.

Los bytes quinto y sexto especifican la coordenada X del sprite, mientras que los séptimo y octavo especifican la Y. Ambas serán números entre -32768 y +32767. Esto nos permite crear un sprite inicialmente en unas coordenadas que no existan en la pantalla real del ordenador, y hacer que aparezca posteriormente. Si la coordenada X está entre 32512 y 32767 (o lo que es lo mismo, si el byte sexto es 127), se considerará al sprite como desactivado y ni se le moverá ni se le dibujará. A diferencia del Basic, la coordenada Y comienza

por 0 en la parte superior y va aumentando hacia abajo.

El noveno byte nos dice cuántas fases de animación tiene el sprite. Si se trata de un sprite inanimado, pondremos una fase.

El décimo indica varias cosas. En primer lugar, indica el color del sprite. Los tres primeros bits indican el color de la tinta, y el séptimo bit (el bit 6) indica el nivel de brillo. Los sprites no tienen color de papel propio; adoptan el del fondo por el que pasan. Por otra parte, el cuarto bit (el bit 3) indica, en el caso de sprites animados, de qué tipo de animación se trata. Para sprites inanimados, su valor es indiferente. Si vale 0 la anima-

1 2 3 4 5 1 2 3 4 5 1 2 3
4 5...

Si, por el contrario, especificamos animación de adelante-atrás, obtendremos la secuencia:

1 2 3 4 5 4 3 2 1 2 3 4 5
4 3 2 1 2...

Por último, en este décimo byte, está también, en el bit 4 el indicador de línea recta (con un 0) y trayectoria compleja (con un 1). La opción de línea recta se usará, como su propio nombre indica, para sprites que se muevan en línea recta, sea horizontal, vertical o diagonal. Sin embargo, habrá casos en que preferiremos que nuestro sprite siga una trayectoria preestablecida dis-

En el caso de un movimiento en línea recta, estos bytes indican los incrementos de X e Y respectivamente. Dichos incrementos son números entre -128 y +127, que se suman cada vez a las coordenadas del sprite para producir el movimiento. Cuanto mayor sea su valor absoluto, mayor será la velocidad con que se mueva el sprite, pero menor su suavidad. Los incrementos más normales están entre -4 y +4. En la coordenada X, un incremento positivo producirá un movimiento hacia la derecha, y negativo, hacia la izquierda. En la Y, un incremento positivo producirá un movimiento hacia abajo, y negativo hacia arriba.

En el caso de un movimiento de trayectoria prefijada, estos dos bytes indican la dirección donde se encuentran los datos acerca de la misma. En esta dirección pondremos dichos datos de la siguiente forma: para delimitar la trayectoria, se colocan grupos de dos bytes que indican los incrementos X e Y a efectuar en cada momento. Pero además, puede que queramos que el sprite cambie su forma durante el movimiento. Por ejemplo, si queremos dibujar una nave girando, no bastará con dar los incrementos que delimiten la trayectoria de giro, sino que, además, deberemos ir dibujando un gráfico un poco girado respecto del anterior para que parezca que la nave está realmente girando. Para ello, entre los grupos de dos bytes que delimitan el movimiento, colocaremos un 126, y a continuación, la dirección de memoria donde se encuentra el nuevo gráfico. Para terminar la trayectoria, colocaremos un 127 y ésta se repetirá desde el principio.

El byte decimotercero es utilizado como contador pa-

Fig. 3

Gráficos usados en la demostración

Dirección	Gráfico	Dimensiones (en caracteres)	Número de fases
47300	COHETE	2x2	1
47364	RELOJ	3x3	8
48516	FLECHA →	2x2	1
48580	FLECHA ↗	2x2	1
48644	FLECHA ↑	2x2	1
48708	FLECHA ↖	2x2	1
48772	FLECHA ←	2x2	1
48836	FLECHA ↙	2x2	1
48900	FLECHA ↓	2x2	1
48964	FLECHA ↘	2x2	1

ción será cíclica, mientras que si vale 1, será una animación de adelante-atrás. La primera es la que usaremos, por ejemplo, para un helicóptero cuyas hélices giran. La segunda la usaremos, por ejemplo, para un pez que mueve la cola. Si tenemos cinco fases de animación, que podemos numerar del 1 al 5, y especificamos animación cíclica, las fases se irán repitiendo de esta forma:

tinta de la línea recta, por ejemplo, un satélite que gira en círculo alrededor de un planeta. Para esto sirve la opción de trayectoria. La forma de establecer la trayectoria la vamos a ver a continuación.

El significado de los bytes undécimo y decimosegundo no es el mismo si el sprite se mueve en línea recta o siguiendo una trayectoria especial.

ra la animación, y deberemos inicializarlo a 0.

Los bytes decimocuarto y decimoquinto se usan como contador para la trayectoria. Si estamos utilizando un movimiento en trayectoria preestablecida, deberemos inicializarlos con los mismos valores que los bytes undécimo y decimosegundo, es decir, con la dirección de los datos de la trayectoria. Si el movimiento es rectilíneo, estos bytes no se usan.

Por último, los bytes decimosexto y decimoséptimo indican la cantidad de memoria que ocupa cada gráfico del sprite, ya sea gráfico propiamente dicho o máscara, y se calcula multiplicando el ancho en caracteres por el alto de pixels. Este valor es para el uso interno de la rutina, y no habría sido necesario incluirlo en la tabla de sprites, pues se puede calcular a partir de los contenidos de los bytes tercero y cuarto, pero se ha incluido por razones de velocidad.

EL PROGRAMA

Antes de profundizar más en el funcionamiento de la rutina, sería preferible teclear ahora el programa para ver los resultados y comprender así mejor cómo funciona. En primer lugar, habremos de teclear el listado 1 y grabarlo en cinta con autoejecución en la línea 9999. A continuación, y con ayuda del Cargador Universal de Código Máquina, publicado innumerables veces en Microhobby y Micromanía, teclearemos el listado 2 y lo grabaremos en cinta, a continuación, del Basic con el nombre «Sprite.Code Data», indicando como dirección de comienzo 47140, y como longitud 1888. Ahora podemos teclear el listado 3 en el Cargador Universal de Código

Máquina, o el listado 4 con un ensamblador, preferiblemente el Gens, y ensamblar. En ambos casos, grabaremos el resultado a continuación de lo anterior como «Sprite.Code USR», indicando como dirección la 61953, y como longitud 1495. Ahora, ya lo tenemos todo listo. Rebobinemos y carguemoslo todo. El verdadero programa controlador de sprites es el del listado 3 ó el 4. El Basic y el listado 2 sirven para efectuar una demostración de sus posibilidades, que son las que veremos al

crear en la tabla anteriormente explicada. Es necesario advertir que si queremos, por ejemplo, crear tres sprites, estos deberán ser los números 0, 1 y 2, es decir, los tres primeros. El programa Basic utiliza una subrutina que lee los datos de líneas DATA y hace los POKES pertinentes, pero cada uno puede hacerse su propia subrutina como quiera, teniendo en cuenta cuáles son los datos que hay que indicar, y que ya han sido explicados. Una vez creados los datos de los sprites, activa-

inicializada al activarse las interrupciones, así que si queremos que nuestros sprites se muevan sobre algún fondo o dibujo, tendremos que dibujarlo en la pantalla, justo antes de hacer el RANDOMIZE USR 61953. Una consecuencia del uso de una pantalla en memoria, es que no podremos dibujar ni escribir nada en la pantalla del ordenador, porque las interrupciones se encargan de copiar constantemente la pantalla de memoria en la pantalla del ordenador, así que nada más es-



finalizar la carga. Dicha demostración se compone de siete ejemplos, desde el más simple hasta el más complejo. Estos ejemplos no están pensados solamente para que veamos cómo se mueven los sprites, sino para que, estudiando el programa Basic, podamos comprender mejor cómo manejarlos desde nuestros propios programas. En él podemos ver cuál es la secuencia a realizar para que los sprites comiencen a moverse: en primer lugar, se introducen los datos de cada uno de los sprites que queremos

remos las interrupciones con RANDOMIZE USR 61953 e indicaremos el programa controlador de sprites, cuántos sprites estamos utilizando, haciendo POKE 23681, N, donde N es el número de sprites. A partir de ese momento, los sprites comenzarán a moverse.

Para evitar parpadeos y otras fealdades varias, todos los dibujos se hacen en una copia de la pantalla que se encuentra en otra dirección de memoria, y luego el resultado se vuelca a la pantalla del ordenador. Esta copia de la pantalla es

cribir algo, se borrará automáticamente. Para poder escribir algo, deberemos hacerlo antes de activar las interrupciones, o escribiendo directamente en la pantalla de memoria, para lo cual veremos más adelante dónde y cómo está colocada.

Independientemente de la rutina de sprites, hay una subrutina que puede resultar muy útil en algunos casos. Se trata de la que comprueba el choque entre dos sprites. La forma de utilizarla es la siguiente: POKE 23729,A:POKE 23728,B:

LET C=USR 62082, donde A y B son los códigos de dos sprites, y C terminará valiendo 1 si los dos sprites están chocando o 0 si no lo están. Si lo que queremos es averiguar si un sprite está chocando con cualquier otro, sin importarnos cuál, sustituiremos B por un 255.

Comentemos un poco los ejemplos de la demostración, porque ilustran bastante bien las características de los sprites.

El primer ejemplo es el más sencillo. Un único sprite de color rojo y sin anima-

oportunos, en este caso, las direcciones 53004 Y 53006.

En el tercer ejemplo, vemos como actúa la máscara para producir un efecto muy convincente de que el sprite está por delante del fondo. Desgraciadamente, podemos comprobar también cómo en el Spectrum es imposible evitar la famosa mezcla de colores, pero ésta es reducida al mínimo posible.

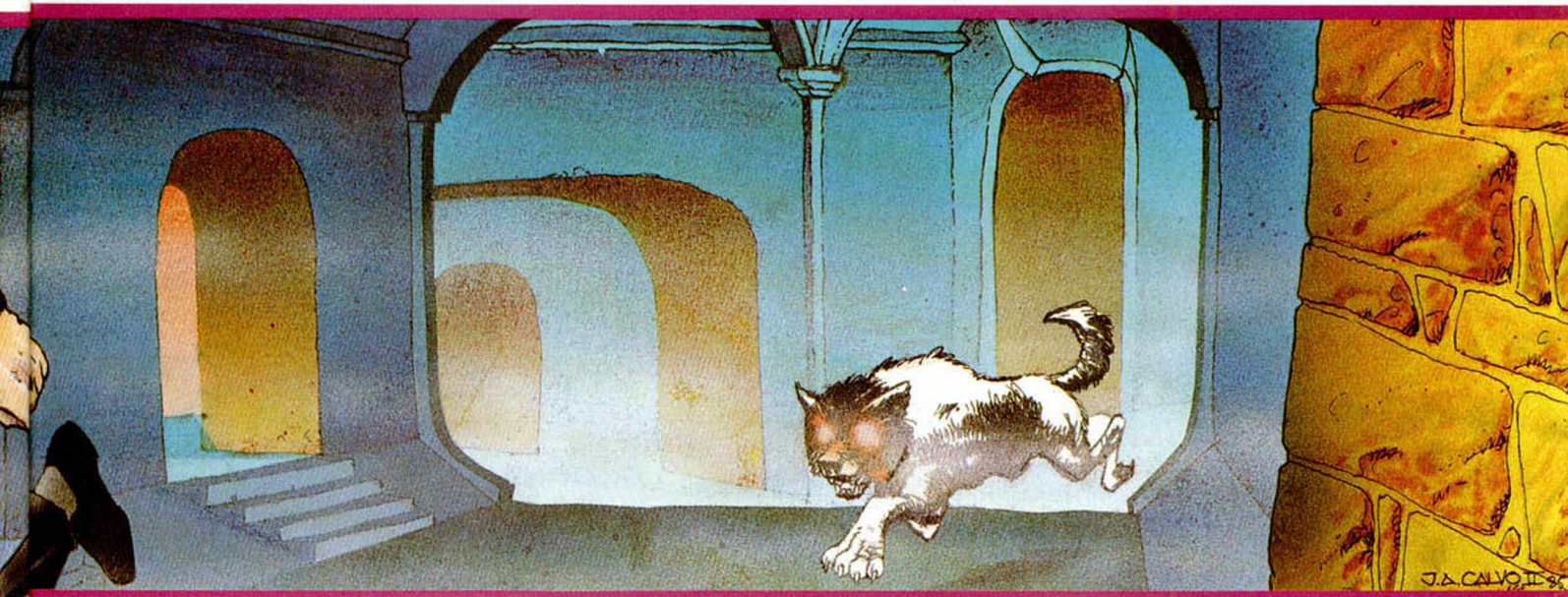
El cuarto ejemplo consta ya de dos sprites para ver cómo al cruzarse uno pasa por delante del otro. Pode-

El sexto ejemplo muestra lo llena que puede parecer la pantalla cuando se colocan los 17 sprites circulando en todas direcciones.

Por último, el séptimo no se limita a crear los sprites y dejar que se muevan, sino que, con la ayuda de la subrutina y de comprobación de choque, produce unos sonidos cada vez que el cohete es interceptado por una flecha. Es un ejemplo muy simplificado de lo fácil que es hacer un juego utilizando esta rutina para controlar los sprites.

tos de un ejemplo y los de otro, y no los lea todos de una vez.

La rutina de interrupciones, la pantalla de memoria, y algunas zonas de datos más, ocupan por completo de la dirección 53000 en adelante, así que nuestros programas y gráficos deben situarse por debajo de aquí. Puede parecer que ocupa demasiado, pero, de hecho, es más o menos lo que suelen ocupar las rutinas y áreas de datos destinados al mismo fin en programas comerciales. Hay



ción aparece a la izquierda de la pantalla y se mueve hacia la derecha.

El segundo ejemplo nos muestra una especie de reloj con una aguja siempre quieta y la otra girando en sentido horario. Como deberíais haber supuesto, se trata de animación cíclica. Además, podéis comprobar cómo un sprite no necesita estar siempre moviéndose, puede estar en una posición fija, poniendo los incrementos X e Y a 0. En este modo, podemos controlar directamente la posición del sprite, pokeando en los lugares

mos comprobar también cómo el sprite toma el color de papel que haya en la pantalla, lo que contribuye a evitar un poco la mezcla de colores que acabamos de mencionar.

El quinto ejemplo muestra, por una parte, un ejemplo práctico del uso de trayectorias preestablecidas con una flecha que gira, y por otra parte un grupo de cuatro flechas paradas de las que dos tienen animación cíclica y dos de adelante-atrás, para que se puedan apreciar bien las diferencias entre una y otra.

Para hacer vuestras primeras pruebas, podéis utilizar también el programa Basic de demostración, con sólo cambiar los DATAs de algún ejemplo. Para ayudaros, tenéis en la figura 3 una tabla con las direcciones y demás datos de los gráficos usados para la demostración, y en la figura 4 un resumen de qué valores y en qué orden debéis poner en los DATAs. Advertencia importante: al final de los datos de todos los ejemplos, es necesario añadir un cero a los DATAs para que el programa distinga entre los da-

que pensar que solamente la pantalla de memoria ocupa ya unos 7 K. Pero si todavía os parece mucho, vamos a ver con detenimiento qué es lo que hay a partir de esa dirección.

MAPA DE MEMORIA

Si sólo pensáis usar la rutina desde Basic y no tenéis ningún interés en el Código Máquina, no es necesario que leáis este apartado ni el siguiente, pues en innecesario lo que se dice en ellos

para poder utilizar los sprites. Pero si sabéis Código Máquina, continuad leyendo y podréis sacar mucho más provecho a la rutina.

En la figura 5 tenéis un esquema del mapa de memoria. Lo primero que tenemos, entre las direcciones 53000 y 53288, es la tabla de sprites, ya explicada anteriormente, y cuyo inicio será denominado TASPRI a partir de ahora.

Lo siguiente, entre 53288 y 53543, es otra tabla, TABLDI, que no hemos explicado antes porque es de uso interno de la rutina. En esta se almacenan una serie de datos de los sprites, pero solamente de los que se van a dibujar en pantalla (como hemos visto, puede haber sprites fuera de ésta). En la figura 6 hay un resumen de los 15 bytes de que consta cada elemento. Los dos primeros bytes indican la dirección en la pantalla de memoria a la que va a ir el gráfico. Los dos siguientes indican la dirección del gráfico, y los otros dos, la de la máscara. El séptimo byte indica el número de scans, o lo que es lo mismo, la altura en pixels. El octavo byte indica el ancho en caracteres. Tanto éste como el anterior no tienen por qué corresponder con los datos en la tabla de sprites, ya que éstos se refieren a las dimensiones del gráfico que se va a dibujar en pantalla, y si éste se encuentra en un lado o una esquina, la porción de gráfico que se ve es menor que el gráfico entero. El noveno byte nos dice cuántos caracteres de ancho quedan fuera de la pantalla. Comprobaremos su utilidad al estudiar el programa. El décimo byte es el llamado byte de rotación. Puede valer 248, 250, 252 ó 254, y es el byte alto de la dirección de comienzo de una tabla. Son, por tanto,

Fig. 4

Valores a colocar en los DATAS en el programa Basic

- Dirección del primer gráfico.
- Ancho en caracteres.
- Alto en pixels.
- Coordenada X.
- Coordenada Y.
- Número de fases de animación.
- Indicador: 0 = animación cíclica, 1 = adelante/atrás.
- Indicador: 0 = línea recta, 1 = trayectoria compleja.
- Color (tinta + 64 x brillo).
- Si línea recta:
 - Incremento X.
 - Incremento Y.
- Si trayectoria compleja:
 - Dirección de los datos de la trayectoria.

cuatro tablas. La primera va de 63488 a 63999, la segunda de 64000 a 64511, la tercera de 64512 a 65023, y la cuarta de 65024 a 65535. Las cuatro se crean en el momento de activar las interrupciones. Lo que hay en ellas es el resultado de girar cada uno de los 256 posibles números que puede contener un byte, 6 bits hacia la derecha, en la tabla cuarta, 4 en la tercera, 2 en la segunda y 0 en la primera. La utilidad es que a la hora de dibujar un gráfico, no tendremos que utilizar instrucciones de rotación que hacen más lento el proceso, puesto que ya tenemos el trabajo hecho. Por ejemplo, si queremos girar el número 47, 4 bits a la derecha (el número 47 puede formar parte de un gráfico, ya que sabemos que todos los gráficos, en el ordenador se reducen a números binarios, con un equivalente decimal), el proceso a efectuar sería cargar el registro H con 252, que es el número que corresponde a la tabla

tercera, después cargar I con 47, que es el número que queremos girar, y la dirección formada contendrá el resultado deseado. Como al girar un byte, hay una parte que se sale, ésta se almacena en otra dirección, justo 256 bytes más arriba, así que basta con incrementar H para obtenerla. Si no os ha quedado demasiado claro, posiblemente lo entenderéis mejor cuando veáis la subrutina que crea la tabla. Como muchos de vosotros ya habréis adivinado, el que sólo haya cuatro tablas quiere decir que el programa no trabaja realmente en alta resolución en este sentido horizontal, sino que el mínimo desplazamiento es de dos pixels. Esto no supone un gran problema, pues la suavidad obtenida es más que suficiente. Sin embargo, para mayor comodidad se permite que se den las coordenadas como si el mínimo desplazamiento fuera de dos pixels. La verdad es que este sistema del uso de tablas con los bytes desplazados no es de nueva creación, ha sido utilizado en varios programas comerciales, entre los que cabe destacar «Cyberun», cuyo rápido scroll habría sido imposible con métodos convencionales. Pero continuemos con el contenido de la tabla TADIBL. El undécimo byte indica los atributos del sprite. El decimosegundo y el decimotercero indican la dirección en la pantalla de memoria, donde van a ir los atributos del sprite. Por último, los bytes decimocuarto y decimoquinto indican el número de filas y el de columnas, respectivamente, que ocupa el gráfico en la zona de atributos.

Continuando con el mapa de memoria, entre las direcciones 53544 y 54566, está el espacio denominado

Fig. 5

Mapa de memoria

Dirección	Etiqueta	Contenido
53000	TASPRI	Tabla de datos de los sprites existentes.
53289	TABLDI	Tabla de datos de los gráficos que se van a dibujar.
53544	SPARES	Espacio reservado para guardar los trozos de pantalla.
54568(7)	ORIGSC	Pantalla de memoria.
60904	ORIGAT	Atributos de la pantalla de memoria.
61696		Tabla de interrupciones.
61953	INICIO	Subrutina de inicialización y activación.
62075	DESACT	Subrutina de desactivación.
62082	COMCHO	Subrutina de comprobación de choque.
62194	ENTINT	Subrutina principal de interrupciones.
63488		Tablas de rotaciones.

LISTADO 1

```

1 GO TO 1000
10 RANDOMIZE USR 62075: CLS :
LET NUMSPR=23681: LET N=0: LET A
=53000
20 READ D: IF D=0 THEN RETURN
30 RANDOMIZE D: POKE A,PEEK 23
670: POKE A+1,PEEK 23671
40 READ H,V: POKE A+2,H: POKE
A+3,V: RANDOMIZE H+U: POKE A+15,
PEEK 23670: POKE A+16,PEEK 23671
50 READ X,Y: POKE A+5,INT (X/2
56): POKE A+4,X-256+INT (X/256):
POKE A+7,INT (Y/256)
Y-256+INT (Y/256)
60 READ M,C: POKE A+8,M: POKE
A+12,C
70 READ R,C0: POKE A+9,C0+C+8+
R+16: IF R=1 THEN READ D2: RANDO
MIZE D2: POKE A+10,PEEK 23670: P
OKE A+11,PEEK 23671: POKE A+13,P
EEK 23670: POKE A+14,PEEK 23671:
GO TO 90
80 READ IX,IY: POKE A+10,IX: P
OKE A+11,IY
90 LET N=N+1: LET A=A+17: GO T
0 20 REM EJEMPLO 1
100 DATA 47300,2,16,0,88,1,0,0,
110 DATA 47364,3,24,120,80,8,0,
2,2,0,0
120 REM EJEMPLO 2
130 DATA 47364,3,24,120,80,8,0,
0,1,0,0,0
140 REM EJEMPLO 3
150 DATA 48580,2,16,-32,192,1,0,
0,2,2,-2,0
160 REM EJEMPLO 4
170 DATA 47300,2,16,0,88,1,0,0,
1,3,0
180 DATA 47364,3,24,256,80,8,0,
0,0,-2,0,0
190 REM EJEMPLO 5
200 DATA 48516,2,16,100,100,1,0,
1,2,47140
210 DATA 48516,2,16,0,0,8,0,0,0
0,0
220 DATA 48516,2,16,16,16,8,0,0,
0,0
230 DATA 48516,2,16,0,16,8,1,0,
0,0
240 DATA 48516,2,16,16,0,8,1,0,
0,0,0
250 REM EJEMPLO 6
260 DATA 48516,2,16,0,0,8,0,0,0
2,1
270 DATA 48516,2,16,0,16,8,1,0,
0,2,1
280 DATA 47300,2,16,0,80,1,0,0,
4,3,0
290 DATA 48516,2,16,-66,20,8,1,
0,0,1,0
300 DATA 48516,2,16,-50,36,8,1,
0,0,1,0
310 DATA 47300,2,16,0,0,1,0,0,0
6,0
320 DATA 48516,2,16,-66,36,8,0,
0,0,1,0
330 DATA 48516,2,16,-50,20,8,0,
0,0,1,0
340 DATA 47364,3,24,0,255,8,0,0,
0,0,-4
350 DATA 48580,2,16,0,200,1,0,0
,2,2,-2
360 DATA 48708,2,16,255,255,1,0
,0,3,-3,-3
370 DATA 47364,3,24,255,72,8,0,
0,5,-2,0
380 DATA 48516,2,16,90,100,1,0,
1,0,47140
390 DATA 48516,2,16,0,20,1,0,1,
2,47140
400 DATA 47364,3,24,128,88,8,1,
0,0,0,0
410 DATA 47300,2,16,-60,255,1,0
,0,1,1,-1
420 DATA 48516,2,16,90,124,1,0,
1,2,47140,0
430 REM EJEMPLO 7
440 DATA 47300,2,16,0,160,1,0,0
,3,2,0
450 DATA 48900,2,16,0,0,1,0,0,0
,0,2
460 DATA 48900,2,16,32,0,1,0,0,
0,0,5
470 DATA 48900,2,16,64,-24,1,0,
0,0,0,3
480 DATA 48900,2,16,96,16,1,0,0
,0,0,3
490 DATA 48900,2,16,128,24,1,0,
0,0,3
500 DATA 48900,2,16,160,0,1,0,0
,0,0,2
510 DATA 48900,2,16,192,-32,1,0
,0,0,2
520 DATA 48900,2,16,224,-164,1,
0,0,0,3,0
1000 PRINT TAB 3;"SPRITES POR IN
TERRUPCIONES";TAB 4;"PROGRAMA
DE DEMOSTRACION";#0;" PULSA UNA
TECLA PARA CONTINUAR"
1010 PAUSE 0
1020 RESTORE : GO SUB 10: LET X=
1: GO SUB 1120: PAUSE 256
1030 GO SUB 10: LET X=2: GO SUB
1120: PAUSE 200
1040 GO SUB 10: FOR X=0 TO 255 S
TEP 4: PLOT 0,0: DRAW X,175: PLO
T 255,175: DRAW -X,-175: NEXT X:
LET X=3: GO SUB 1120: PAUSE 256
1050 GO SUB 10: FOR X=0 TO 21: F
OR Y=3 TO 6: PRINT PAPER Y;"
":NEXT Y: NEXT X: LET X=4:
GO SUB 1120: PAUSE 290
1060 GO SUB 10: LET X=5: GO SUB
1120: PAUSE 200
1070 GO SUB 10: PRINT : FOR X=1
TO 21*32: PRINT CHR$(65+INT (RN
D*28)):NEXT X: LET X=6: GO SUB
1120: PAUSE 600
1080 GO SUB 10: LET X=7: GO SUB
1120: POKE 23729,0: FOR X=1 TO 9
0
1090 POKE 23728,255: IF USR 6208
2 THEN BEEP .01,RND*60
1100 NEXT X
1110 RANDOMIZE USR 62075: PAUSE
10: RUN
1120 PRINT AT 0,10;"EJEMPLO No.
";X: AT 0,10: OVER 1:
RANDOMIZE USR 61953: POKE N
UMSPR,N: RETURN
9999 CLEAR 47139: LOAD ""CODE 47
140,1888: LOAD ""CODE 61953,1495
: RUN

```

SPARES. Es una zona de trabajo donde se almacenarán los trozos de pantalla que van a ser borrados al dibujar los sprites, para poder restablecerlos después. La cantidad de memoria necesaria depende de los sprites que se vayan a utilizar y sus dimensiones. El valor que se le ha dado es un valor promedio que será suficiente para casi todos los casos. Si para vosotros resulta insuficiente, podéis colocar esta zona en otra parte de la

memoria, cambiando el EQU de la etiqueta SPARES en el listado ensamblador. En la dirección 54568 se encuentra la pantalla de memoria. Como sabéis, la pantalla del Spectrum tiene 256 píxels de ancho (32 caracteres) por 192 de alto, así que ocupa $32 \times 192 = 6144$ bytes (sin contar con los atributos), así que, en principio esto es lo que debería ocupar la pantalla de memoria. Sin embargo, esto no es así. Lo que pasa es

que para poder generalizar al máximo la subrutina que se encarga de dibujar los gráficos en esta pantalla, necesitamos un carácter más de ancho, en el que se dibujarán restos de los sprites que, por estar a medias en la pantalla, no deben ser dibujados enteros. De esta manera, la pantalla de memoria ocupará $33 \times 192 = 6336$ bytes. En realidad, ocupa un byte más, porque también para generalización de la subrutina del di-

bujó, se necesita que esta columna adicional sea un píxel más alta que el resto de la pantalla. Así que en realidad, la zona ocupada por la pantalla comienza en 54567, pero la dirección que se corresponde con el inicio de la pantalla real del ordenador, es la 54568, que es la que tiene el nombre de ORIGSC. A continuación de la pantalla, se encuentran sus atributos, en la dirección 60904, con la etiqueta ORIGAT. Al

igual que antes, necesitamos una columna más, con lo que ocuparán $33 \times 24 = 792$ bytes. En este caso, no necesitamos reservar especialmente otro byte antes de la dirección 60904, pues nos sirve el último de la zona de la pantalla, que será a la vez el último byte de la pantalla y el primero de los atributos.

En la dirección 61696 se encuentra la tabla de interrupciones. Se ha dicho muchas veces en esta y en otras revistas que en el modo 2 de interrupciones el microprocesador toma el contenido del registro I y el del bus de datos, que siempre es 255, y forma una dirección, de la que se extrae la dirección definitiva de la subrutina de interrupciones. Sin embargo, hay algunos periféricos que, por no estar demasiado bien hechos, hacen que el contenido del bus de datos en el momento de las interrupciones no sea 255. Esto quiere decir que la dirección donde el microprocesador buscará la dirección de las interrupciones, puede ser cualquiera cuyo byte alto sea el contenido del registro I. Para conseguir que sea cual sea el valor del bus de datos, se lea la misma dirección para la subrutina de interrupciones, no hay más remedio que elegir para la subrutina una dirección cuyos byte alto y bajo sean iguales, y llenar con el valor de estos, 257 bytes a partir de la dirección XX00, siendo XX el contenido del registro I. En nuestro caso, para que la tabla de interrupciones esté en 61696, I valdrá $61696 / 256 = 241$, y la subrutina de interrupciones comenzará en $62194 = F2F2h$, así que la tabla estará llena de $242 = F2h$. Es fácil ver que entonces, sea cual sea el contenido del bus de datos, la dirección formada con el

registro I estará entre 61696 y 61951, y que el contenido de cualquiera de estas direcciones y de las siguientes será 242, con lo que siempre se saltará a la dirección $242 * 256 + 242 = 62194$.

Tras la tabla de interrupciones tenemos, en 61953, INICIO, la subrutina que inicializa las interrupciones y crea la pantalla de memoria y las tablas de rotaciones.

En 62075 está DESACT la subrutina que desactiva las interrupciones y vuelve al modo normal.

En 62082 está la subrutina COMCHO, de comprobación de choque de sprites. En realidad no cabe entera aquí, pues se montaría sobre la dirección donde deben comenzar las interrupciones, por lo que parte de ella se encuentra al final de todo el bloque de la rutina de interrupciones y sus subrutinas.

En 62194 se encuentra la subrutina principal de interrupciones, junto con todas sus subrutinas y variables (excepto NUMSPR y SPRICH, que se encuentran en variables del sistema no usadas). Detrás, como ya hemos dicho, continúa la subrutina de comprobación de choque, que aunque no es muy larga, ha sido cortada en dos para aprovechar mejor la memoria, rellenando los huecos.

Y lo último que hay en la memoria son las ya mencionadas tablas de rotaciones, que comienzan en 63488. A propósito de estas tablas, seguramente os habréis preguntado qué sentido tiene almacenar un byte girado 0 veces (que es lo que hay en la primera tabla), puesto que el resultado será el mismo. La respuesta es bien sencilla. Si no incluyéramos esa tabla, sería necesario hacer dos subrutinas de dibujo, una para cuando

el gráfico está en el primer píxel de un carácter y otra para el resto de los casos. El resultado quedaría menos elegante, sería más difícil de comprender y ocuparía prácticamente la misma memoria.

FUNCIONAMIENTO

El proceso que hay que realizar para mover los sprites por la pantalla es bastante complejo y lleva bastante tiempo. Por ello, lo vamos a dividir en dos partes, de tal modo que una vez haremos una parte, y la siguiente vez haremos la otra parte, y a la siguiente volveremos a hacer la primera, y así. Hacemos el volcado en pantalla de la pantalla de memoria, que es el proceso más lento, y la segunda, haremos el resto, que se puede dividir en las siguientes subetapas:

- Borrar los sprites de sus posiciones antiguas.

- Calcular nueva posición de los sprites y el gráfico que toque de los que compongan la animación de cada uno.

- Hacer todos los cálculos necesarios para el dibujo de los gráficos y su posterior borrado, almacenando los resultados en la tabla TABLDI, a la vez que se almacenan los trozos de pantalla que van a ser ocupados por los sprites.

- Dibujar los sprites que se encuentran en la pantalla.

Ahora que sabemos todas las tareas a realizar, podemos comenzar a comentar el listado de la rutina.

El punto de entrada de ENTIT. Aquí, lo primero que

hacemos es guardar todos los registros que vamos a utilizar. A continuación, estudiamos el contenido de la variable ESTADO. Si es 0, deberemos volcar la pantalla, y si es 1, hacer todo lo demás. En este segundo caso saltamos a NOVOLC. En el primero, llamamos a la subrutina que vuelca la pantalla y hacemos que ESTADO valga 1 para la próxima vez que se produzca una interrupción, para salir después por SALINT.

En el caso de encontrarnos en la segunda parte, ponemos ESTADO a 0 para la próxima vez. Después nos disponemos a borrar todos los sprites dibujados la última vez. Para ello, tomamos el contenido de NUBLBO, que excede en uno al número buscado. Antes de entrar en el bucle que borrará todos los gráficos, inicializamos IX con TABLDI, tabla que tiene los datos de los sprites que fueron dibujados y ahora queremos borrar, y HL con SPARES, donde están almacenados uno detrás de otro los trozos de pantalla necesarios para efectuar el borrado. Ahora, si el contenido de NUBLBO era 1, es que no habíamos dibujado ningún gráfico la última vez, y salimos del bucle sin haber entrado en él.



Ahora vamos, tomando de la tabla los datos necesarios. En DE cargamos la dirección en la pantalla de trabajo donde comenzará el proceso. Ahora, en vez de cargar en registros, hay unos datos que los vamos a meter en una dirección de memoria, de tal manera que luego serán recogidos directamente por una instrucción del tipo LD C,n,

siendo n el dato en cuestión. Este sistema les resultará familiar a los que hayan leído los artículos sobre cómo se programa un juego, publicados en los MICROHOBBY 97-106, y escritos por el mismo autor. En BORPOI+1 y en BORPOD+1 guardamos la cantidad de bytes de ancho que hemos de restaurar. En AUMPOI+1 y en AUMPOD+1, guardamos

33 menos ese número, que será lo que hay que sumar a la dirección destino de la pantalla, tras restaurar un scan para lograr la dirección del siguiente. Por último, cargamos A con el número de scans que debemos restaurar. En BORPOI, aunque veamos un LD C,0 estamos cargando C con el ancho del bloque a restaurar, pues es el valor que había-

mos metido en BORPOI+1. Como en B teníamos 0, en BC tenemos el número de bytes a renovar; en DE tenemos el destino, y el origen en HL, porque los trozos de pantalla que vamos a recuperar han sido anteriormente guardados en SPARES, así que tenemos los parámetros necesarios para hacer un LDIR. Ahora calculamos la dirección en panta-



lla del siguiente scan, sumándole al contenido de DE tras el `l dir`, el número calculado antes y medido en `AUMPOI+1` (por si no lo habéis notado, os diré que en la pantalla de trabajo, los scans están uno tras otro, a diferencia de cómo están en la pantalla del ordenador). Tras esto, cerramos el bucle para terminar con todos los scans. Ahora, nos queda restaurar los atributos. Para ello, cargamos en DE la dirección destino de atributos, y en A el número de filas que ocupa. Por lo demás, la restauración de los atributos es idéntica a la que acabamos de hacer. Tras restaurar por completo un trozo de pantalla, cerramos el bucle que los restaurará todos. Ya hemos terminado la primera de las subfases antes citadas.

Ahora continuamos por `INSABO`. Aquí hacemos un bucle en el que iremos moviendo todos los sprites y calculando los datos para la tabla `TABLDI` en los que estén en la pantalla. Dentro de este bucle, `IY` será el puntero para el elemento correspondiente al sprite dentro de la tabla de sprites `TASPRI`, `IX` el de `TABLDI`, y `DE` señalará la primera dirección libre de la zona `SPARES`, pues en este bucle también será donde guardaremos los trozos de pantalla que hagan falta. Dentro del bucle, lo que encontramos son llamadas a dos subrutinas. `MOVERR` se encargará de calcular las nuevas coordenadas del sprite, teniendo en cuenta si se mueve en línea recta o según trayectoria prefijada, y de calcular la fase de animación en que se encuentra, si la tiene. `CREADA` crea el elemento en la tabla `TABLDI` y guarda los trozos de pantalla.

Una vez guardados todos

los trozos de pantalla, ya podemos dibujar los sprites en sus nuevas posiciones, llamando a `DIBUJA`, que se apoyará en los datos de `TABLDI`. Ahora, antes de volver, hacemos un `rst 56` para leer el teclado, y después entramos en `SALINT`, donde se recuperan los registros y se vuelve. Nótese que tal y como está puesto, y teniendo en cuenta que la fase de volcado de pantalla abarca el tiempo de dos interrupciones, la lectura del teclado se hará tres veces menos de lo normal. Esto hace que la respuesta del mismo al teclear sea bastante pobre, pero da un poco más de velocidad al no tener que ejecutarse tantas veces la subrutina de lectura de teclado. Si pensamos usar los sprites desde Código Máquina, podría ser útil quitar las líneas 96, 97 y 98 para ahorrar aún más tiempo.

Y esto es todo el programa principal. Ahora quedan las subrutinas, que son cuatro: `VUELCA`, `MOVERR`, `CREADA` y `DIBUJA`.

`VUELCA` es la que se encarga de copiar la pantalla de trabajo en la pantalla del ordenador. Se trata del proceso que más tiempo consume, ya que hay que mover casi 7 K. La rutina vuelca todos los scans de arriba a abajo, y por cada 8, vuelca una fila de atributos. Puede pareceros que sería más sencillo volcar primero toda la pantalla sin atributos, y luego los atributos, pero el resultado sería que el movimiento quedaría mucho menos suave. Con este procedimiento habrá pequeñas zonas de la pantalla donde sprites parecerán doblarse; pero en la inmensa mayoría de los casos, la suavidad de movimientos será perfecta.

Veamos cómo funciona la subrutina. En los registros

alternativos guardamos en `HL` el origen de pantalla de trabajo, y en `DE` el de la pantalla del ordenador, además de poner `C a 0`. En los registros normales ponemos en `HL` el origen de los atributos de memoria, y en `DE` su correspondiente en la pantalla del ordenador. Asimismo, cargamos `C` con 0 y `B` con 25 (estas dos últimas operaciones de una sola vez). `B` será el contador del bucle que se repetirá para cada fila, dentro del cual deberemos volcar ocho scans y una fila de atributos. El bucle se repetirá 22 ve-

ces, a pesar de que inicialicemos `B` con 25. La razón es que, como veremos enseguida, a cada paso por el bucle, el registro `BC` será decrementado 32 veces, así que al final de las 22 veces que se repetirá el bucle, `BC` habrá sido decrementado en $22 \times 32 = 704$, con lo que `B` habrá sido decrementado en 3. Por tanto, deberemos empezar con `B` valiendo 3 más de las veces que se repita el bucle. Dentro del bucle de las 22 filas, volvemos a los registros alternativos para volcar los scans. Cargamos `B` con nueve para entrar en un bucle que se repetirá ocho veces, adivinad porqué. Dentro de este segundo bucle, trasladamos los 32 bytes que componen un scan de la pantalla desde la idem de memoria a la real. Posiblemente os sorprenda que para esto utilicemos 32 instrucciones `LDI`. Pero resulta que esto es mucho más rápido que cargar `BC` con 32 y hacer un `LDIR`, y además, así podemos utilizar el registro `B` como contador. Estos `DLLs` y los que veremos a continuación son la causa de que en un bucle tengamos que cargar `B` con 25 y en el otro 9 en vez de 22 y 8 respectivamente. Tras esto, calculamos en `DE` la dirección del siguiente scan. Dentro de este bucle siempre vamos a pasar de un scan a otro de la misma fila, y, como muchos ya sabréis, el algoritmo para pasar de un scan a otro de la misma fila es simplemente incrementar el byte alto de la dirección. Pero antes debemos restarle 32, porque al hacer los `LDIs` es como si le hubiéramos sumado 32. Como no hay registros dobles libres, nos valemos de `A` para hacer la resta. Primero le restamos 32 a `E`. Si no se nos produce acarreo, la resta ya está bien hecha y procedemos a

Fig. 6

TABLA DE DATOS DE LOS GRÁFICOS QUE SE VAN A DIBUJAR

Dirección relativa	Contenido
<code>DIR+0</code>	Dirección de destino en pantalla de memoria.
<code>DIR+2</code>	Dirección del gráfico.
<code>DIR+4</code>	Dirección de la máscara.
<code>DIR+6</code>	Número de scans.
<code>DIR+7</code>	Bytes de ancho.
<code>DIR+8</code>	Bytes que no entran en pantalla.
<code>DIR+9</code>	Byte de rotación.
<code>DIR+10</code>	Byte de atributos.
<code>DIR+11</code>	Dirección de destino en atributos de memoria.
<code>DIR+13</code>	Número de filas de atributos que ocupan el gráfico.
<code>DIR+14</code>	Número de columnas de atributos.

incrementar D. Pero si hay acarreo, deberemos restarle uno a D, pero como después tendríamos que incrementarlo de nuevo, no hacemos ninguna de las cosas. Antes de cerrar el bucle, HL necesita ser incrementado para saltarnos la columna de más que ya dijimos que había en la pantalla de trabajo. Tras salir del bucle, en el que habremos volcado los ocho scans, hacemos los ajustes necesarios para pasar al primer

scan de la siguiente fila, teniendo en cuenta que es posible que estemos pasando de un tercio a otro. No creo que estos ajustes merezcan más explicación, pues ya se han gastado muchos litros de tinta hablando de cómo se calcula el scan siguiente a uno dado en la pantalla del Spectrum. Ahora volvemos a los otros registros, y trasladamos una fila de atributos. Tras incrementar HL para saltarnos la columna

sobrante, cerramos el bucle.

Llegamos ahora a MOVERR, la subrutina que calcula las nuevas coordenadas de cada uno de los sprites. A esta subrutina se llega con IY apuntando a los datos del sprite en tratamiento dentro de la tabla de sprites TASPRI. IX, por su parte, apuntará al lugar donde debemos crear el elemento correspondiente

de la tabla TABLDI. Lo primero que hacemos al entrar es comprobar que el sprite esté activado, para lo cual, como ya hemos dicho, comprobamos que no sea 127 el sexto byte de la tabla (señalado por IY+5). Si es 127, regresamos inmediatamente. A continuación, guardamos en D el décimo byte de la tabla, que era el que indicaba el color, el tipo de

LISTADO 2

LÍNEA DATOS CONTROL

Table with columns for line number, data, and control values. The data column contains hex strings representing sprite data, and the control column contains decimal values. The table is rotated and partially obscured by a yellow background.

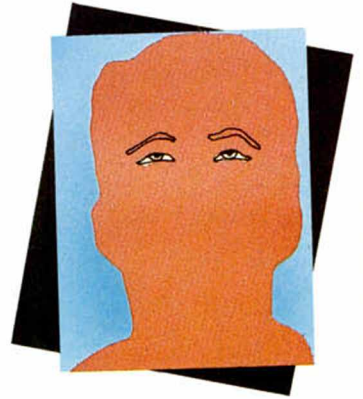
DUMP: 40.000
N.º BYTES: 1.888

animación y el tipo de movimiento. Aislamos el color y lo guardamos en su lugar correspondiente en la tabla TABLDI. Ahora, si el movimiento es en línea recta, saltamos a RECLIN. Si es en trayectoria preestablecida, cargamos en HL la dirección de los datos de la trayectoria. Leemos el valor contenido en esa dirección. Si es 127, la trayectoria ha finalizado y cargamos en HL la dirección inicial para volverla a repetir. Tanto si hacemos esto como si no, continuamos por NOFITA. Ahora comprobamos si el valor leído es un 126. En este caso, los dos siguientes bytes indicarán la nueva dirección inicial de los gráficos correspondientes al sprite, y los pasamos a su lugar en la tabla TASPRI, para retroceder a continuación a RECUPERE y leer un nuevo dato de la tabla de datos de la trayectoria. Cuando nos encontramos con un dato que no es 127 ni 126, llegamos a TRANOR. Este dato será el incremento X, y lo pasamos a E, cargando A el siguiente valor que será el incremento Y, tras actualizar el contador de trayectoria, saltamos a COCORE, donde nos reuniremos con la bifurcación hecha en el caso de un movimiento en línea recta. En este caso saltábamos a RECLIN. Aquí, cargamos en E el incremento X y en A el Y, para entrar en COCORE con los mismo datos que si hubiéramos venido desde el caso de trayectoria prefijada. En COCORE, pasamos el incremento Y a BC. Hay que tener en cuenta que los incrementos son números con signo, y al pasarlos a un registro doble, el registro bajo deberá ser igual que el byte de incremento, pero el alto, deberá ser 0 para un incremento positivo y 255 para un incremento negativo. Es-

to lo conseguimos con el CP 128 y el CCF, que pone el banderín de acarreo alzado para un número negativo, y el SBC A,A, que dejará en A un 0 si el acarreo estaba bajo, y un -1 (un 255) si estaba alto. Este incremento, una vez convertido en un número de 16 bits, se lo sumamos a la coordenada Y, y guardamos la nueva coordenada. A continuación, hacemos lo propio con la X. Ya hemos actualizado las coordenadas. Ahora vamos a pasar a calcular qué gráfico hay que dibujar, de los varios que puede tener un sprite, para lo cual tendremos que tener en cuenta la fase de animación. Por eso cargamos en E el número total de fases de animación y en A la fase en la que estaba la última vez. Ahora saltamos a ADETRA si la animación es del tipo adelante-atrás. Si es cíclica, simplemente incrementamos la fase, y si hemos llegado a la última, la ponemos a 0. Después saltamos a TRAREN. Nótese que en la animación cíclica, coinciden la fase actual con el gráfico que hay que dibujar. Esto quiere decir que si la fase es 0 habrá que dibujar el primer gráfico, si es 1, el segundo, etc. En ADETRA tratamos la animación de adelante-atrás. En este tipo de animación, si por ejemplo, tenemos cinco gráficos

distintos para la animación, primero habrá cinco fases que coincidirán con las de la cíclica. Pero después, habrá tres más en las que se dibujarán los tres gráficos del centro en orden inverso: primero el cuarto, luego el tercero, y después el segundo. En general, si hay N gráficos distintos, la cantidad de fases por las que pasaremos será de $2*N-2$. Por eso ADETRA, tras incrementar el contador de fase, le restamos ese número. Si el resultado es 0, el ciclo habrá terminado y comenzaremos de nuevo por la fase 0. Pero antes hemos guardado el resultado de la resta. Ahora tenemos que calcular qué es el que hay que dibujar. Si la fase actual es menor que la cantidad de gráficos distintos que hay, nos encontramos en la parte que es igual que en la animación cíclica, por lo que la fase es igual al número del gráfico. En caso contrario, bastará que calculemos $2*N-2$ -fase actual, pero eso es lo que habíamos calculado antes, salvo que con signo contrario así que en PARETR lo recuperamos (había sido guardado en PARETR+1), lo cambiamos de signo y continuamos por TRAREN como en los otros casos. En TRAREN, cargamos DE con la cantidad de memoria que ocupa cada gráfico, y la multiplicamos por dos caras para calcular la que ocupa cada gráfico con su máscara. Este es el valor que le tendremos que sumar a la dirección del primer gráfico del sprite tantas veces como indique el contenido de A. Tras hacer esto, guardamos la nueva dirección del gráfico en el lugar que le corresponde en la tabla TABLDI. Y esto es todo lo que hace MOVERR.

Llegamos ahora a CREADA, posiblemente, la subrutina más complicada. Básicamente,



mente, lo que se hace en ella es averiguar si hay que dibujar todo el gráfico, o éste se encuentra parcialmente en la pantalla, y calcular todos los datos para que luego DIBUJA se encargue ya de dibujar los gráficos. Lo primero que hacemos es calcular qué tabla de rotaciones de las cuatro existentes es la que vamos a utilizar. En realidad, lo que calculamos es el byte alto de su dirección de inicio (el bajo es 0). Tras guardar esto en su lugar, comprobamos si la coordenada X es positiva y menor de 256. En este caso saltamos a XPOSIT. En caso contrario, si no es mayor de -256, retornamos inmediatamente, pues el sprite se encuentra fuera de la pantalla. Cuando es negativa mayor que -256, comprobamos si es mayor o igual que -6, en cuyo caso, el trozo del gráfico que no se ve en pantalla será menor de un carácter. Esto quiere decir que habrá que dibujar el gráfico entero, pero empezando en la columna sobrante de la pantalla de memoria, para que quede aquí el trozo que no se verá en pantalla. Por eso cargamos en BC ORIGSC-1 y en HL ORIGAT-1. Es en estos registros donde vamos a construir las direcciones destino del gráfico en la pantalla de trabajo y en los atributos de la misma. Como hemos dicho que dibujaremos el gráfico entero, hace-



mos que el ancho del gráfico en la pantalla sea igual al ancho del gráfico en sí, y que el número de bytes de ancho que no entran en la pantalla es 0. Tras esto saltamos a PARTEY para hacer cálculos similares con la coordenada Y. Continuamos en NEGARE, a donde llegamos cuando la coordenada X es mayor que -256 pero menor que -6. Como las dimensiones de los gráficos en la magnitud X viene dada en caracteres, vamos a pasar esta coordenada X de alta a baja resolución, para lo cual la dividimos por 8, teniendo en cuenta que se trata de un número negativo, y por lo tanto, los bits que entren

es porque el sprite está completamente fuera de la pantalla, y retornamos enseguida. Si el resultado excede de cero, coincidirá con la cantidad de caracteres de ancho que del gráfico cabrán en la pantalla, y lo guardamos en su lugar en TABLDI. Ahora si que hay un trozo del gráfico que no se dibuja por estar fuera de la pantalla. El ancho de este trozo será igual a la coordenada en baja resolución del sprite, sólo que cambia de signo. Pero para esos bytes que no se van a dibujar sean tomados de la derecha, como debería ser, y no de la izquierda, este número no sólo lo guardaremos en IX+8, sino que se lo sumaremos a la dirección de comienzo del gráfico. Por último, cargamos en BC y HL, ORIGSC-1 y ORIGAT-1 respectivamen-



por la izquierda deben ser unos y no ceros. A esta coordenada en baja resolución, le sumamos el ancho del gráfico. Si el resultado es cero o ni siquiera llega,

para que, como antes, el trozo del primer byte a dibujar que no debe salir en pantalla, quede en la columna extra. Pero si el gráfico empieza justo en la posición de un carácter, no habrá ningún byte que quede a medias entre dos direcciones, o lo que es lo mismo, el primer byte a dibujar del gráfico quedará completamente dentro de la pantalla, por lo que hacemos que BC y HL incrementen en uno sus valores antes de saltar a

PARTEY. Seguimos ahora por XPOSIT. Aquí la coordenada X estará entre 0 y 255, y pueden ocurrir dos casos: que el sprite quede totalmente dentro o que se salga parte por la derecha. Vamos a comprobarlo. Si la X es 0, consideramos desde el principio que el sprite está totalmente dentro de la pantalla. En caso contrario, calculamos la columna en baja resolución donde terminaría, el dibujo. Si no llega a 33, el gráfico cabe entero. En caso contrario, al valor obtenido le restamos 32 y ya tenemos cuantos bytes de ancho no tienen que ser dibujados. Lo restamos del ancho del sprite y obtenemos cuántos han de serlo. En RECOIN calculamos la dirección destino en la pantalla de trabajo y saltamos a PARTEY. En INTEGE, lo único que hacemos es indicar que el ancho del gráfico que cabe en pantalla es igual al ancho total de éste y que el ancho de la parte que no cabe es 0, y después retrocedemos a RECOIN para calcular la dirección de la pantalla de trabajo. Y llegamos a PARTEY. Aquí se hace lo mismo que acabamos de hacer con la X, así que no la vamos a comentar tan en detalle, sino sólo a ver las diferencias. Aquí, cuando un sprite no está completamente en la pantalla, no necesitamos calcular nada más que cuantos scans quedan dentro, y no cuántos fuera, aunque cuando se sale por arriba, igual que cuando con la X se salía por la izquierda, habrá que modificar la dirección de comienzo del gráfico, sumándole el ancho del gráfico tantas veces como scans quedan fuera de la pantalla. Lo único que se hace y que no se hacía con la X es calcular cuántas filas de atributos ocupan el gráfico. En el caso

de un gráfico que se sale por arriba o por abajo, cogemos el número de scans que caben en pantalla, lo dividimos por 8, y si no resultado exacto, al cociente le sumamos 1 y nos olvidamos del resto, y ese cociente será lo que buscábamos. En el caso de un sprite que cabe completamente en la pantalla, se calcula la fila de la pantalla dentro de la que está el primer scan del gráfico, y la fila dentro de la que está el último, se restan, se le suma 1 y obtenemos el resultado deseado. Continuaremos ahora la explicación más detalladamente desde COYPOS. En primer lugar, guardamos en su lugar correspondiente en la tabla la dirección de la pantalla de memoria a la que va a ir el gráfico. A continuación calculamos la dirección de la máscara a partir de la dirección del gráfico y de la memoria ocupada por un gráfico. Ahora calculamos cuántas columnas de atributos ocupa el gráfico, si está en el principio de una columna, ocupará tantas columnas como su ancho en caracteres, pero en caso contrario, ocupará una más. Ahora recuperamos de la pila a DE y HL (que han sido guardados en el fragmento de listado que no hemos visto con detenimiento), y volvemos a guardar HL. Tras meter la dirección de atributos en su sitio y aumentar en 1 el contenido de la variable NUBLBO para indicar que hay un sprite más para dibujar, llegamos a las líneas que se encargan de guardar en SPARES (recuérdese que DE apunta a SPARES) cada uno de los trozos de pantalla donde luego van a ir los sprites. Tampoco necesitan comentarios, pues son prácticamente idénticas a las que tomaban estos trozos de SPARES y los copiaban en la

pantalla. Al final, actualizamos IX para que apunte al siguiente elemento de la tabla.

Y llegamos a DIBUJA, la subrutina que, utilizando el gráfico y la máscara y todos los datos de la tabla TABL-DI, se encarga de efectuar los dibujos de los sprites en la pantalla de memoria. El mayor problema de comprensión que puede presentar esta subrutina es que maneja demasiados datos y puede uno perderse. Como

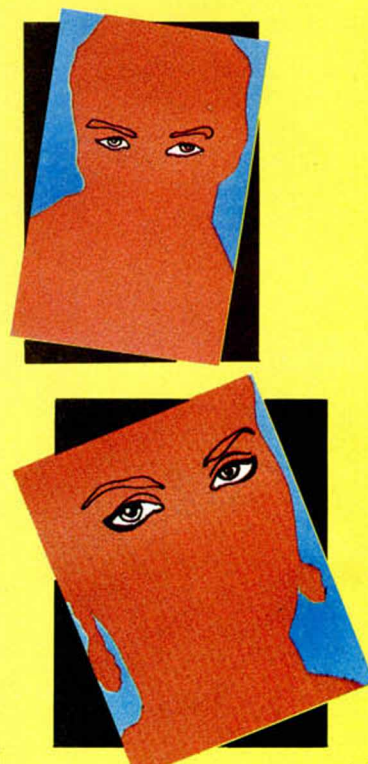
contador para el bucle, que se repetirá tantas veces como sprites haya que dibujar, utilizamos A guardado en la pila. Veamos todos los datos que cargamos dentro de este bucle antes de proceder a efectuar el dibujo. En DE cargamos la dirección de la pantalla de trabajo. En H, el byte alto de la tabla de rotación. En PONUDO+1, el valor 255 girado N veces (siendo N el número de pixels que debe-

mos desplazar todo el gráfico antes de dibujarlo). En POSNUM+1, almacenamos un 255 girado (8-N) veces hacia la izquierda. Ahora intercambiamos los registros con los alternativos. En HL, cargamos la dirección del gráfico para luego pasarla a IY. En HL cargamos la dirección de la máscara. En POSBYT+1, cargamos el ancho en ca-

racteres. En SUMVAL+1, cargamos 33 menos el número de scans de ancho. Este es el valor que hay que sumarle a la dirección de pantalla tras haber dibujado un scan para obtener la dirección del siguiente. C será el contador de los scans que debemos dibujar. Por

LISTADO 3

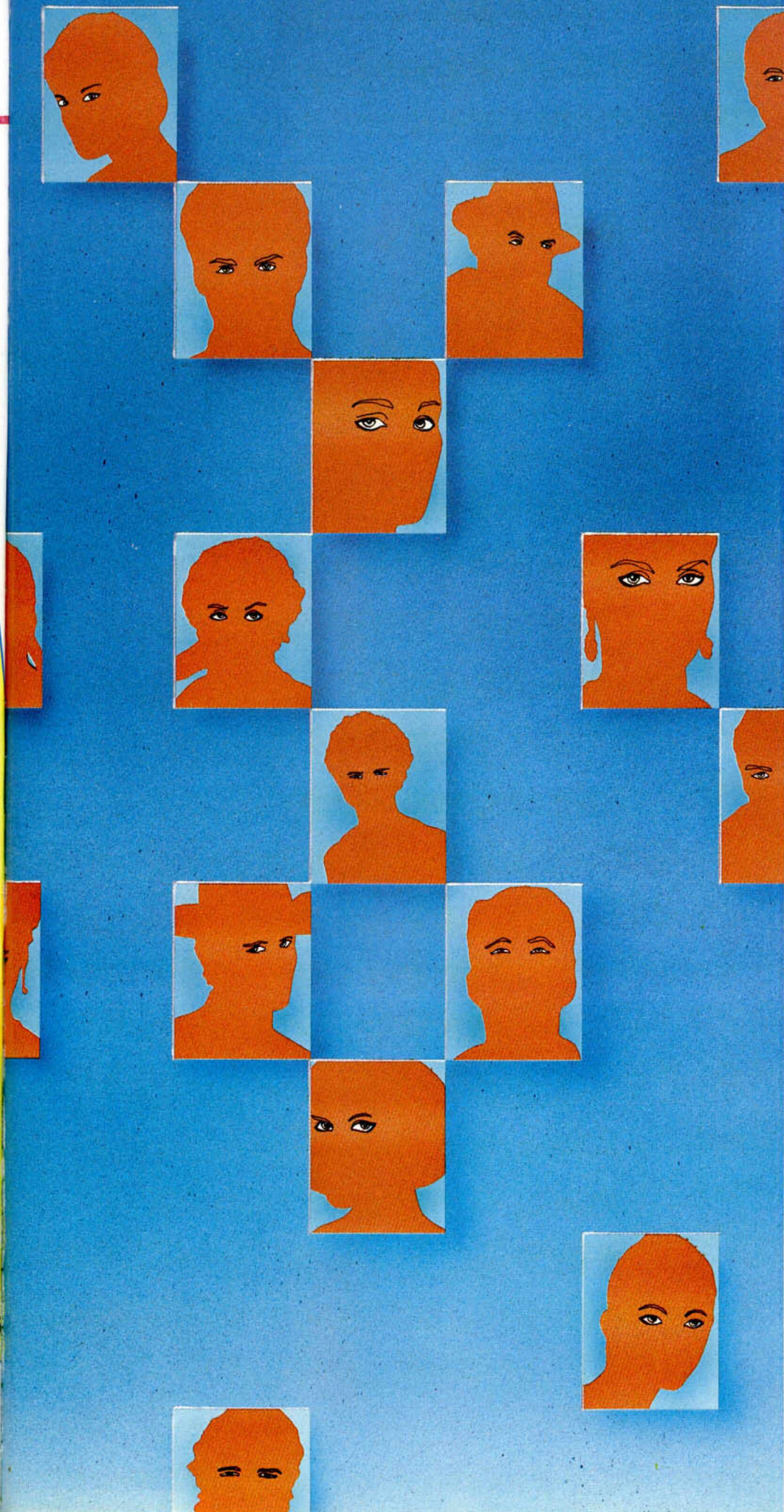
LÍNEA	DATOS	CONTROL
1	F33EF1ED4706002100F1	1134
2	36F22310FB35F21128D5	1164
3	2100403EC008E5012000	621
4	EDB013E1247CE607200A	1096
5	7DC6206F38047CD60867	975
6	083D20E321005811E8ED	935
7	3E18012000EDB0133D20	644
8	F7ED5E0E0426F83E0991	1098
9	912E00451E0057152806	444
10	CB38CB1B18F770247325	1060
11	2D20EC24240D20E13E01	718
12	329DF7AF32815C329CF7	1353
13	FBC93E3FED55ED47C9ED	1646
14	4B805C0C280A0DC83F2	1044
15	010000D003C9DD21B05C	935
16	3A815CDD340008ED48B0	1048
17	5C78B92807CDB3F20101	1072
18	00D8063D20E908CDE21B9	980
19	F722B7F7219EAF722EAF2	1659
20	CDFFF2D021C0F722CB3F7	1798
21	21AAF722EAF7A7ED527830	1711
22	EB79CDB3F7A7ED527830	1641
23	087C2F677D2F6F2379CD	926
24	C7F7C300000000000000	641
25	00F3F5C5D5E5D5E5D5E5	1802
26	F5C5D5E5FDE53A9CF7A7	1994
27	C213F3CDAFF33E01329C	1348
28	F7C39EF32128D1AF329C	1506
29	4A08DD5E00DD5601DD7E	1313
30	0E3244F3325EAF3DD7E06	1052
31	213248F3325EAF3DD7E06	1261
32	06000E00EDB00E0B09	1138
33	EB3DC243F3DD7E00DD5E	691
34	0BD0560C0E0EDB00E00	1475
35	EB09EB3DC259F30E0FDD	771
36	0908C321F33E01329DF7	1316
37	1128D1DD2129D0F2D108	1063
38	CF3A815C47A72814C5D5	1194
39	CD5FF4D1CD1BF5011100	1248
40	FD09C110EFCDE5F6F0E1	1868
41	FDE5FFFDE1E1D1C1F108	2091
42	D9DDE1E1D1C1F1FED4D	631
43	2128D51100400E00921	823
44	E8ED110058010019D906	1834
45	09EDA0EDA0EDA0EDA0ED	1985
46	A0EDA0EDA0EDA0EDA0ED	1985
47	A0EDA0EDA0EDA0EDA0ED	1985
48	A0EDA0EDA0EDA0EDA0ED	1985
49	A0EDA0EDA0EDA0EDA0ED	1985
50	A0EDA0EDA0EDA0EDA0ED	1474
51	A0EDA0EDA07BD6205F38	758
52	01142310B67BC6205F38	1446
53	047AD60857D9EDA0EDA0	1985
54	EDA0EDA0EDA0EDA0EDA0	1985
55	EDA0EDA0EDA0EDA0EDA0	1985
56	EDA0EDA0EDA0EDA0EDA0	1985
57	EDA0EDA0EDA0EDA0EDA0	1985
58	EDA0EDA0EDA0EDA0EDA0	1985
59	EDA0EDA0EDA0EDA0EDA0	1509
60	2305C2C1F3C9FD7E05FE	1443
61	7FC8FD7E0957F6647DD77	1131
62	0ACB622831FD6E0DFD66	1186
63	0E7EFE7F2007FD6E0AFD	1079
64	660B7EFE7E200E237EFD	
65		



DUMP: 40.000
N.º BYTES: 1.49

último, en DE cargamos el número de caracteres de ancho que no vamos a dibujar del sprite. Sobre LOS-CAN se cerrará el bucle para cada scan.

Aquí intercambiamos los registros. Ahora tenemos que tener bien claro cuál es el proceso a seguir para dibujar cada scan. Cada byte del gráfico tendrá que ser colocado entre dos direcciones de memoria dentro de la pantalla. Por tanto, en una dirección de la pantalla entrarán fragmentos de dos bytes del gráfico. Sabiendo que la tabla de rotaciones nos da por un lado, la parte del byte del gráfico que va en la dirección de pantalla actual y por otro lado la parte irá en la misma dirección que la primera parte del siguiente byte de gráfico, el proceso para cada byte sería: obtener la primera parte del byte. Junlarla con la segunda parte del anterior byte. Almacenar el resultado en pantalla. Obtener la segunda parte del byte y guardarla para su uso con el byte siguiente. En realidad, sería un poco más complicado, pues hay que hacer esto para el byte del gráfico y para el de la máscara. El paso descrito como guardar el resultado en pantalla, será en realidad hacer un AND de lo contenido en pantalla con el resultado de la máscara, a continuación hacer un OR con el resultado del gráfico y por último almacenar el resultado en pantalla. A partir de ahora, para intentar clarificar un poco las cosas, vamos a llamar B1, C1, etc., a los registros del juego que hemos inicializado primero, es decir, en el que DE tiene la dirección de la pantalla y HL la de la tabla de rotaciones, y B2, C2, etc., a los del otro juego, es decir, en el que HL contine la dirección de la máscara



Actualidad, pokes, mapas, trucos,
los mejores juegos y programas para
SPECTRUM, AMSTRAD, COMMODORE y MSX



Todo el universo
del Software
mes a mes

MICROMANÍA ya está a la venta
¡Pídela en tu Kiosco!

y DE el número de bytes que no hay que dibujar. Para guardar la segunda parte del byte del gráfico, utilizaremos B1 para el gráfico y C1 para la máscara. Pero en el caso del primer byte de un scan, estos registros no podrán contener la segunda parte del byte anterior, porque éste no existe. Por esto, lo que hacemos es invertarnos un byte anterior, que para que no afecte al dibujo debe ser 0 para el gráfico y 255 (una vez girado y tomada su segunda parte) para la máscara.

Éstos son los valores que debemos cargar en B1 y C1 antes de entrar en el bucle para todos los bytes de un scan, y lo hacemos en POSNUM (recuérdese que previamente habíamos cargado en POSNUM+1 la segunda parte del resultado de rotar 255). Ahora volvemos al juego de registros 2, e inicializamos B2 que va a ser el contador del bucle para los bytes de un scan (el número de scan lo habíamos puesto en POSBYT+1). Sobre LOBYTS se cerrará el bucle. Aquí, guardamos en A' el byte del gráfico y en A el de máscara. Volvemos al juego 1. Cargamos en A la primera parte del byte de máscara y la juntamos con la segunda del byte anterior que estaba en C, y después cargamos en C la segunda parte del byte actual. Ahora pasamos a A el byte del gráfico y después a L para calcular la dirección correspondiente en la tabla de rotación. Como hemos incrementado H, HL estará apuntando a la segunda parte del byte, y no a la primera.

Cargamos en A la segunda

parte del byte anterior y en B la de éste para usarlo la próxima vez.

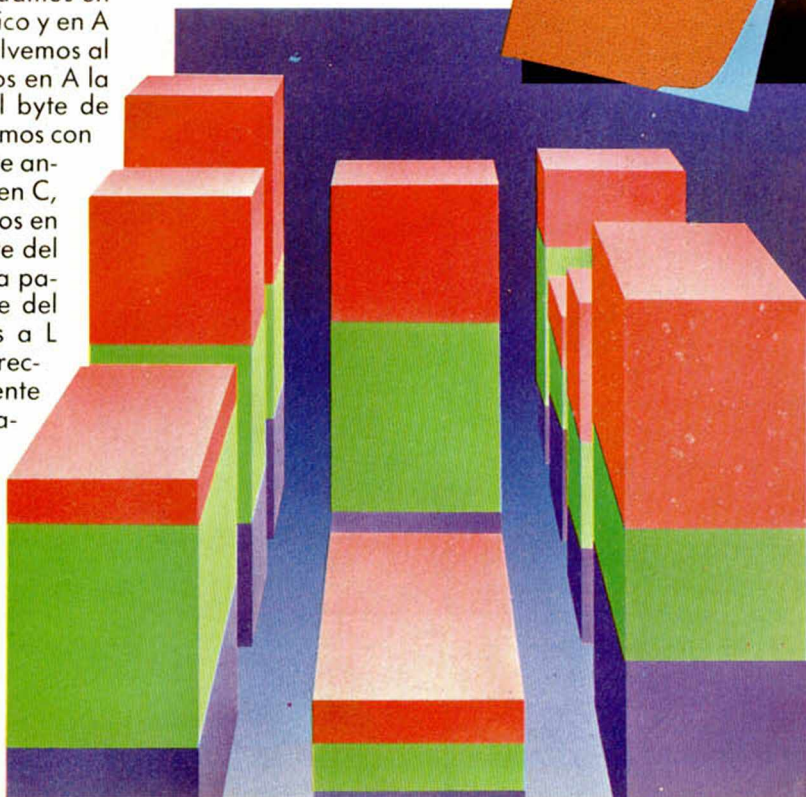
Ahora decrementamos H y juntamos la segunda parte del byte anterior, que está en A con la primera de éste. Lo guardamos temporalmente en L y obtenemos la máscara, hacemos el AND con la pantalla, el OR con el gráfico y lo guardamos en la pantalla. Ahora volvemos al juego 2 para cerrar el bucle de cada byte de un scan. Sumamos DE a HL e IY para saltarnos la parte del gráfico que no ha de ser dibujada y de nuevo al juego 1. Ahora tenemos que dibujar la segunda parte del último byte del scan que no ha sido dibujada dentro del bucle. Al igual que como ocurría con el primer scan, con este último debemos inventarnos un byte siguiente, del que el gráfico sea 0 y la máscara 255. En PONUDO juntamos la primera parte del

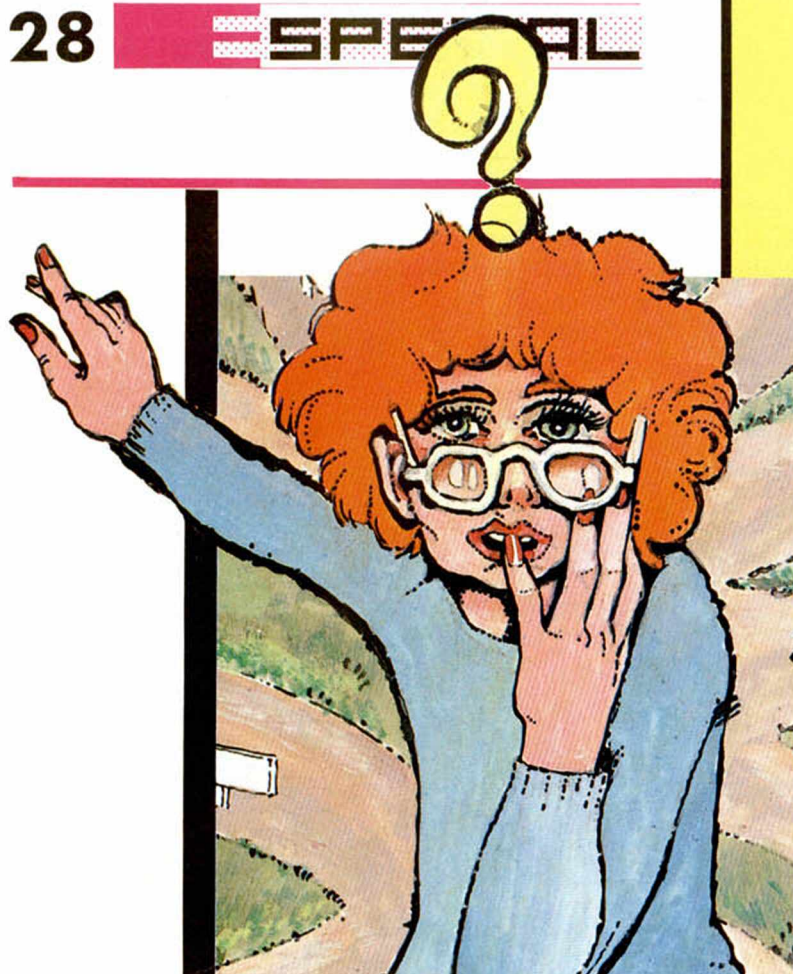
imaginario byte de máscara (colocada anteriormente en PONUDO+1) con la segunda del byte que ha quedado sin dibujar. Ha-

remos el AND con la pantalla y el OR con el gráfico y lo metemos en pantalla. Ahora le sumamos a la dirección de pantalla el valor necesario para calcular la dirección del siguiente scan (dicho valor se encontraba en SUMVAL+1).

Por último, sólo nos queda volver al juego 2 y cerrar el bucle de los scans. Ahora nos queda dibujar los atributos. Para ello, cargamos en HL la dirección destino de los atributos en la pantalla de memoria, en ATDIRG+1 el número de columnas que hay que rellenar, en SUMATT+1, 33-el número anterior (ya deberíais saber para qué lo vamos a utilizar), en E el byte de atributos, en D una máscara para las partes del atributo de la pantalla que no deben cambiar (el papel y el flash), y en A el número de filas a rellenar. En ATLAFX se cierra el bucle para cada fila, y en ATBLDI el de cada byte de la fila. Dentro de éste, tomamos el byte de atributos de la pantalla, nos quedamos con el PAPEL y el FLASH mediante un AND D, lo juntamos con INK y BRIGHT y lo guardamos en la pantalla. Por lo demás, es prácticamente idéntico este proceso al de volcar los atributos de pantalla a SPARES o de SPARES a pantalla, que ya ha sido visto. Tras dibujar los atributos, recuperamos A, que nos decía cuántos gráficos había que dibujar, y actualizamos IX para que apunte al siguiente elemento.

Con esto hemos terminado todo lo relacionado con el programa principal. Lo único que quedan son las tres subrutinas anexas, INICIO, DESACT y COMCHO, que son tan sencillas y poco interesantes y que no merecen una explicación a fondo.





SUGERENCIAS Y ADVERTENCIAS

Tal y como está, la zona de pantalla sobre la que se dibujan los sprites es toda la parte superior de ésta, las 22 líneas, pero no hay razón para que la zona usada tenga la altura que queramos. Por ejemplo, podemos reservar los dos tercios superiores para la pantalla de juego y utilizar el tercio inferior para marcadores y otras cosas. Cuanto más pequeña sea la zona destinada a sprites, mayor será la velocidad a la que vayan nuestros programas. Los ajustes que deberemos hacer son los siguientes: Si queremos que la zona en la que aparezcan los sprites tenga un alto de N filas, tendremos que hacer POKE 62400,X, donde X es igual a N+1 si N está entre 1 y 8, a N+2 si N está entre 9 y 16, y a N+3 si N es mayor de 16. Si no queremos que esta zona empiece justo arriba de la pantalla po-

demos pokear en las direcciones 62387 y 62388 la dirección de la pantalla del ordenador dónde queremos que empiece, y en 62396 y 62397 la dirección de atributos.

Es necesario advertir que si usamos demasiados sprites o demasiado grandes, se puede producir el bloqueo del ordenador. La única solución a esto, como ya hemos dicho, es cambiar en el listado Ensamblador la ubicación de la zona SPARES.

Por último advertirles a los usuarios del Spectrum 128 o +2 que el programa funcionará perfectamente en modo 48 K, pero en 128 sólo funcionará si no intentamos interrumpir el programa con las interrupciones activadas, pues en este momento se bloqueará el ordenador.

Esperamos que este programa os sea de utilidad y consigáis con él hacer programas de calidad comercial.

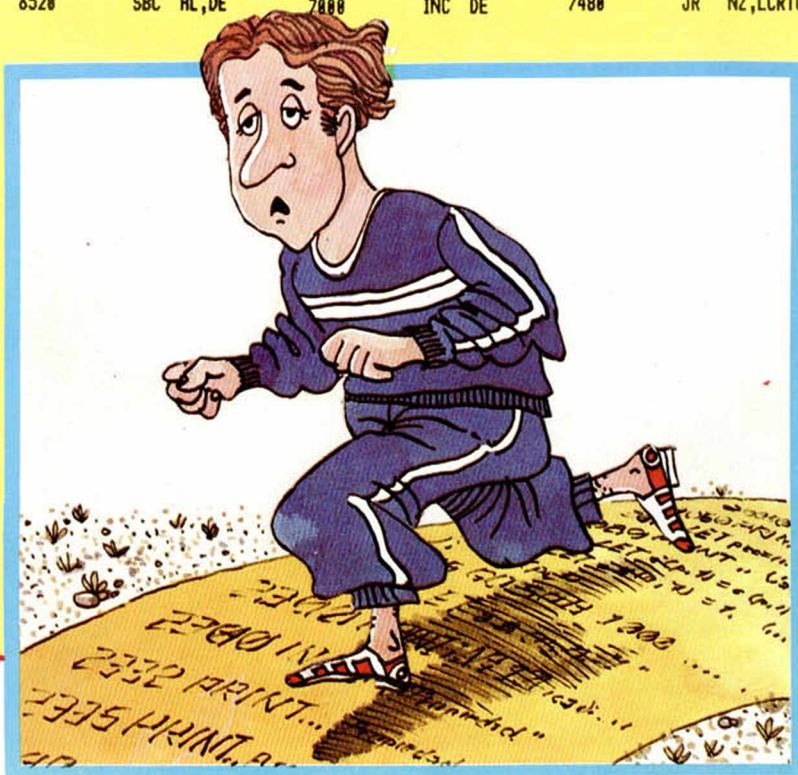
LISTADO ENSAMBLADOR

10	*D+	770	INSABO	LD	A,1
20	*C-	780		LD	(NUBLBO),A
30	;	790		LD	DE, SPARES
40	;	800		LD	IX,TABLDI
50	;	810		LD	IY,TASPRI
60	;	820		LD	A,(NUMSPR)
70	;	830		LD	B,A
80	;	840		AND	A
90	;	850		JR	Z,_NOSPRI
100	;	860	INMAIN	PUSH	BC
110		870		PUSH	DE
120	;	880		CALL	MOVERR
130	;	890		POP	DE
140	ENTINT	900		CALL	CREADA
150		910		LD	BC,17
160		920		ADD	IY,BC
170		930		POP	BC
180		940		DJNZ	INMAIN
190		950		CALL	DIBUJA
200		960	NOSPRI	POP	IY
210		970		PUSH	IY
220		980		RST	56
230		990	SALINT	POP	IY
240		1000		POP	HL
250		1010		POP	DE
260		1020		POP	BC
270		1030		POP	AF
280		1040		EX	AF,AF'
290		1050		EXX	
300		1060		POP	IX
310		1070		POP	HL
320		1080		POP	DE
330		1090		POP	BC
340	;	1100		POP	AF
350	NOVOLC	LD	HL, SPARES		
360		XOR	A		
370		LD	(ESTADO),A		
380		LD	A,(NUBLBO)		
390		LD	IX,TABLDI		
400	INBOTO	DEC	A		
410		JR	Z,INSABO		
420		EX	AF,AF'		
430		LD	E,(IX+8)		
440		LD	D,(IX+1)		
450		LD	A,(IX+14)		
460		LD	(BORPOI+1),A		
470		LD	(BORPOD+1),A		
480		NEG			
490		ADD	A,33		
500		LD	(AUMPOI+1),A		
510		LD	(AUMPOD+1),A		
520		LD	A,(IX+6)		
530		LD	B,8		
540	BORPOI	LD	C,8		
550		LDIR			
560	AUMPOI	LD	C,8		
570		EX	DE,HL		
580		ADD	HL,BC		
590		EX	DE,HL		
600		DEC	A		
610		JP	NZ,BORPOI		
620		LD	A,(IX+13)		
630		LD	E,(IX+11)		
640		LD	D,(IX+12)		
650	BORPOD	LD	C,8		
660		LDIR			
670	AUMPOD	LD	C,8		
680		EX	DE,HL		
690		ADD	HL,BC		
700		EX	DE,HL		
710		DEC	A		
720		JP	NZ,BORPOD		
730		LD	C,15		
740		ADD	IX,BC		
750		EX	AF,AF'		
760		JP	INBOTO		
770		INSABO	LD	A,1	
780			LD	(NUBLBO),A	
790			LD	DE, SPARES	
800			LD	IX,TABLDI	
810			LD	IY,TASPRI	
820			LD	A,(NUMSPR)	
830			LD	B,A	
840			AND	A	
850			JR	Z,_NOSPRI	
860	INMAIN	PUSH	BC		
870		PUSH	DE		
880		CALL	MOVERR		
890		POP	DE		
900		CALL	CREADA		
910		LD	BC,17		
920		ADD	IY,BC		
930		POP	BC		
940		DJNZ	INMAIN		
950		CALL	DIBUJA		
960	NOSPRI	POP	IY		
970		PUSH	IY		
980		RST	56		
990	SALINT	POP	IY		
1000		POP	HL		
1010		POP	DE		
1020		POP	BC		
1030		POP	AF		
1040		EX	AF,AF'		
1050		EXX			
1060		POP	IX		
1070		POP	HL		
1080		POP	DE		
1090		POP	BC		
1100		POP	AF		
1110		EI			
1120		RETI			
1130	;				
1140	;				
1150	VUELCA	LD	HL,ORIGSC		
1160		LD	DE,16384		
1170		LD	C,8		
1180		EXX			
1190		LD	HL,ORIGAT		
1200		LD	DE,22528		
1210		LD	BC,25*256		
1220	LVOLCA	EXX			
1230		LD	B,9		
1240	LOPPNU	LDI			
1250		LDI			
1260		LDI			
1270		LDI			
1280		LDI			
1290		LDI			
1300		LDI			
1310		LDI			
1320		LDI			
1330		LDI			
1340		LDI			
1350		LDI			
1360		LDI			
1370		LDI			
1380		LDI			
1390		LDI			
1400		LDI			
1410		LDI			
1420		LDI			
1430		LDI			
1440		LDI			
1450		LDI			
1460		LDI			
1470		LDI			
1480		LDI			
1490		LDI			
1500		LDI			
1510		LDI			
1520		LDI			

R DE MANEJO DE PANTALLAS

1530	LDI	2290	LD A,(HL)	3850 ;		3810	SUB C	4570	SRL C	
1540	LDI	2300	LD (1Y+0),A	3860 CREADA	LD A,(1Y+4)	3820	LD B,A	4580	SRL C	
1550	LDI	2310	INC HL	3870	LD C,A	3830	JP PARTEY	4590	LD A,(1Y+6)	
1560	LD A,E	2320	LD A,(HL)	3880	AND 6	3840 INTEGE	LD A,(1Y+2)	4600	ADD A,(1Y+3)	
1570	SUB 32	2330	LD (1Y+1),A	3890	ADD A,248	3850	LD (1X+7),A	4610	ADD A,7	
1580	LD E,A	2340	INC HL	3100	LD (1X+9),A	3860	XOR A	4620	SRL A	
1590	JR C,NOINCD	2350	JP RECUPE	3110	LD A,(1Y+5)	3870	LD (1X+8),A	4630	SRL A	
1600	INC D	2360 TRANOR	LD E,A	3120	AND A	3880	JP RECOIN	4640	SRL A	
1610 NOINCD	INC HL	2370	INC HL	3130	JR Z,XPOSIT	3890 PARTEY	LD A,(1Y+7)	4650	SUB C	
1620	DJNZ LOPPNU	2380	LD A,(HL)	3140	INC A	3900	AND A	4660	LD (1X+13),A	
1630	LD A,E	2390	INC HL	3150	RET NZ	3910	JR Z,YPOSIT	4670	EX AF,AF'	
1640	ADD A,32	2400	LD (1Y+13),L	3160	LD A,C	3920	INC A	4680	LD C,A	
1650	LD E,A	2410	LD (1Y+14),H	3170	ADD A,6	3930	RET NZ	4690	JP COYPOS	
1660	JR C,SITERC	2420	JP COCORE	3180	JR NC,NEGARE	3940	LD A,(1Y+6)	4700 MOSCAB	LD A,192	
1670	LD A,D	2430 RECLIN	LD A,(1Y+10)	3190	LD HL,ORIGAT-1	3950	ADD A,(1Y+3)	4710	SUB (1Y+6)	
1680	SUB 8	2440	LD E,A	3200	LD BC,ORIGSC-1	3960	RET NC	4720	LD (1X+6),A	
1690	LD D,A	2450	LD A,(1Y+11)	3210	LD A,(1Y+2)	3970	RET Z	4730	ADD A,7	
1700 SITERC	EXX	2460 COCORE	LD C,A	3220	LD (1X+7),A	3980	LD (1X+6),A	4740	SRL A	
1710	LDI	2470	CP 128	3230	XOR A	3990	NEG	4750	SRL A	
1720	LDI	2480	CCF	3240	LD (1X+8),A	4000	ADD A,(1Y+3)	4760	SRL A	
1730	LDI	2490	SBC A,A	3250	JP PARTEY	4010	PUSH HL	4770	LD (1X+13),A	
1740	LDI	2500	LD B,A	3260 NEGARE	SCF	4020	PUSH DE	4780 COYPOS	LD (1X+0),C	
1750	LDI	2510	LD L,(1Y+6)	3270	RRA	4030	LD L,(1X+2)	4790	LD (1X+1),B	
1760	LDI	2520	LD H,(1Y+7)	3280	SRA A	4040	LD H,(1X+3)	4800	LD L,(1X+2)	
1770	LDI	2530	ADD HL,BC	3290	SRA A	4050	LD D,0	4810	LD H,(1X+3)	
1780	LDI	2540	LD (1Y+6),L	3300	LD B,A	4060	LD E,(1Y+2)	4820	LD E,(1Y+15)	
1790	LDI	2550	LD (1Y+7),H	3310	ADD A,(1Y+2)	4070 NESUDI	ADD HL,DE	4830	LD D,(1Y+16)	
1800	LDI	2560	LD A,E	3320	RET NC	4080	DEC A	4840	ADD HL,DE	
1810	LDI	2570	LD C,A	3330	RET Z	4090	JP NZ,NESUDI	4850	LD (1X+4),L	
1820	LDI	2580	CP 128	3340	LD (1X+7),A	4100	LD (1X+2),L	4860	LD (1X+5),H	
1830	LDI	2590	CCF	3350	LD A,B	4110	LD (1X+3),H	4870	LD D,(1X+7)	
1840	LDI	2600	SBC A,A	3360	NEG	4120	LD A,(1X+6)	4880	LD A,(1X+9)	
1850	LDI	2610	LD B,A	3370	LD (1X+8),A	4130	ADD A,7	4890	AND 6	
1860	LDI	2620	LD L,(1Y+4)	3380	ADD A,(1X+2)	4140	SRL A	4900	JR Z,SANUBY	
1870	LDI	2630	LD H,(1Y+5)	3390	LD (1X+2),A	4150	SRL A	4910	INC D	
1880	LDI	2640	ADD HL,BC	3400	LD A,0	4160	SRL A	4920 SANUBY	LD (1X+14),D	
1890	LDI	2650	LD (1Y+4),L	3410	ADC A,(1X+3)	4170	LD (1X+13),A	4930	POP DE	
1900	LDI	2660	LD (1Y+5),H	3420	LD (1X+3),A	4180	JP COYPOS	4940	POP HL	
1910	LDI	2670	LD A,(1Y+12)	3430	LD A,C	4190 YPOSIT	LD A,(1Y+6)	4950	PUSH HL	
1920	LDI	2680	LD E,(1Y+8)	3440	LD HL,ORIGAT-1	4200	CP 192	4960	LD (1X+11),L	
1930	LDI	2690	BIT 3,D	3450	LD BC,ORIGSC-1	4210	RET NC	4970	LD (1X+12),H	
1940	LDI	2700	JP NZ,ADETRA	3460	AND 6	4220	SRL A	4980	LD H,B	
1950	LDI	2710	INC A	3470	JR NZ,PARTEY	4230	SRL A	4990	LD L,C	
1960	LDI	2720	CP E	3480	INC HL	4240	SRL A	5000	LD A,(1X+14)	
1970	LDI	2730	JR NZ,NOFICI	3490	INC BC	4250	PUSH BC	5010	LD (PUBYSA+1),A	
1980	LDI	2740	XOR A	3500	JP PARTEY	4260	LD BC,33	5020	LD (PUATSA+1),A	
1990	LDI	2750 NOFICI	LD (1Y+12),A	3510 XPOSIT	LD A,C	4270	INC A	5030	NEG	
2000	LDI	2760	JP TRAREN	3520	SUB 1	4280 CALATT	DEC A	5040	ADD A,33	
2010	LDI	2770 ADETRA	INC A	3530	JR C,INTEGE	4290	JR Z,ATCOMP	5050	LD (PUSUSA+1),A	
2020	LDI	2780	LD D,A	3540	SRL A	4300	ADD HL,BC	5060	LD (ATSUSA+1),A	
2030	INC HL	2790	SUB E	3550	SRL A	4310	JP CALATT	5070	LD A,(NUBLBO)	
2040	DEC B	2800	SUB E	3560	SRL A	4320 ATCOMP	POP BC	5080	INC A	
2050	JP NZ,LVOLCA	2810	ADD A,2	3570	ADD A,(1Y+2)	4330	PUSH HL	5090	LD (NUBLBO),A	
2060	RET	2820	LD (PARETR+1),A	3580	CP 33	4340	PUSH DE	5100	LD A,(1X+6)	
2070 ;		2830	JR Z,FIANCI	3590	JR C,INTEGE	4350	LD H,B	5110	LD B,0	
2080 ;		2840	LD A,D	3600	SUB 32	4360	LD L,C	5120 PUBYSA	LD C,0	
2090 MOVERR	LD A,(1Y+5)	2850 FIANCI	LD (1Y+12),A	3610	LD (1X+8),A	4370	LD DE,33	5130	LDIR	
2100	CP 127	2860	CP E	3620	NEG	4380	LD A,(1Y+6)	5140	PUSUSA	LD C,0
2110	RET Z	2870	JR C,TRAREN	3630	ADD A,(1Y+2)	4390	AND A	5150	ADD HL,BC	
2120	LD A,(1Y+9)	2880 PARETR	LD A,0	3640	LD (1X+7),A	4400	JR Z,PRIFIL	5160	DEC A	
2130	LD D,A	2890	NEG	3650 RECOIN	LD A,C	4410 CALFIL	ADD HL,DE	5170	JR NZ,PUBYSA	
2140	AND 71	2900 TRAREN	LD E,(1Y+15)	3660	SRL A	4420	DEC A	5180	POP HL	
2150	LD (1X+10),A	2910	LD D,(1Y+16)	3670	SRL A	4430	JP NZ,CALFIL	5190	LD A,(1X+13)	
2160	BIT 4,D	2920	SLA E	3680	SRL A	4440	LD B,H	5200 PUATSA	LD C,0	
2170	JR Z,RECLIN	2930	RL D	3690	LD C,A	4450	LD C,L	5210	LDIR	
2180	LD L,(1Y+13)	2940	LD L,(1Y+0)	3700	LD HL,ORIGAT	4460 PRIFIL	LD A,(1Y+6)	5220 ATSUSA	LD C,0	
2190	LD H,(1Y+14)	2950	LD H,(1Y+1)	3710	ADD A,L	4470	ADD A,(1Y+3)	5230	ADD HL,BC	
2200 RECUPE	LD A,(HL)	2960	AND A	3720	LD L,A	4480	JR C,MOSCAB	5240	DEC A	
2210	CP 127	2970	JR Z,PRIFAS	3730	ADC A,H	4490	CP 193	5250	JP NZ,PUATSA	
2220	JR NZ,NOFITA	2980	LD B,A	3740	SUB L	4500	JR NC,MOSCAB	5260	LD C,15	
2230	LD L,(1Y+10)	2990 MULTIP	ADD HL,DE	3750	LD H,A	4510	LD A,(1Y+3)	5270	ADD IX,BC	
2240	LD H,(1Y+11)	3000	DJNZ MULTIP	3760	LD A,C	4520	LD (1X+6),A	5280	RET	
2250	LD A,(HL)	3010 PRIFAS	LD (1X+2),L	3770	LD BC,ORIGSC	4530	LD A,C	5290 ;		
2260 NOFITA	CP 126	3020	LD (1X+3),H	3780	ADD A,C	4540	EX AF,AF'	5300 ;		
2270	JR NZ,TRANOR	3030	RET	3790	LD C,A	4550	LD C,(1Y+6)	5310 DIBUJA	LD IX,TABLDI	
2280	INC HL	3040 ;		3800	ADC A,B	4560	SRL C	5320	LD A,(NUBLBO)	

5330	DIBLUP	DEC	A	6050	SUB	L	4530	RET	7810	POP	HL	7490	LD	A,1				
5340		RET	Z	6060	LD	H,A	4540	COORDS	CALL	DITABS	7820	INC	H	7500	LD	(NUBLBO),A		
5350		PUSH	AF	6070	EX	DE,HL	4550	CALPOI	JP	0	7830	LD	A,H	7510	XOR	A		
5360		LD	E,(IX+0)	6080	EXC		4560	COORDX	LD	L,(IY+4)	7840	AND	7	7520	LD	(NUMSPR),A		
5370		LD	D,(IX+1)	6090	DEC	C	4570	LD	H,(IY+5)	7850	JR	NZ,INIPRT	7530	LD	(ESTADO),A			
5380		LD	H,(IX+9)	6100	JP	NZ,LOSCAN	4580	RET		7860	LD	A,L	7540	EI				
5390		LD	L,255	6110	LD	L,(IX+11)	4590	COORDY	LD	L,(IY+6)	7870	ADD	A,32	7550	RET			
5400		LD	A,(HL)	6120	LD	H,(IX+12)	4600	LD	H,(IY+7)	7880	LD	L,A	7560	;				
5410		LD	(PONUDO+1),A	6130	LD	A,(IX+14)	4610	RET		7890	JR	C,INIPRT	7570	;				
5420		INC	H	6140	LD	(ATDIRG+1),A	4620	DITABS	LD	IY,TASPRI	7100	LD	A,H	7580	DESACT	LD	A,63	
5430		LD	A,(HL)	6150	NEG		4630	AND	A	7110	SUB	8	7590	IM	1			
5440		DEC	H	6160	ADD	A,33	4640	RET	Z	7120	LD	H,A	7600	LD	1,A			
5450		LD	(POSNUM+1),A	6170	LD	(SUMATT+1),A	4650	PUSH	DE	7130	INIPRT	EX	AF,AF'	7610	RET			
5460		EXC		6180	LD	E,(IX+10)	4660	LD	DE,17	7140	DEC	A	7620	;				
5470		LD	L,(IX+2)	6190	LD	D,184	4670	SUMDIS	ADD	IY,DE	7150	JR	NZ,INILAP	7630	;			
5480		LD	H,(IX+3)	6200	LD	A,(IX+13)	4680	DEC	A	7160	LD	HL,22528	7640	COMCHO	LD	BC,(SPRICH)		
5490		PUSH	HL	6210	ATLAFX	EX	AF,AF'	4690	JR	NZ,SUMDIS	7170	LD	DE,ORIGAT	7650	INC	C		
5500		POP	IY	6220	ATDIRG	LD	B,0	4700	POP	DE	7180	LD	A,24	7660	JR	Z,BCONTO		
5510		LD	L,(IX+4)	6230	ATBLDI	LD	A,(HL)	4710	RET		7190	INATLA	LD	BC,32	7670	DEC	C	
5520		LD	H,(IX+5)	6240	AND	D	4720	;		7200	LDIR		7680	CALL	CHOSUB			
5530		LD	A,(IX+7)	6250	OR	E	4730	;		7210	INC	DE	7690	LD	BC,0			
5540		LD	(POSBYT+1),A	6260	LD	(HL),A	4740	NUMSPR	EQU	23681	7220	DEC	A	7700	RET	NC		
5550		NEG		6270	INC	HL	4750	TABLDI	EQU	53289	7230	JR	NZ,INATLA	7710	INC	BC		
5560		ADD	A,33	6280	DJNZ	ATBLDI	4760	TASPRI	EQU	53000	7240	IM	2	7720	RET			
5570		LD	(SUMAL+1),A	6290	SUMATT	LD	C,0	4770	ORIGSC	EQU	54568	7250	LD	C,4	7730	;		
5580		LD	C,(IX+6)	6300	ADD	HL,BC	4780	ORIGAT	EQU	68904	7260	LD	H,248	7740	BCONTO	LD	IX,SPRICH	
5590		LD	E,(IX+8)	6310	EX	AF,AF'	4790	SPARES	EQU	53544	7270	LCRTCR	LD	A,9	7750	LD	A,(NUMSPR)	
5600		LD	D,0	6320	DEC	A	4800	SPRICH	EQU	23728	7280	SUB	C	7760	BUCHAL	INC	(IX+0)	
5610	LOSCAN	EXC		6330	JP	NZ,ATLAFX	4810	;		7290	SUB	C	7770	EX	AF,AF'			
5620	POSNUM	LD	BC,0	6340	POP	AF	4820	;		7300	LD	L,0	7780	LD	BC,(SPRICH)			
5630		EXC		6350	LD	C,15	4830	ORG	61953	7310	ROTACU	LD	B,L	7790	LD	A,B		
5640	POSBYT	LD	B,0	6360	ADD	IX,BC	4840	INICIO	DI	7320	LD	E,0	7800	CP	C			
5650	LOBYTS	LD	A,(IY+8)	6370	JP	DIBLUP	4850	LD	A,241	7330	LD	D,A	7810	JR	Z,SASPRI			
5660		INC	IY	6380	;		4860	LD	I,A	7340	VERROT	DEC	D	7820	CALL	CHOSUB		
5670		EX	AF,AF'	6390	;		4870	LD	B,0	7350	JR	Z,VEREXI	7830	LD	BC,1			
5680		LD	A,(HL)	6400	ESTADO	DEFB	0	4880	LD	HL,61696	7360	SRL	B	7840	RET	C		
5690		INC	HL	6410	NUBLBO	DEFB	1	4890	INTCRE	LD	(HL),242	7370	RR	E	7850	SASPRI	EX	AF,AF'
5700		EXC		6420	;		4900	INC	HL	7380	JR	VERROT	7860	DEC	A			
5710		LD	L,A	6430	;		4910	DJNZ	INTCRE	7390	VEREXI	LD	(HL),B	7870	JR	NZ,BUCHAL		
5720		LD	A,(HL)	6440	DIMEXS	LD	E,(IY+2)	4920	LD	(HL),242	7400	INC	H	7880	DEC	BC		
5730		OR	C	6450	SLA	E	4930	LD	DE,ORIGSC	7410	LD	(HL),E	7890	RET				
5740		INC	H	6460	SLA	E	4940	LD	HL,16384	7420	DEC	H	7900	;				
5750		LD	C,(HL)	6470	SLA	E	4950	LD	A,192	7430	DEC	L	7910	CHOSUB	LD	HL,COORDX		
5760		EX	AF,AF'	6480	JP	CONTIN	4960	INILAP	EX	AF,AF'	7440	JR	NZ,ROTACU	7920	LD	(CALPOI+1),HL		
5770		LD	L,A	6490	DIMEYS	LD	E,(IY+3)	4970	PUSH	HL	7450	INC	H	7930	LD	HL,DIMEXS		
5780		LD	A,B	6500	CONTIN	LD	D,0	4980	LD	BC,32	7460	INC	H	7940	LD	(ANCAIT+1),HL		
5790		LD	B,(HL)	6510	AND	A	4990	LDIR		7470	DEC	C	7950	CALL	CHOREA			
5800		DEC	H	6520	SBC	HL,DE	7800	INC	DE	7480	JR	NZ,LCRTCR	7960	RET	NC			
5810		OR	(HL)										7970	LD	HL,COORDY			
5820		LD	L,A										7980	LD	(CALPOI+1),HL			
5830		EX	AF,AF'										7990	LD	HL,DIMEYS			
5840		EX	DE,HL										8000	LD	(ANCAIT+1),HL			
5850		AND	(HL)										8010	CHOREA	LD	A,B		
5860		OR	E										8020	CALL	COORDS			
5870		LD	(HL),A										8030	EX	DE,HL			
5880		INC	HL										8040	LD	A,C			
5890		EX	DE,HL										8050	CALL	COORDS			
5900		EXC											8060	AND	A			
5910		DJNZ	LOBYTS										8070	SBC	HL,DE			
5920		ADD	HL,DE										8080	LD	A,B			
5930		ADD	IY,DE										8090	JR	NC,BIENCO			
5940		EXC											8100	LD	A,H			
5950		LD	A,C										8110	CPL				
5960	PONUDO	OR	0										8120	LD	H,A			
5970		EX	DE,HL										8130	LD	A,L			
5980		AND	(HL)										8140	CPL				
5990		OR	B										8150	LD	L,A			
6000		LD	(HL),A										8160	INC	HL			
6010	SUNMAL	LD	A,0										8170	LD	A,C			
6020		ADD	A,L										8180	BIENCO	CALL	DITABS		
6030		LD	L,A										8190	ANCAIT	JP	0		
6040		ADC	A,H															



6 GRANDES EXITOS EN UNO

MÁS UN JUEGO GRATIS (DUET)

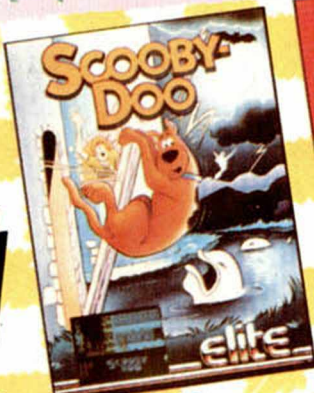
1.750 Ptas.
VERSION CASSETTE

1750 PTAS

7 PROGRAMAS = 250 PTAS

CADA JUEGO

6 PAK



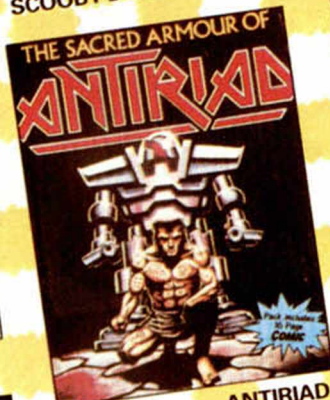
SCOOBY DOO



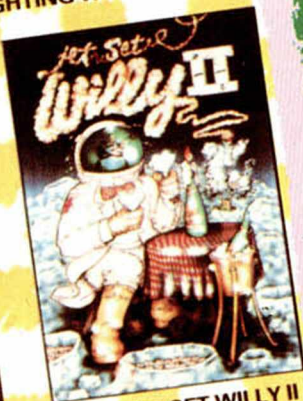
FIGHTING WARRIOR



1942



ANTIRIAD



JET SET WILLY II



SPLIT PERSONALITIES



DUET

PLUS BONUS GAME -
DUET. PREVIOUSLY
UNRELEASED,
SIMULTANEOUS
TWO-PLAYER ACTION.

6 PAK

DISPONIBLE EN
Spectrum
Commodore
Amstrad
Amstrad Disk



David LÓPEZ GUAITA

MOVIMIENTO

Y TECLADO

Con esta rutina no sólo podrás definir las teclas y mover un sprite principal por la pantalla, sino trasladar gráficos secundarios al mismo tiempo, borrar zonas de la pantalla e incluso colorear zonas de atributos.

Action es una rutina formada por tres partes diferenciadas: una rutina de definición de teclas o elección de joystick, un programa que se encarga de chequear el teclado, actualizar la posición de un objeto en pantalla y escribir los nuevos datos en un buffer y la rutina por interrupciones que imprime en pantalla todos los gráficos que haya en el buffer, vaciándolo.

Con esta rutina sacarás el máximo rendimiento si la usas desde Código Máquina con esta rutina.

FUNCIONAMIENTO

Para describir el funcionamiento, vamos a hacerlo dividiendo el programa en sus tres partes:

Definición de teclas.

Con esta rutina comienza el programa. A grandes rasgos, el programa pregunta en principio si se quiere usar con joystick Kempston o redefinir las teclas. Si se redefinen las teclas, el ordenador chequeará una a una todas las semifilas del teclado. Si hay alguna tecla pulsada, el ordenador almacenará sus datos en la memoria, y pasará a la siguiente dirección para definir otra tecla, hasta acabar.

El inicio está en la

dirección 63450; en las líneas 70-80 el ordenador abre el canal dos de la pantalla, llamando a una rutina de la ROM. Después, se carga en DE la dirección de un texto a imprimir, en BC la duración del texto y se llama a la subrutina de la ROM PR-STRING, que se encarga de imprimir todo el texto.

Así, con el primer menú ya en pantalla, el ordenador leerá la primera semifila del teclado para ver si se ha pulsado alguna tecla. Como probablemente ya sabrás, el teclado se divide en ocho semifilas (fig.1).

Para leer una semifila hay que describir en el acumulador el dato de esa semifila, (los números de la figura 1),





para después hacer un IN A,(FE). Con esto cargaremos en la mitad superior del registro de direcciones el valor del registro A, y en la mitad inferior el valor FEh. Al ejecutarlo, se nos devuelve en el registro A un valor con las teclas pulsadas en la semifila.

Si no hay ninguna tecla pulsada, el valor es XXX11111. Si hubiera alguna tecla pulsada, el valor de uno de los cinco primeros bits se pondría a 0. De esta forma, el primer bit se pondría a cero si pulsáramos la tecla de la semifila más alejada del centro del teclado, el segundo bit si pulsáramos la siguiente tecla, etc.

En la rutina, el ordenador chequea la semifila 1-5, y si hay una tecla pulsada, (1 ó 2), sus bits se pondrán a 0, y el ordenador saltará a las

distintas partes del programa.

Si se ha pulsado la tecla 1 para jugar con el joystick Kempston, el ordenador irá a KEMPS, en el cual simplemente cambiará una dirección de salto de la rutina de control del sprite para que en vez de chequear el teclado teclee el joystick, y después volverá al Basic. Si se pulsa la tecla 2 se procede a observar todo el teclado con

la rutina DEFIN. Lo primero que se hace es imprimir otro texto, (líneas 250-270). Se coloca en la dirección de salto de la rutina número 2 el valor de teclado, por si hubiera sido cambiado por error, y se carga en HL el valor de una dirección de tabla. Entre la línea 330 y 380 se borran los ocho bytes de la tabla. En esta tabla iremos escribiendo los datos de cada tecla seleccionada: byte de mayor peso del bus de direcciones y valor de la semifila si la tecla está pulsada.

Después de limpiar la tabla se salta a ESPERA, rutina entre las líneas 1370 y 1420 que se encarga de detener la ejecución hasta que no haya ninguna tecla pulsada.

El registro L contiene el número de la tecla que estamos redefiniendo. Desde la línea 410 hasta la 690 se ponen los atributos de una determinada opción de la pantalla en FLASH 1, y los de la opción anterior en FLASH 0. Esto lo hace en función del registro L, que tiene el número de opción que se está preguntando.

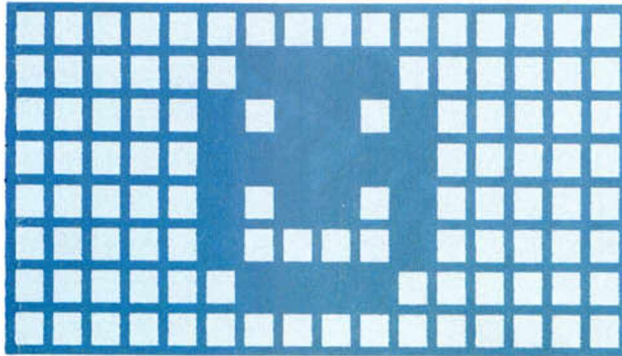
A partir de la línea 700 empieza la rutina de definición propiamente dicha. Se carga en A el valor 0111111b. Esto se hace por la siguiente razón; si observas el valor que hay que darle al registro A, a primera vista no parece haber ninguna relación entre ellos. Pero el poner esos valores en binario, observa el resultado:

```
7Fh=01111111b
BFh=10111111b
DFh=11011111b
EFh=11101111b
F7h=11110111b
FBh=11111011b
FDh=11111101b
FEh=11111110b
```

Como pueden observar, para chequear el teclado se puede cargar en A el valor de la semifila, y haciendo ocho rotaciones podemos leer todas las semifilas del teclado.

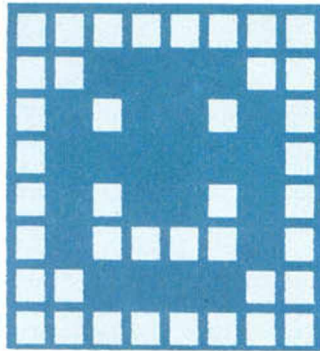
Esto es precisamente lo que ocurre entre las líneas 700 y 790. El programa va rotando A, y comparando los valores de la semifila seleccionada con 00011111b; si alguna tecla está pulsada, se salta a INTEC.





En INTEC cargamos en E el valor de la semifila que tiene alguna tecla pulsada, y entre las líneas siguientes cargamos en D el valor que se ha devuelto en el registro A con las teclas pulsadas.

Desde la línea 870 hasta la 970, se observa si en la semifila hay dos teclas pulsadas al mismo tiempo, y si es así, se vuelve hacia el inicio de la redefinición.

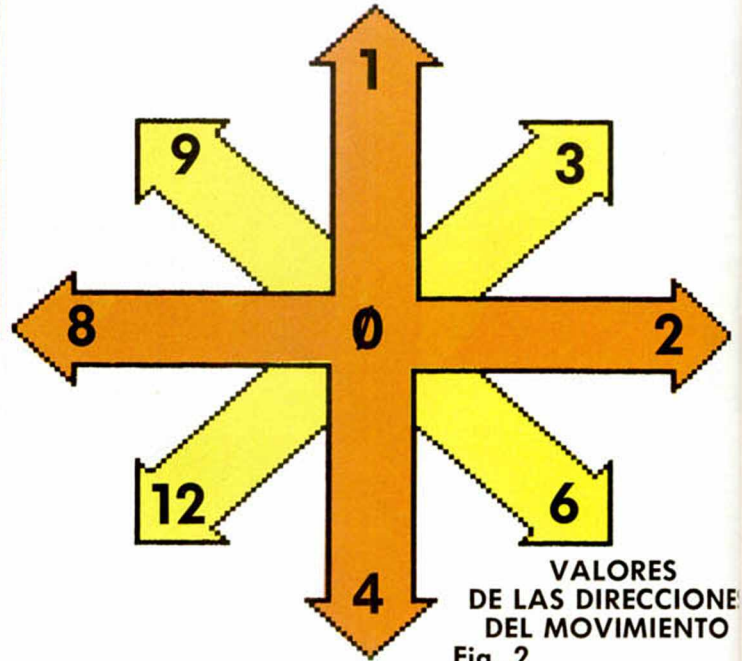


En ONMASC se comparan todos los datos de teclas ya seleccionadas con los datos de la actual, y si se han seleccionado dos teclas iguales de nuevo se vuelve al inicio de la rutina de definición, en CHECK. Si se han comprobado todas las anteriores teclas y ninguna es igual a la anterior se salta a TODOS. Allí se calcula la dirección en la tabla donde pondremos los datos de la tecla que ha sido pulsada.

Al hacer esto, se llama a SONY, subrutina que produce un ruido. Observa en la rutina SONY (líneas 1430-1520) como para preservar el color del borde lo tomamos de la variable BORDCR (23624), antes de hacer el sonido.

Una vez hecho todo esto, volvemos a la rutina principal. Cargamos de nuevo L con el valor de la opción que estamos preguntando, y si esta es ya 4, es decir, se han seleccionado ya las cuatro teclas, volvemos al Basic.

Si aún no son cuatro las teclas seleccionadas, volvemos a PREG hasta acabar.



VALORES DE LAS DIRECCIONES DEL MOVIMIENTO
Fig. 2

ACTUALIZACIÓN DE POSICIÓN DESPUÉS DE CHEQUEAR EL TECLADO

Empieza con ZCONTR, donde verás dos «DEFB». Con ellos se hace en realidad un salto a la dirección que en principio es KEYBO. Esta dirección de salto es precisamente la que cambiábamos al seleccionar la opción de joystick Kempston. Al hacer esto, la rutina saltará a KEYBO o a

JOYST, según lo hayamos definido antes.

A continuación voy a explicar los datos y las variables que usa la rutina, los cuales empiezan en la dirección 65500:

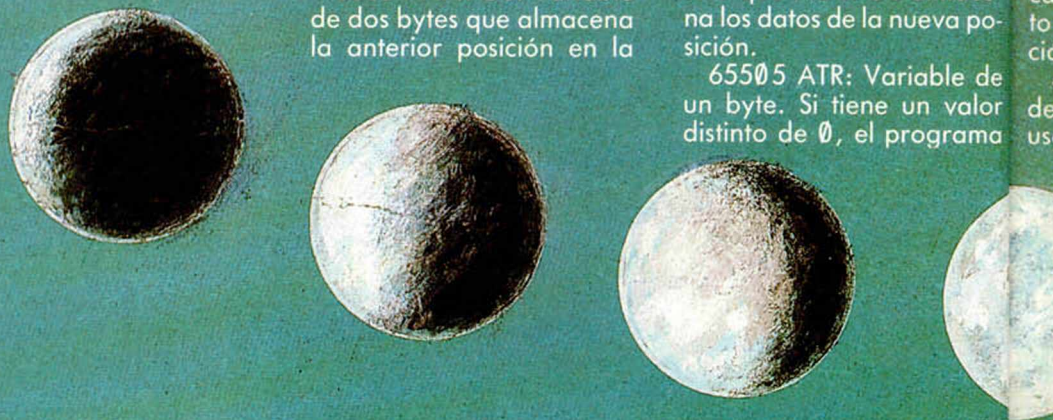
65500: Variable de un byte SWITCH. Se usa para seleccionar entre dos posibles formas de impresión del gráfico principal, una lenta y otra rápida. Las ventajas y desventajas de las rutinas las explicaré más tarde.

65501 ANPOS: Variable de dos bytes que almacena la anterior posición en la

pantalla del gráfico principal. En 65501 guardamos la posición en el eje X en alta resolución y en 65502 el valor del eje Y también en alta resolución. Al contrario de lo que sucede en Basic con PLOT, las coordenadas en la rutina tienen su origen en el extremo superior izquierdo, con lo cual la variable Y es distinta.

65503 NUEPOS: Variable de dos bytes que almacena la nueva posición del sprite. Es igual que ANPOS, sólo que NUEPOS almacena los datos de la nueva posición.

65505 ATR: Variable de un byte. Si tiene un valor distinto de 0, el programa



SEMIFILAS DEL TECLADO

1 #F7 5

6 #EF 0

Q #FB T

Y #DF P

A #FD G

H #BF ENT

CAPS #FE V

B #7F SPA

cambiará los atributos de la pantalla por los que la figura con el valor de ATR. Es decir, si ATR es 4, el ordenador pondrá los atributos de la pantalla con el valor 4.

En cambio, si es cero, el ordenador no cambiará los atributos de la pantalla.

65506 DIMEN: Variable de dos bytes; contiene las dimensiones del gráfico principal. En 65506 guardamos el ancho del sprite en baja resolución, es decir, en caracteres; en 65507 el alto del sprite en alta resolución o número de scans.

65508 HORIZ: Variable de un byte. El programa la usa para almacenar la di-

rección horizontal que lleva el movimiento del gráfico. Es 2 si el movimiento es hacia la derecha, 0 si no hay movimiento horizontal y 8 si el movimiento es hacia la izquierda.

65509 VERTI: Variable de un byte; es similar a la variable anterior, sólo que el programa la utiliza para el movimiento vertical. Su valor cuando el movimiento es hacia arriba es 1; si no hay movimiento vertical es 0 y si el movimiento es hacia abajo es 8.

65510 ANIDIR: Variable de dos bytes en la cual

guardamos la dirección de memoria del último gráfico impreso.

65515 TABLA: Variable de dos bytes que almacena la dirección de inicio de la tabla en la que se guardan los valores de todas las teclas predefinidas. Normalmente su valor es 65515, pero se puede poner en cualquier zona de la memoria.

65512 GRAFIC: Variable de dos bytes. Indica el comienzo de una tabla que

contiene las direcciones de inicio de los gráficos del personaje principal según cual sea su dirección. Es decir, que para cada dirección el ordenador buscará en esta tabla para encontrar el inicio del gráfico que corresponde a esa dirección del objeto. Como esto parece algo lioso, vamos a poner un ejemplo. Observa la figura 2.

En la figura están representadas las sumas que se obtienen para cada movimiento del personaje del contenido de las variables HORIZ y VERTI. Supongamos que el gráfico se mueve en diagonal hacia abajo y hacia la izquierda. Entonces, la suma de VERTI y HORIZ sería 12. El ordenador multiplica por dos este dato, y lo suma a GRAFIC. La posición de memoria resultante es el inicio del gráfico en el cual se representa por ejemplo a nuestro héroe caminando hacia abajo y hacia la izquierda.

Después de haberte explicado la utilidad de cada variable, proseguiré con la explicación del programa.

Según esté SWITCH a 1 o a otro valor, el ordenador imprimirá el gráfico con la subrutina IMPRESS o con la subrutina FASTER. Te preguntaré el proqué de haber puesto dos subrutinas que realizan lo mismo. La respuesta estriba en su propio funcionamiento. Cuando imprimimos un gráfico en la posición 4 por ejemplo en horizontal, tenemos que rotar hacia la derecha cuatro veces cada uno de los bytes del gráfico. Esto hace que la impresión sea muy lenta, especialmente cuando hay que imprimir varios gráficos y éstos son muy grandes. La solución a este problema se encuentra en FASTER. Esta rutina trabaja con los gráficos ya rotados. Supón que quieres imprimir



con FASTER un gráfico que está situado también en la posición 4 del eje X. Entonces, el programa buscará en su memoria dónde empieza el gráfico que ya está rotado cuatro veces, y lo imprimirá en la pantalla. El defecto de FASTER es que ocupa mucha más memoria, ya que hay que introducir un gráfico por cada posible posición. No obstante, hay algunos casos en los que el número de gráficos puede ser de cuatro, de dos o de uno. Si movemos el gráfico de dos en dos pixels, sólo necesitaremos cuatro gráficos, ya que si el ordenador al empezar a mover el sprite comienza en el pixel de coordenada X igual a 0, las posibles posiciones para X serían 0, 2, 4, 6. Si X fuera 8, bastaría con incrementar el número de columna para la impresión.

Por otro lado, IMPRES imprime en OVER 1 y FASTER simplemente imprime encima de lo anterior.

En la figura 2 verás los dos gráficos necesarios para un movimiento que se desplace de cuatro en cuatro pixels.

Otra ventaja que tiene imprimir con FASTER es que con IMPRES hay que imprimir primero el gráfico en la anterior posición para borrarlo y después imprimir el nuevo gráfico. Con FASTER podemos hacerlo imprimiendo un sólo gráfico. El truco consiste en hacer que en el dibujo del sprite hay un marco que tenga de ancho el número de pixels del pasod el movimiento. Así, un sprite se mueve de dos en dos pixels, necesitará un marco alrededor de toda la figura con un ancho de dos pixels. Entonces, con imprimir una vez el gráfico, quedarían borrados los restos del anterior.

Una vez hechas todas estas aclaraciones, voy a se-

guir explicando la rutina de lectura del teclado y actualización de las variables de posición.

Lo primero que se hace una vez que estamos en KEYBO es deshabilitar las interrupciones para tener mayor velocidad y que no se imprima ningún gráfico mientras formamos sus datos. El programa carga en BC el alto y el ancho del sprite; pone las variables HORIZ y VERTI A 0 y carga en IX el valor de NUEPOS. Después carga DE con el valor de NUEPOS y lo copia en ANPOS.

Para chequear el teclado, se carga en HL el inicio de la tabla que contine los datos de todas las teclas, y se van comprobando si esas teclas son pulsadas ahora. Para ello primero se carga en el acumulador el byte de mayor peso del bus de direcciones, luego con el valor devuelto por el teclado se hace un complemento y se hace un OR con el valor de la tecla predefinida. Si el resultado es de 255, quiere decir que la tecla ha sido pulsada. Entonces llama a la subrutina de dirección correspondiente; en ella observa si la nueva posición haría que parte del sprite saliera fuera de la pantalla y si es así vuelve a la rutina principal. En caso negativo, cambia NUEPOS con la nueva posición y pone HORIZ o VERTI como corresponda. Después volverá al programa.

Todo esto se repite hasta que ya hemos observado todas las teclas. Con la rutina de joystick Kempston los pasos son muy similares, pero no hace falta mirar en las teclas de la tabla, basta con tomar el dato del port 233.

Después KEYBO y JOYST confluyen en CAMBIA. Esta parte se ocupa ya de poner los datos en el buffer para que luego se impriman por

interrupciones. La línea 2290 decide si se debe utilizar IMPRES, y si es así salta a LNTIMP. Si no se ha saltado, se utilizará FASTER; en ese caso se copia en ANPOS NUEPOS y se carga en DE. A continuación, tomando el valor de buffer se llama a PUTTER.

Buffer es una variable de un byte situada en la dirección 62000. Contiene el número de sprites que se deben imprimir. Inmediatamente después de la dirección 62000 se colocan los datos de todos los sprites que se deseen imprimir.

PUTTER es una rutina que comienza en la línea 3990; sirve para calcular en qué lugar del buffer se deben introducir los siguientes datos.

Para ello toma el valor del acumulador y lo multiplica por 7 llamando a MULTI, rutina que hace una multiplicación entre dos números de 16 bits y deja el resultado en HL. HL toma así la dirección donde se deben guardar los gráficos y se llama a la rutina principal. En ella IX toma el valor de HL, y pkea con el valor 2.

Es hora de que te explique cómo se colocan los datos para que el impresor por interrupciones haga una cosa u otra.

En primer lugar se carga en el acumulador buffer y se multiplica por 7 para hallar la dirección donde debemos introducir los datos. La rutina de impresión tiene cuatro funciones: imprimir con IMPRES, imprimir con FASTER, borrar una zona de la pantalla y cambiar los atributos de una zona de la pantalla.

Seleccionamos la opción que queremos con un número al principio de cada bloque de 7 datos:

- 1 = IMPRESS
- 2 = FASTER
- 3 = BORRA
- 4 = PAINT

Este número irá con un prefijo delante de otros 6 datos. Estos datos serán, dependiendo de la función que hayamos escogido:

1 IMPRESS

Dirección del prefijo uno en la memoria: 62001 (por ejemplo)

62001...1

62001+1 Dirección del gráfico que queremos imprimir.

62001+3 Dimensiones del gráfico que vamos a imprimir; primero se almacena el ancho del sprite en caracteres y después el alto del sprite en scans.

62001+5 Posición del gráfico en la pantalla. Se pkea primero la posición en la coordenada X en pixels y después la de la coordenada Y en pixels. Recuerda que el origen del sistema de coordenadas está situado en la esquina superior izquierda.

2 FASTER

62001...2

Los datos son básicamente idénticos. Las únicas diferencias es que la dirección de los datos de 62001+3 no apuntará directamente al gráfico, sino a una tabla. En esta tabla deberán estar las direcciones de inicio de cada gráfico desplazando, según el número de pixels que se haya separado la posición X de la columna situada a su izquierda. Por ejemplo:

X=27. Excede en tres pixels de la columna de su izquierda ($8 \times 3 = 24$).

TABLA

Exceso de pixel	Dirección
1	60000
2	60100
3	60200
4	60300, etc

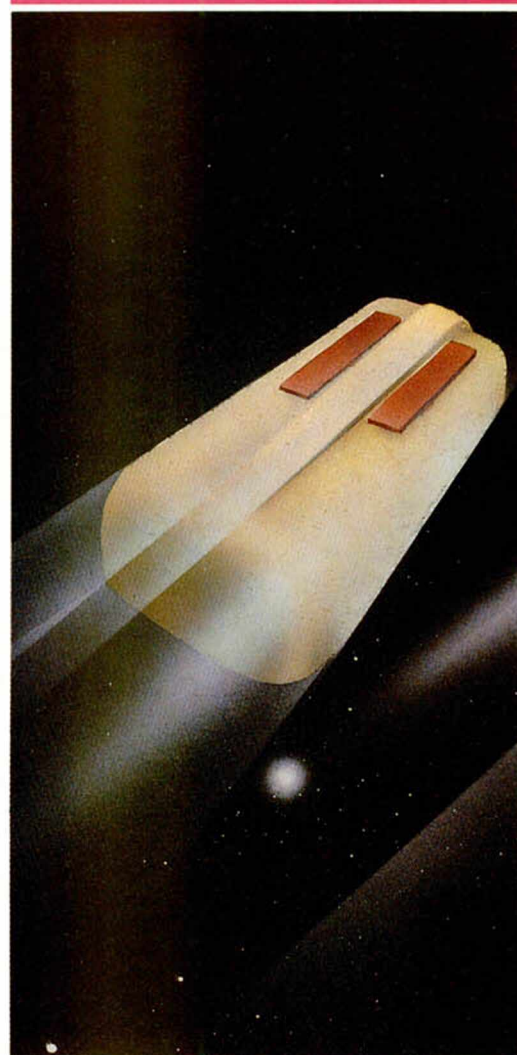
La tabla podrías ubicarla en la dirección 50000, por ejemplo, y ese es el dato que debes de introducir en dirección. Esta tabla estaría

LISTADO ENSAMBLADOR ACTION

60	ORG 63450	700	CHECK LD A,%01111111	1340	CP 4	1950	CP 255	2590	RETI
70	LD A,2	710	LD E,A	1350	JP NZ,PREG	1960	JP NZ,CONT1	2600	LNTIMP LD A,(BUFFER)
80	CALL 5633	720	CHECK1 LD A,E	1360	RET	1970	CALL DEREK	2610	INC A
90	LD BC,53	730	RRCA	1370	ESPERA XOR A	1980	CONT1 INC HL	2620	CALL PUTTER
100	LD DE,TEXT	740	LD E,A	1380	IN A,(254)	1990	LD A,(HL)	2630	PUSH HL
110	CALL #203C	750	IN A,(#FE)	1390	CPL	2000	IN A,(#FE)	2640	PUSH IX
120	AUNNO LD A,#F7	760	AND %00011111	1400	AND #1F	2010	AND 31	2650	POP DE
130	IN A,(#FE)	770	CP %00011111	1410	JP NZ,ESPERA	2020	CPL	2660	POP IX
140	BIT 0,A	780	JP NZ,INTEC	1420	RET	2030	INC HL	2670	LD (IX-7),1
150	JP Z,KEMPS	790	JP CHECK1	1430	SONY LD B,7	2040	OR (HL)	2680	LD HL,ANTDIR
160	BIT 1,A	800	INTEC LD D,A	1440	LD A,(23624)	2050	CP 255	2690	LD A,(HL)
170	JP Z,DEFIN	810	LD A,E	1450	SRL A	2060	JP NZ,ENVERT	2700	INC HL
180	JP AUNNO	820	IN A,(#FE)	1460	SRL A	2070	CALL IZQUI	2710	LD H,(HL)
190	KEMPS LD HL,SALTO	830	AND %00011111	1470	SRL A	2080	ENVERT INC HL	2720	LD L,A
200	LD DE,JOYST	840	CP %00011111	1480	SONY2 OUT (254),A	2090	LD A,(HL)	2730	LD (IX+1),L
210	LD (HL),E	850	JP Z,PREG	1490	XOR %00010000	2100	IN A,(#FE)	2740	LD (IX+2),H
220	INC HL	860	LD D,A	1500	HALT	2110	AND 31	2750	LD (IX-4),C
230	LD (HL),D	870	LD C,2	1510	DJNZ SONY2	2120	CPL	2760	LD (IX-3),B
240	RET	880	LD B,4	1520	RET	2130	INC HL	2770	LD (IX),1
250	DEFIN LD BC,00	890	DOBLE BIT 4,A	1530	TEXT	2140	OR (HL)	2780	LD (IX+3),C
260	LD DE,TEXT2	900	JP Z,MASC	1540	DEFB 22,10,5,16,2,1	2150	CP 255	2790	LD (IX+4),B
270	CALL #203C	910	SLA A	1550	DEFB 17,0	2160	JP NZ,CONT2	2800	LD (IX+5),E
280	LD HL,SALTO	920	DJNZ DOBLE	1560	DEFM *1. JOYSTICK K	2170	CALL ABAJO	2810	LD (IX+6),D
290	LD DE,KEYBO	930	JP ONMASC	1570	DEFB 22,12,5,16,6	2180	JP CAMBIA	2820	CALL TABLER
300	LD (HL),E	940	MASC DEC C	1580	DEFM *2. DEFINE CON	2190	CONT2 INC HL	2830	LD (IX-6),L
310	INC HL	950	JP Z,CHECK	1590	TEXT2 DEFB 22,10,5,16,7	2200	LD A,(HL)	2840	LD (IX-5),H
320	LD (HL),D	960	SLA A	1600	DEFM *DERECHA "	2210	IN A,(#FE)	2850	LD HL,NUEPOS+1
330	LD HL,TABLA	970	DJNZ DOBLE	1610	DEFB 22,11,5,16,6	2220	AND 31	2860	LD D,(HL)
340	LD B,8	980	ONMASC LD B,0	1620	DEFM *IZQUIERDA "	2230	CPL	2870	DEC HL
350	XOR A	990	LD C,L	1630	DEFB 22,12,5,16,5	2240	INC HL	2880	LD E,(HL)
360	LIMPIA LD (HL),A	1000	LD HL,TABLA	1640	DEFM *ABAJO "	2250	OR (HL)	2890	DEC HL
370	INC HL	1010	COMPAR LD A,B	1650	DEFB 22,13,5,16,2	2260	CP 255	2900	LD (HL),D
380	DJNZ LIMPIA	1020	CP 4	1660	DEFM *ARRIBA "	2270	JP NZ,CAMBIA	2910	DEC HL
390	LD L,8	1030	JP Z,TODOS	1670	ZCONTR DEFB 195	2280	CALL ARRIBA	2920	LD (HL),E
400	CALL ESPERA	1040	LD A,E	1680	DEFM KEYBO	2290	CAMBIA LD A,(SWITCH)	2930	LD (IX-2),E
410	PREG	1050	CP (HL)	1690	KEYBO DI	2300	CP 1	2940	LD (IX-1),D
420	PUSH HL	1060	JP Z,POSIB	1700	LD IX,DIMEN	2310	JP Z,LNTIMP	2950	LD HL,BUFFER
430	LD A,L	1070	INC HL	1710	LD C,(IX)	2320	LD HL,NUEPOS	2960	INC (HL)
440	ADD A,9	1080	INC HL	1720	LD B,(IX+1)	2330	INC HL	2970	INC (HL)
450	LD HL,22528+5	1090	INC B	1730	LD (IX+2),0	2340	LD D,(HL)	2980	LD A,(ATR)
460	LD C,A	1100	JP COMPAR	1740	LD (IX+3),0	2350	DEC HL	2990	AND A
470	LD B,0	1110	POSIB LD A,D	1750	LD IX,NUEPOS	2360	LD E,(HL)	3000	JP NZ,TOPAIN
480	LD A,32	1120	INC HL	1760	LD L,(IX)	2370	DEC HL	3010	ADIOS EI
490	SUMA ADD HL,BC	1130	CP (HL)	1770	LD H,(IX+1)	2380	LD (HL),D	3020	RETI
500	DEC A	1140	JP NZ,NOCHEC	1780	PUSH HL	2390	DEC HL	3030	DEREK LD A,C
510	JP NZ,SUMA	1150	LD L,C	1790	POP IX	2400	LD (HL),E	3040	SLA A
520	LD B,10	1160	JP CHECK	1800	LD HL,ANPOS	2410	LD A,(BUFFER)	3050	SLA A
530	NOFLSH LD A,(HL)	1170	NOCHEC INC B	1810	LD E,(HL)	2420	CALL PUTTER	3060	SLA A
540	AND %01111111	1180	INC HL	1820	INC HL	2430	PUSH HL	3070	ADD A,2
550	LD (HL),A	1190	JP COMPAR	1830	LD D,(HL)	2440	POP IX	3080	ADD A,E
560	INC HL	1200	TODOS LD HL,TABLA	1840	INC HL	2450	LD (IX),2	3090	RET C
570	DJNZ NOFLSH	1210	LD A,C	1850	LD (HL),E	2460	CALL TABLER	3100	PUSH HL
580	LD A,22	1220	SLA A	1860	INC HL	2470	LD (IX+1),L	3110	LD HL,NUEPOS
590	ADD A,L	1230	ADD A,L	1870	LD (HL),D	2480	LD (IX+2),H	3120	LD A,E
600	JP NC,NOLLEV	1240	JP NC,NOC!	1880	LD HL,TABLA	2490	LD (IX+3),C	3130	ADD A,2
610	INC H	1250	INC H	1890	LD A,(HL)	2500	LD (IX+4),B	3140	LD E,A
620	NOLLEV LD L,A	1260	NOC! LD L,A	1900	IN A,(#FE)	2510	LD (IX+5),E	3150	LD (HL),E
630	LD B,10	1270	LD (HL),E	1910	INC HL	2520	LD (IX+6),D	3160	LD HL,HORIZ
640	SIFLSH LD A,(HL)	1280	INC HL	1920	AND 31	2530	LD HL,BUFFER	3170	LD (HL),2
650	OR 128	1290	LD (HL),D	1930	CPL	2540	INC (HL)	3180	POP HL
660	LD (HL),A	1300	CALL SONY	1940	OR (HL)	2550	LD A,(ATR)	3190	RET
670	INC HL	1310	LD L,C			2560	AND A	3200	I2QUI LD A,E
680	DJNZ SIFLSH	1320	INC L			2570	JP NZ,TOPAIN	3210	SUB 2
690	POP HL	1330	LD A,L			2580	EI	3220	RET C

3230	PUSH HL	3870	INC HL	4510	LD (IX+11),D	5150	CALL FASTER	5790	AND %00000111
3240	LD HL,NUEPOS	3880	LD D,(HL)	4520	LD (IX+12),A	5160	POP BC	5800	AND A
3250	LD E,A	3890	LD H,D	4530	LD HL,BUFFER	5170	POP IX	5810	JP 2,EXACT
3260	LD (HL),E	3900	LD L,E	4540	INC (HL)	5180	JP FOLLOW	5820	INC C
3270	LD HL,HORIZ	3910	LD DE,ANTDIR	4550	JP ADIOS	5190	ALNT PUSH IX	5830	EXACT CALL DIRPAD
3280	LD (HL),8	3920	LD A,L	4560	ZLAUNC LD IX,#DFD	5200	PUSH BC	5840	FAST PUSH BC
3290	POP HL	3930	LD (DE),A	4570	LD (IX),195	5210	LD C,(IX+1)	5850	LD B,8
3300	RET	3940	LD A,H	4580	LD HL,INTPR	5220	LD B,(IX+2)	5860	PUSH DE
3310	ABAJ0 LD A,2	3950	INC DE	4590	LD (IX+1),L	5230	LD E,(IX+3)	5870	LDIR
3320	ADD A,8	3960	LD (DE),A	4600	LD (IX+2),H	5240	LD D,(IX+4)	5880	POP DE
3330	ADD A,D	3970	POP DE	4610	LD HL,#FE00	5250	LD L,(IX+5)	5890	POP BC
3340	CP 192	3980	RET	4620	LD BC,#00FD	5260	LD H,(IX+6)	5900	LD A,D
3350	RET NC	3990	PUTTER PUSH DE	4630	ESCRIB LD (HL),C	5270	CALL IMPRESS	5910	AND %00000111
3360	PUSH HL	4000	PUSH BC	4640	INC HL	5280	POP BC	5920	CP 7
3370	LD HL,NUEPOS+1	4010	LD B,8	4650	DJNZ ESCRIB	5290	POP IX	5930	JP 2,TOM1
3380	SUB B	4020	LD C,A	4660	LD (HL),C	5300	JP FOLLOW	5940	INC D
3390	LD D,A	4030	LD D,8	4670	LD A,#FE	5310	ABORRA PUSH BC	5950	JP FAST1
3400	LD (HL),D	4040	LD E,7	4680	LD I,A	5320	LD C,(IX+1)	5960	TOM1 LD A,E
3410	LD HL,VERTI	4050	CALL MULTI	4690	IM 2	5330	LD B,(IX+2)	5970	ADD A,32
3420	LD (HL),4	4060	LD DE,BUFFER+1	4700	RET	5340	LD E,(IX+3)	5980	JP C,TOM2
3430	POP HL	4070	ADD HL,DE	4710	INTPR PUSH AF	5350	LD D,(IX+4)	5990	LD E,A
3440	RET	4080	POP BC	4720	PUSH BC	5360	CALL BORRA	6000	LD A,D
3450	ARRIBA LD A,D	4090	POP DE	4730	PUSH DE	5370	POP BC	6010	SUB 7
3460	SUB 2	4100	RET	4740	PUSH IX	5380	JP FOLLOW	6020	LD D,A
3470	RET C	4110	JOYST DI	4750	PUSH HL	5390	HLUEGO XOR A	6030	JP FAST1
3480	PUSH HL	4120	LD IX,DIMEN	4760	LD C,8	5400	LD (BUFFER),A	6040	TOM2 LD E,A
3490	LD D,A	4130	LD C,(IX)	4770	LD IX,BUFFER+1	5410	RST #38	6050	INC D
3500	LD HL,NUEPOS+1	4140	LD B,(IX+1)	4780	QUEDA? LD A,(BUFFER)	5420	POP HL	6060	FAST1 DJNZ FAST
3510	LD (HL),D	4150	LD (IX+2),8	4790	CP C	5430	POP IX	6070	RET
3520	LD HL,VERTI	4160	LD (IX+3),8	4800	JP 2,HLUEGO	5440	POP DE	6080	BORRA
3530	LD (HL),1	4170	LD IX,NUEPOS	4810	LD A,(IX)	5450	POP BC	6090	LD A,E
3540	POP HL	4180	LD L,(IX)	4820	CP 2	5460	POP AF	6100	SRL E
3550	RET	4190	LD H,(IX+1)	4830	JP 2,AFAS	5470	EI	6110	SRL E
3560	MULTI LD HL,8	4200	PUSH HL	4840	CP 1	5480	RETI	6120	SRL E
3570	LD A,16	4210	POP IX	4850	JP 2,ALNT	5490	GRAFIC EQU 65512	6130	AND A
3580	TRIU BIT 8,E	4220	LD HL,ANPOS	4860	CP 3	5500	SWITCH EQU 65500	6140	JP 2,BORRA1
3590	JR 2,TOPI	4230	LD E,(HL)	4870	JP 2,ABORRA	5510	ANPOS EQU 65501	6150	INC C
3600	ADD HL,BC	4240	INC HL	4880	PUSH BC	5520	NUEPOS EQU 65503	6160	BORRA1 CALL DIRPAD
3610	TOPI CCF	4250	LD D,(HL)	4890	LD C,(IX+1)	5530	BUFFER EQU 62000	6170	BORRA2 PUSH BC
3620	SRL D	4260	INC HL	4900	LD B,(IX+2)	5540	SALTO EQU 62001	6180	XOR A
3630	RR E	4270	LD (HL),E	4910	LD E,(IX+3)	5550	ATR EQU 65505	6190	PUSH DE
3640	SLA C	4280	INC HL	4920	LD D,(IX+4)	5560	DIMEN EQU 65506	6200	MILAN LD (DE),A
3650	RL B	4290	LD (HL),D	4930	LD A,(IX+5)	5570	HORIZ EQU 65508	6210	INC E
3660	DEC A	4300	IN A,(223)	4940	CALL PAINT	5580	VERTI EQU 65509	6220	DEC C
3670	JP NZ,TRIU	4310	BIT 8,A	4950	POP BC	5590	ANTDIR EQU 65510	6230	JP NZ,MILAN
3680	RET	4320	JP 2,JOYST1	4960	FOLLOW PUSH IX	5600	TABLA EQU 65515	6240	POP DE
3690	TABLER LD HL,HORIZ	4330	CALL DERE	4970	POP HL	5610	FASTER LD A,E	6250	POP BC
3700	LD A,(HL)	4340	JOYST1 IN A,(223)	4980	LD A,7	5620	AND %00000111	6260	LD A,D
3710	INC HL	4350	BIT 1,A	4990	ADD A,L	5630	SLA A	6270	AND %00000111
3720	ADD A,(HL)	4360	JP 2,JOYST2	5000	JP NC,PENSE	5640	ADD A,L	6280	CP 7
3730	SLA A	4370	CALL IZQUI	5010	INC H	5650	JP NC,MAZ	6290	JP 2,TOM3
3740	PUSH AF	4380	JOYST2 IN A,(223)	5020	PENSE LD L,A	5660	INC H	6300	INC D
3750	LD HL,GRAFIC	4390	BIT 2,A	5030	PUSH HL	5670	MAZ LD L,A	6310	JP BORRA3
3760	LD A,(HL)	4400	JP 2,JOYST3	5040	POP IX	5680	PUSH DE	6320	TOM3 LD A,E
3770	INC HL	4410	CALL ABAJO	5050	INC C	5690	LD E,(HL)	6330	ADD A,32
3780	LD H,(HL)	4420	JOYST3 IN A,(223)	5060	JP QUEDA?	5700	INC HL	6340	JP C,TOM4
3790	LD L,A	4430	BIT 3,A	5070	AFAS	5710	LD D,(HL)	6350	LD E,A
3800	POP AF	4440	JP 2,CAMBIA	5080	PUSH BC	5720	PUSH DE	6360	LD A,D
3810	ADD A,L	4450	CALL ARRIBA	5090	LD L,(IX+1)	5730	POP HL	6370	SUB 7
3820	JP NC,TABL1	4460	JP CAMBIA	5100	LD H,(IX+2)	5740	POP DE	6380	LD D,A
3830	INC H	4470	TOPAIN LD (IX+7),4	5110	LD C,(IX+3)	5750	LD A,E	6390	JP BORRA3
3840	TABL1 LD L,A	4480	LD (IX+8),C	5120	LD B,(IX+4)	5760	SRL E	6400	TOM4 LD E,A
3850	PUSH DE	4490	LD (IX+9),B	5130	LD E,(IX+5)	5770	SRL E	6410	INC D
3860	LD E,(HL)	4500	LD (IX+10),E	5140	LD D,(IX+6)	5780	SRL E	6420	BORRA3 DJNZ BORRA2

6430	RET	7880	RET	7730	TOM21	LD	A,L
6440	DIRPAN	PUSH	DE	7740	SUB	E	
6450	PUSH	HL		7750	ADD	A,32	
6460	POP	DE		7760	JP	C,TOM22	
6470	CALL	DIRPAD		7770	LD	L,A	
6480	PUSH	DE		7780	LD	A,H	
6490	POP	HL		7790	SUB	7	
6500	POP	DE		7800	LD	H,A	
6510	RET			7810	JP	IMPR3	
6520	PAINT	PUSH	AF	7820	TOM22	LD	L,A
6530	SRL	B		7830	INC	H	
6540	SRL	B		7840	JP	IMPR3	
6550	SRL	B		7850	IMPR2	SRL	L
6560	LD	A,E		7860	SRL	L	
6570	AND	%00000111		7870	SRL	L	
6580	AND	A		7880	CALL	DIRPAN	
6590	JP	Z,XEXAC		7890	PUSH	IX	
6600	INC	C		7900	PUSH	DE	
6610	XEXAC	LD	A,D	7910	POP	BC	
6620	AND	%00000111		7920	POP	DE	
6630	AND	A		7930	LD	A,C	
6640	JP	Z,YEXAC		7940	IMPR7	PUSH	BC
6650	INC	B		7950	LD	B,A	
6660	YEXAC	SRL	D	7960			
6670	SRL	D		7970	PUSH	AF	
6680	SRL	D		7980	LDIR	LD	A,(DE)
6690	SRL	E		7990	XOR	(HL)	
6700	SRL	E		8000	LD	(HL),A	
6710	SRL	E		8010	INC	DE	
6720	PUSH	DE		8020	INC	L	
6730	POP	HL		8030	DJNZ	LDIR	
6740	CALL	DIRAT		8040	POP	AF	
6750	POP	AF		8050	LD	B,A	
6760	LD	H,C		8060	DEC	B	
6770	PAIN1	LD	C,H	8070	DEC	HL	
6780	PUSH	DE		8080	PUSH	AF	
6790	PAIN2	LD	(DE),A	8090	LD	A,H	
6800	INC	DE		8100	AND	%00000111	
6810	DEC	C		8110	CP	7	
6820	JP	NZ,PAIN2		8120	JP	Z,TTM1	
6830	POP	DE		8130	LD	A,L	
6840	LD	L,A		8140	SUB	B	
6850	LD	A,E		8150	LD	L,A	
6860	ADD	A,32		8160	INC	H	
6870	JP	NC,SINCA		8170	JP	IMPR6	
6880	INC	D		8180	TTM1	LD	A,L
6890				8190	SUB	B	
6900	SINCA	LD	E,A	8200	ADD	A,32	
6910	LD	A,L		8210	JP	C,TTM2	
6920	DJNZ	PAIN1		8220	LD	L,A	
6930				8230	LD	A,H	
6940	RET			8240	SUB	7	
6950	DIRAT	LD	A,H	8250	LD	H,A	
6960	SRA	A		8260	JP	IMPR6	
6970	SRA	A		8270	TTM2	LD	L,A
6980	SRA	A		8280	INC	H	
6990	ADD	A,#58		8290	IMPR6	POP	AF
7000	LD	D,A		8300	POP	BC	
7010	LD	A,H		8310	DJNZ	IMPR7	
7020	AND	7		8320	RET		
7030	RRCA			8330	ZQUIT	LD	A,#3F
7040	RRCA			8340	LD	I,A	
7050	RRCA			8350	IM	1	
7060	ADD	A,L		8360	RET		
7070	LD	E,A		8370	ZFLAG		
7080	RET						
7090	DIRPAD	LD	A,D				
7100	SLA	A					
7110	SLA	A					
7120	AND	%11100000					
7130	OR	E					
7140	LD	E,A					
7150	LD	A,D					
7160	AND	%11000000					
7170	SRL	A					
7180	SRL	A					
7190	SRL	A					
7200	PUSH	AF					
7210	LD	A,D					
7220	AND	%00000111					
7230	LD	D,A					
7240	POP	AF					
7250	OR	D					
7260	SET	6,A					
7270	LD	D,A					
7280	RET						
7290	IMPRES	PUSH	BC				
7300	POP	IX					
7310	LD	A,L					
7320	AND	%00000111					
7330	LD	B,A					
7340	AND	A					
7350	JP	Z,IMPR2					
7360	SRL	L					
7370	SRL	L					
7380	SRL	L					
7390	CALL	DIRPAN					
7400	IMPR3	PUSH	DE				
7410	LD	D,0					
7420	LD	C,0					
7430	IMPR5	LD	A,(IX)				
7440	PUSH	BC					
7450	IMPR4	SRL	A				
7460	RR	C					
7470	DJNZ	IMPR4					
7480	ADD	A,D					
7490	XOR	(HL)					
7500	LD	(HL),A					
7510	INC	L					
7520	LD	D,C					
7530	LD	C,B					
7540	POP	BC					
7550	INC	IX					
7560	DEC	E					
7570	JP	NZ,IMPR5					
7580	LD	A,D					
7590	XOR	(HL)					
7600	LD	(HL),A					
7610	POP	DE					
7620	DEC	D					
7630	RET	Z					
7640	LD	A,H					
7650	AND	%00000111					
7660	CP	7					
7670	JP	Z,TOM21					
7680	LD	A,L					
7690	SUB	E					
7700	LD	L,A					
7710	INC	H					
7720	JP	IMPR3					



DEMO 1

```

5 BORDER 0: INK 7: PAPER 0: C
LEAR 24999: FOR a=65500 TO 65520
: POKE a,0: NEXT a
10 POKE 65506,3: POKE 65507,16
: POKE 65505,4: POKE 65510,56: P
OKE 65511,199: POKE 65512,80: PO
KE 65513,195: LOAD ""CODE 50000:
LOAD ""CODE 50500: LOAD ""CODE
51000: RANDOMIZE USR 63450: RAND
OMIZE USR 64401
20 RANDOMIZE USR 63849: PAUSE
1: GO TO 20
6000 FOR A=50000 TO 50030 STEP 2
: POKE A,68: POKE (A+1),197: NEX
T A: GO TO 20
6010 GO TO 10

```

LISTADO 1.1 DUMP: 50.000. N.º BYTES: 30

LÍNEA	DATOS	CONTROL
1	38C738C778C758C7B8C7	1499
2	38C798C738C7F8C718C8	1532
3	38C738C7D8C738C738C7	1435

En la línea 2470 comienzan a pokear los valores en IX según el criterio que ya antes te he explicado. Incrementa el contenido de buffer, para indicar que se ha añadido un gráfico, y observa si ATR es distinto de 0, en cuyo caso debería saltar a TOPAIN.

TOPAIN deja los datos en el buffer para que se pinte el sprite después de imprimirlo. Finalmente, incrementa de nuevo el contenido de buffer, regresa y vuelve al Basic reponiendo las interrupciones.

En caso de que se hubiera escogido IMPRES, el procedimiento sería muy similar. La diferencia sería que habría que imprimir dos gráficos, uno para borrar el sprite anterior y otro para imprimir el nuevo. La dirección del sprite anterior la tomaría de ANTDIR, y la posición del registro IX, en el cual la habíamos dejado desde la línea 1790. Soltaría a TOPAIN si fuera necesario, y para finalizar vuelve al Basic. Con esto hemos acabado la segunda parte, y llegamos al impresor por interrupciones.

IMPRESOR POR INTERRUPCIONES

La rutina ZLAUNC, en la línea 4560, establece las interrupciones en modo 2. ZLAUNC parece excesivamente complicada, ya que las interrupciones se establecen con tres órdenes, pero no es así.

Una interrupción se fija al cargar en el registro I un valor que será el byte de mayor peso de una dirección. El byte de menor peso se toma del valor del bus de datos, que en teoría debería valer 255.

Sin embargo, el interface Kempson puede hacer que este valor cambie, y no sabemos la dirección resultante. Para solucionarlo, la rutina llena desde FE00h hasta FEFFh toda la memoria con el dato FDh. Así caiga donde caiga la interrupción siempre tomará como valor de salto FDFDh.

En esa dirección tenemos la memoria justa para hacer un jPRINTPR. En INTPR primero se salvan todos los re-

gistros, y después se observa si hay algún gráfico que imprimir. Si no hay ninguno, se realiza un rst 38h para actualizar el teclado con la rutina de la ROM. Si estás usando esta rutina desde Código Máquina, puedes quitar esa orden. Para acabar recupera todos los registros y vuelve al Basic.

Si hay más gráficos para imprimir, el ordenador con IX apuntando a la dirección del buffer toma el prefijo, y en base a él salta a las rutinas de preparación de datos, que son AFAST, ALNT, ABORRA y una última que los manda a PAINT. A la vuelta de cada una de estas rutinas, se añade IX 7 y si quedan más gráficos se repite la operación. Finalmente se pone a 0 buffer y se vuelve al Basic.

Cada una de estas rutinas de preparación (AFAST, ALNT, etc.), hacen lo mismo. Toman los datos del buffer y llaman a cada una de las rutinas de impresión, de borrado o de coloreado.

Empezaré explicando FASTER. En ella se toma el exceso de bits de desplazamiento con respecto a la columna, se multiplica por 2 y se busca en la tabla. Después se carga en HL la dirección de comienzo del sprite, y llama a DIRPAD, rutina que calcula la dirección de la memoria de pantalla donde se debe empezar a imprimir el sprite. Al imprimir cada scan, vuelve a tomar el ancho, calcula la dirección de la pantalla que está debajo de la anterior entre las líneas 5900 y 6050 y después decrementa el alto hasta que éste llega 0, en cuyo caso vuelve al Basic.

IMPRES es muy parecida; la diferencia está en que antes de imprimir cada byte tiene que rotarlo, y pasar parte de él a otros bytes.

Además, si el exceso de desplazamiento es 0, salta a IMPR2, en la cual lo hace más rápidamente, ya que no debe rotar nada.

BORRA empieza en la línea 6080. Comprueba si la posición está desplazada de alguna columna, y si es así incrementa el ancho en la línea 6150. Llama a DIRPAD, para calcular la dirección en el archivo de pantalla; pone a 0 los bytes de pantalla de cada scan; después recicla la dirección de pantalla y halla la dirección del scan inferior del anterior entre las líneas 6260 y 6420. Repite esto hasta que acaba con el alto de la zona a borrar y retorna.

PAINT empieza por dividir el alto por ocho y si las posiciones horizontales no corresponden a alguna columna precisamente incrementa el ancho. Llama a DIRAT, subrutina que calcula la posición en el archivo de atributos en base a dos coordenadas cargadas en HL, y devuelve el resultado en DE. A la vuelta a la rutina empieza a llenar la zona de los atributos con el valor del registro A. Después calcula la dirección en el archivo de atributos debajo de la actual posición y lo repite hasta que acaba con el alto.

Aquí finaliza la explicación del funcionamiento; ahora voy a decirte cómo puedes hacerlo.

USO DE ACTION

Para empezar, éstas son las direcciones para activar y desactivar la rutina:

63450 Origen de la rutina y dirección donde debes llamar para redefinir teclas.

63849 Dirección de inicio de la subrutina de actualización de posición con chequeo del teclado. Si vas a hacer un movimiento llevado por esta rutina deberás

LISTADO 1.2 DUMP: 51.000 . N.º BYTES: 257

LÍNEA	DATOS	CONTROL
1	0180018003C003C007E0	879
2	07E00FF00FF000000FF0	996
3	0E700F700F700F700F70	634
4	0FF00003000F003E00FE	589
5	03FC03FC05F80EF81F70	1168
6	23B07BC0E3806F002200	1026
7	1C008000000000000000	36
8	0000FEC0E2F0FAFCF2FF	1911
9	FAPFE2FCFEF0FEC00000	1923
10	00000000000000000000	36
11	2A006B00E3807BC03BB0	1054
12	1F700EF805F803FC03FC	1168
13	00FE003E000F00030FF0	589
14	0E300EF00E300FB00E30	631
15	0FF000000FF00FF007E0	996
16	07E003C003C001800180	879
17	00100038004400DE01C7	562
18	03D60DC40EF81F701FA0	1022
19	3FC03FC07F007C00F000	1001
20	C0000000000000000000	192
21	037F0F613F70FF78FF77	1182
22	3F770F7F037F00000000	454
23	00000000C000F0007C00	556
24	7F003FC03FC01FA01F70	971
25	0EF80DC403D601C700D6	1102
26	00440038001000000000	140

llamarla periódicamente. También se encarga de introducir los datos en el buffer.

64401 En esta dirección se activa el impresor por interrupciones. Una vez puesto en marcha, no es necesario que vuelvas a llamar aquí de nuevo.

65014 Dirección para desconectar el impresor por interrupciones. Hay dos posibilidades: que desees usar la rutina introduciendo directamente en el buffer datos y esperando a que se impriman o definiendo las teclas y llamando a la rutina del teclado.

Si piensas hacer lo primero, deberán introducir los datos como ya he expuesto antes. Para ello deberás tomar el contenido de buffer, multiplicarlo por 7 y añadirle al resultado 62001. Entonces tendrás la dirección donde debes colocar los datos. Los introduces, incrementas el contenido de buffer y ya está.

Te recuerdo que cuando uses IMPRES, la dirección será la del propio gráfico, y si usas FASTER la dirección será la de una tabla con las direcciones de los gráficos desplazados.

Si usas la rutina del teclado, lo primero que debes hacer es darles a las variables los contenidos correctos. Piensa que si trabajas

con IMRES entonces deberás hacer una tabla, y si usas FASTER dos tablas, aparte naturalmente de los propios gráficos. La primera tabla debe tener las direcciones de los gráficos, (o de las segundas tablas, en el caso de FASTER) y se debe de poder acceder a ella sumando VERTI y HORIZ, multiplicando el resultado por dos y añadiéndole GRAFIC. En el contenido de la posición de memoria resultante se debe de encontrar la dirección de inicio de otra tabla o del gráfico, según el tipo de impresión. Cuando es otra tabla, es señal de que se usa FASTER, y en esa tabla deben estar las direcciones de inicio de cada gráfico desplazado. Para acce-

DEMO 2p.

```

5 BORDER 0: PAPER 0: INK 7: C
LS : FOR a=65500 TO 65515: POKE
a,0: NEXT a: POKE 65500,1: POKE
65505,6: POKE 65506,2: POKE 6550
7,16: POKE 65512,80: POKE 65513,
195: LOAD ""CODE 50000: LOAD ""C
ODE 51000: RANDOMIZE USR 63450:
RANDOMIZE USR 64401
10 RANDOMIZE USR 63849: PAUSE
1: GO TO 10
6000 FOR A=50000 TO 50030 STEP 2
: POKE A,56: POKE (A+1),199: NEX
T A
6010 GO TO 10
    
```

LISTADO 2.1

DUMP: 50.000. N.º BYTES: 30

LÍNEA	DATOS	CONTROL
1	44C544C5564C554C584C5	1437
2	000074C50000A4C5B4C5	1051
3	0000000094C538C738C7	855

LISTADO 2.2

DUMP: 50.500. N.º BYTES: 129

LÍNEA	DATOS	CONTROL
1	38C7000068C70000A8C7	925
2	0000E8C7000028C80000	671
3	58C8000098C80000D8C8	1056
4	000018C9000048C90000	498
5	88C90000C8C9000008CA	948
6	000038CA000078CA0000	580
7	B8CA0000F8CA000028CB	1079
8	000068CB0000A8CB0000	678
9	E8CB000018CC000058CC	955
10	000098CC0000D8CC0000	776
11	08CD000048CD000088CD	831
12	0000C8CD0000F8CD0000	858
13	38CE000078CE00000000	588

LISTADO 2.3

DUMP: 51.000. N.º BYTES: 1921

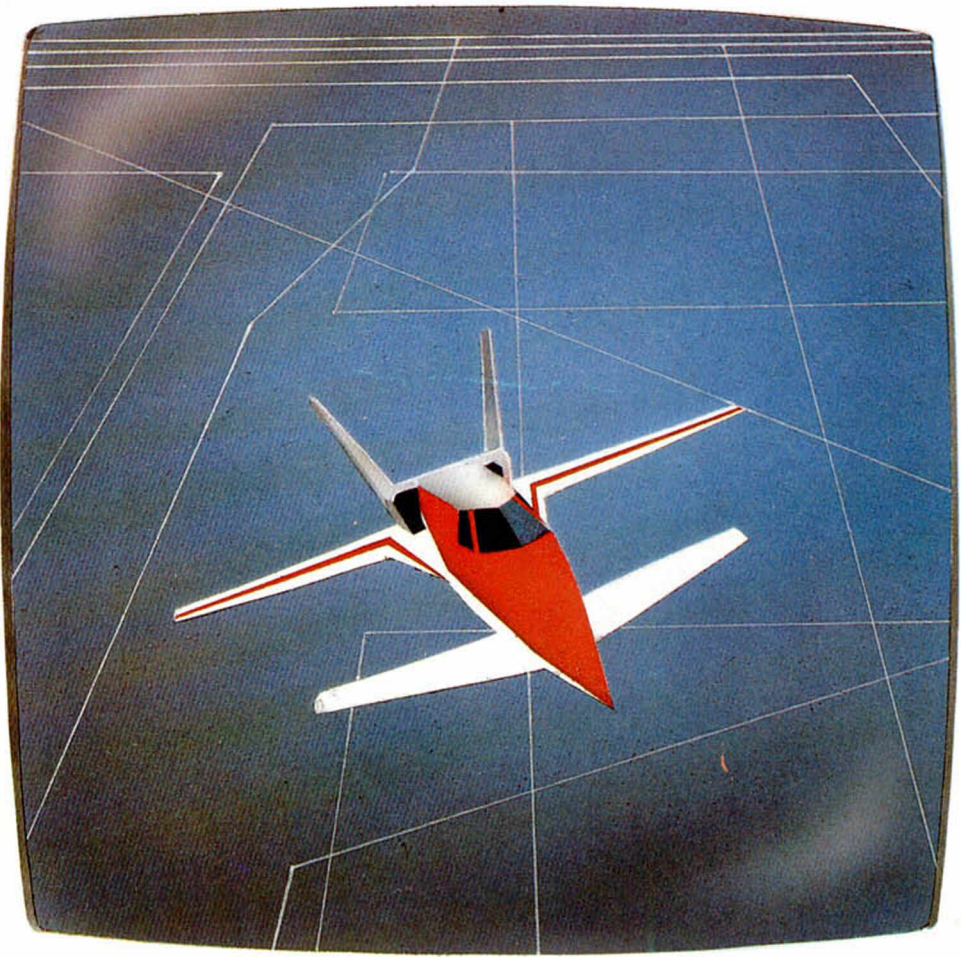
LÍNEA	DATOS	CONTROL
1	0000000000000000300000	48
2	00000003C000342C00F7E	462
3	F01F3CF80F42F0007E00	1026
4	1F00F8157E501F14F80A	815

5	00A80000000000000000	168
6	0000000000000000C0000	12
7	0000000000000000F0000D0	223
8	B00003DFBC0007CF3E00	866
9	03D0BC00001F800007C0	757
10	3E00055F940007C53E00	576
11	02802A0000000000000000	172
12	0000300000000000000003	6
13	C00000342C0000F7EF00	774
14	001F3CF8000F42F000007	877
15	E00001F00F800157E500	925
16	01F14F8000A0A8000000	747
17	0000000000000000000000	192
18	00000000F00000D08000	264
19	003DFBC0007CF3E0003D	1156
20	0BC00001F800007C03E0	803
21	0055F940007C53E00028	869
22	02A0000000000000000000	162
23	000000000000000040000	4
24	16000066700319C80EA7	647
25	241D9CC40B731030EC48	915
26	2F111037B440383B001C	522
27	1C00000000000000000000	28
28	000000000000000010000	1
29	0006000000199C0000C6	385
30	720003A9C90007673100	646
31	02DCC4000C3B12000BC4	714
32	440005ED10000ECC0000	544
33	0707000000000000000000	14
34	00004000000180000006	199
35	670000319C8000EA7240	848
36	01D9CC4000B73100030E	735
37	C48002F11100017B4400	776
38	0383300001C1C0000000	568
39	00000000000010000000	16
40	60000000199C0000C6720	589
41	003A9C9000767310002D	652
42	CC4000C3B12000BC4440	992
43	005ED10000ECC000070	843
44	7000000000000000000000	112
45	000006FDF40ED6800EFE	1127
46	000048081FFFF820004	650
47	0DB6D82221140442241B	631
48	60B0000000000000000000	285
49	0000000001BF7D0003B5	501
50	A000003BF80000120200	502
51	07FFFE000800010002DB	746
52	6C000110880008884500	474
53	035DB400000000000000	292
54	000000000000000000F	111
55	DF4000ED500000EF0000	1091
56	0044808001FFFF800200	901
57	0040016DB68000221100	535
58	02442240016DB6800000	588
59	000001BF7D00003B5A00	631
60	003BF80000012020007F	499
61	FFE000800010002DB6C0	1042
62	0091069000088440005A	591
63	DB60000000000000000000	315
64	F000033C00003D00001D6	729
65	001C0180233C4024CF18	583
66	19301C061BCC01B89C00	682
67	7038000000000000000000	168
68	003C000000CF000000F4	511
69	00000075800007006000	348
70	08CF1000933C600064C	571
71	0700186F300006EE700	726
72	001CE0E000000000000000	42
73	0000000000F00000033	66
74	000003D0000001D6000	378
75	01C018000233C400024C	544
76	F180019301C00061BCC0	1187
77	001BB9C0000703800000	542
78	0000000000000000000003	3
79	C000000CF000000F4000	523
80	0007580000700600003C	353
81	F1000933C600064C070	948
82	00186F300006EE700001	540
83	C0E0000000000000000000	416
84	0000003C0007F3E000F72	663
85	F01B1CD80F7EF0000000	892
86	1F7EF81554501F28F80A	919
87	00A8000000000000000000	168
88	0000000000F0000001FC	268
89	F80003DCBC00006C73600	918
90	03DFBC0000000000007DF	644
91	BE000555140007CA3E00	571
92	02802A0000000000000000	172
93	0000000000000000000003	3
94	000007F3E00000F72F00	675
95	01B1CD8000F7EF000000	997
96	000001F780001554500	770
97	01F28F8000A0A8000000	812
98	00000000F000001FCF80	606
99	003DCBC0006C7360003D	836
100	FBC00000000000007DFBE0	1033
101	00555140007CA3E00028	781
102	02A0000000000000000000	162
103	0F00003CC0000BC0005B	577
104	80018038023CC418F324	874
105	380C9833D86039DD801C	1017
106	0E000000000000000000	14
107	0003C00000F30000002	260
108	F000001AE00000600E00	600
109	008F3100063CC9000E03	476
110	26000CF618000E776000	549

der a ella debe tomar el exceso de pixels que nos resulta de restar la posición X de la columna de pantalla de su izquierda. El resultado es un número entre 0 y 7. Después lo multiplicas por dos y lo añades al inicio de esta segunda tabla, para tomar de esa dirección la posición de inicio del gráfico desplazado. Esto último lo realiza la propia rutina FASTER.

Es evidente que antes de mover así un sprite deberás de haber introducido estas tabla(s) en la memoria, según indique GRAFIC, que normalmente vale 50000.

Para dar color al sprite que estás moviendo dale a ATR un valor distinto de 0, con el cual coloreará los atributos del sprite.



```

131 070380000000000000000000 138
133 0000000000000000F00000003 243
134 CC0000000BC0000006B800 582
135 0018038000023CC40018F 602
136 324003800980033D8600 772
137 039DD80001C0E00000000 793
140 3C0000000F30000002F00 350
141 0001AE000000600E00006 413
142 F3100063CC9000E03260 1076
143 00CF618000E7760000070 893
144 38000000000000000000000 56
146 00002FBF50016B70007F 681
147 701012001FFFF8200004 716
148 0DB6D02442240884401B 772
149 6D680000000000000000000 213
151 0000000000BEFD800005A 556
152 DC00001FDC0004048000 607
153 07FFFE000800010005B6 712
154 DA000221100009108900 431
155 05B6DA0000000000000000 405
157 0000000000000000000002FB 253
158 F6000016B7000007F700 705
159 0101200001FFFF800200 675
160 004000DB6D00000442200 494
161 02884440000B6DB0000000 671
164 000000BEFD800005ADC0 941
165 0001FDC000404800007F 709
166 FFE000800010006DB6C0 1106
167 00A2111000091088000036 530
168 DB600000000000000000000 315
169 00000000000000000000000 32
170 18000E66001398C024E5 768
171 702339B808CED012370C 895
172 0888F4022DE800CC1C00 899
173 38380000000000000000000 112
174 00000000000000000080000 8
175 0006000000399800004E6 524
176 300009395C0008CE6E00 530
177 0233B400048DC3000222 609
178 3D00008B7A0000330700 380
179 000E0E0000000000000000 28
181 0002000000018000000E6 361
182 600001398C00024E5700 461
183 02339B80008CED000123 749
184 70C000888F400022DE80 1031
185 000CC1C00003838000000 659
187 00000000000000800000000 128
188 6000000399800004E6300 482
189 009395C00008CE6E00023 1117
190 364000048DC30002223D0 740
191 0008B7A0000330700000 514
192 E0E0000000000000000000 448
193 0000000000000000000000 0
    
```

LAS DEMOSTRACIONES

Antes de nada te diré que te parecerán muy largas de introducir, especialmente la segunda. Pues bien, para que puedas seguir viendo la demostración puedes usar un truco. En el DUMP del bloque de bytes segundo de la primera demostración introduce tan sólo 32 como duración. De esta manera, sólo tendrás que teclear cuatro líneas de texto. Después, sálvalo normalmente y mientras se ejecute el programa, haz BREAK y GO TO 6000. Así apreciarás el movimiento, aunque el gráfico no tomará direcciones distintas según éste.

Para hacerlo en la segunda rutina, deberás introdu-

DEMO 3p.

```

5 BORDER 0: PAPER 0: INK 7: C
LEAR 24999
7 LOAD ""CODE 50000: LOAD ""C
ODE 51000: LOAD ""CODE 54000: RA
NDOMIZE USR 64401
10 FOR a=49000 TO 49120: POKE
A,136: NEXT A: FOR A=49080 TO 49
120: POKE A,136: NEXT A
20 LET M=49005: LET L=49050
30 FOR A=PI TO 0 STEP -0.07: L
ET K=(176-((SIN A)*30+40)): POKE
M,K: POKE L,K: LET M=M+1: LET L
=L+1: NEXT A
35 PRINT AT 21,0; INK 4;""
40 LET C=USR 54000
    
```

LISTADO 3.1

DUMP: 50.000. N.º BYTES: 30

LÍNEA	DATOS	CONTROL
1	38C7000098C7000018C8	830
2	000098C80000F8C718C8	1023
3	38C738C7D8C738C738C7	1435



cir tan sólo 24 líneas de texto del segundo bloque de bytes, y escribir como duración 240 bytes. Sálvalo, y después de ejecutar el programa, haz BREAK y GO TO 6000.

En cuanto al tema de las demostraciones, la primera hace empleo de IMPRES para mover por la pantalla una flecha con las teclas redefinidas.

La segunda realiza el movimiento de un tanque con una ligera animación de las orugas en el movimiento horizontal.

La tercera sirve para mostrar como también se pueden hacer con esta rutina en el movimiento de personajes secundarios, en este caso con animación.

Esta demostración funciona introduciendo datos directamente en el buffer. Representa a una chica corriendo y dando saltos. Los saltos los he conseguido mediante los datos de una función seno.

La cuarta y última demostración borra y pinta espacios de pantalla al meter datos en el buffer desde el propio Basic.

Un último apunte sobre la rutina; si quieres usar el control de teclado pero con distinta velocidad en el movimiento, cambia los valores numéricos de las líneas 3070, 3130, 3210, 3310 y 3460 por otro valor.

LISTADO 3.2

DUMP: 51.000. N.º BYTES: 481

LÍNEA	DATOS	CONTROL
1	0000000000000000500010	96
2	AC000554000ABE00056C	574
3	0000F000007C00000000	364
4	00E80001E40003C40003	663
5	840007816007ED6003EC	943
6	00000000008000007C00	252
7	003C001E5E001BEE001F	480
8	F70019E7001003800002	652
9	80003C00003E00000000	550
10	0000000000A00000055	95
11	800002AA80000157C000	708
12	002D8000001E0000000F	218
13	80000000000001D0000	157
14	001C8000001C80000038	368
15	800000380000003F6000	343
16	001F600000000000001F	158
17	8000000F80000078000	406
18	00078000003800000003	269
19	A0000007A00000072000	366
20	000D7000000F30000007	195
21	80000001800000000000	257
22	00000022800000556000	343
23	00AAAA000015F000000B	602
24	6000000780000003E000	458
25	00000000000340000007	74
26	20000007200000172000	126
27	003780000017F600000B	463
28	F600000000000040000	250
29	0003E0000003E00000F3	697
30	E00000DF000000FFB800	1094
31	00CF380000801C000000	419
32	140000001E0000001F00	81
33	0000A00000055800002A	295
34	A80000157C0000002D800	531
35	0001E0000000F8000000	473
36	00000001D00000001C800	410
37	0001C800000388000003	343
38	80000003F60000001F600	624
39	00000000001F8000000	249
40	F80000007C0000007C00	496
41	00001C0000002E000000	74
42	2E0000006E0000004E00	234
43	0000E00000006F000000	345
44	37800000000000000000	183
45	00000000000000000000	0

LISTADO 3.3

DUMP: 54.000. N.º BYTES: 55

LÍNEA	DATOS	CONTROL
1	2168BF0673DD2131F2DD	1215
2	360303DD3604200E00DD	606
3	360002DD360150DD3602	689
4	C3F3DD71050C0C7EDD77	1267
5	0623DD35FF01FB7610ED	1194
6	C9000000000000000000	201

LISTADO DESENSAMBLADOR DEMO 3.3

8380	ORG 54000
8390	LD HL,49000
8400	LD B,115
8410	LD IX,BUFFER+1
8420	LD (IX+3),3
8430	LD (IX+4),32
8440	LD C,0
8450	LD (IX),2
8460	LD (IX+1),80
8470	LD (IX+2),195
8480	DEMO DJ
8490	LD (IX+5),C
8500	INC C
8510	INC C
8520	LD A,(HL)
8530	LD (IX+6),A
8540	INC HL
8550	LD (IX-1),1
8560	EI
8570	HALT
8580	DJNZ DEMO
8590	RET

DEMO 4p.

```

10 BORDER 0: PAPER 0: INK 7: C
LS : CLEAR 24999:
20 LET c=USR 64401: FOR a=0 TO
255 STEP 2: PLOT a,0: DRAW 0,17
5: NEXT a
30 FOR a=0 TO 20: LET anc=INT
(RND*31): LET alt=INT (RND*170):
LET x=INT (RND*(31-anc)): LET y
=INT (RND*(192-alt))
40 POKE 62001,3: POKE 62002,an
c: POKE 62003,alt: POKE 62004,x:
POKE 62005,y: POKE 62008,4: POK
E 62009,anc: POKE 62010,alt: POK
E 62011,x: POKE 62012,y: POKE 62
013,(INT (RND*255)): POKE 62000,
2: PAUSE 1
50 NEXT a
    
```

TU COLECCIÓN

ESPECIAL



servida a la carta

ENVÍANOS EL CUPÓN A VUELTA DE CORREO

Recorta o copia este cupón y envíalo a Hobby Press, S.A. Apartado de Correos nº 232. 28080 Alcobendas (Madrid)

Desearía recibir en mi domicilio los siguientes números, registrados en MICROHOBBY al precio de 150 pesetas cada uno.

NOMBRE _____
 FECHA DE NACIMIENTO _____
 CATEGORÍA POSTAL _____
 Para aplicar tu envío, es importante que indiques el código postal.

Haber lanzado a nombre de Hobby Press, S.A.
 Giro postal a nombre de Hobby Press, S.A. Nº _____
 Contra reembolso (pagando 125 pesetas más de gastos de envío y en
 válido sólo para España)

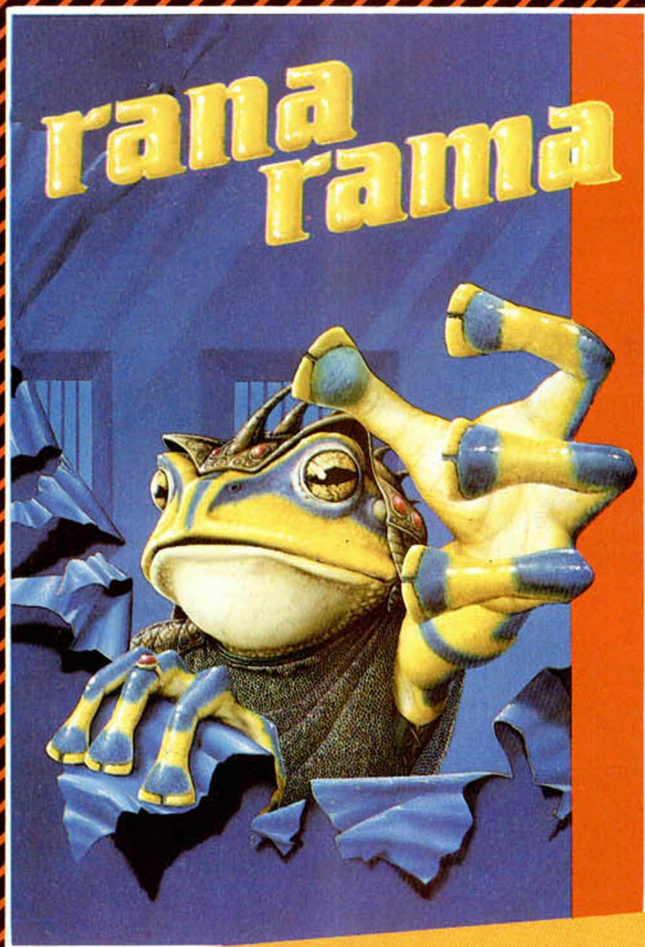
Tarjeta de crédito nº _____
 Visto Master Card American Express

Fecha de radiación de la tarjeta _____
 Nombre del titular de la tarjeta _____
 Fecha y firma _____

APPELLIDOS _____
 DOMICILIO _____
 PROVINCIA _____
 TELÉFONO _____

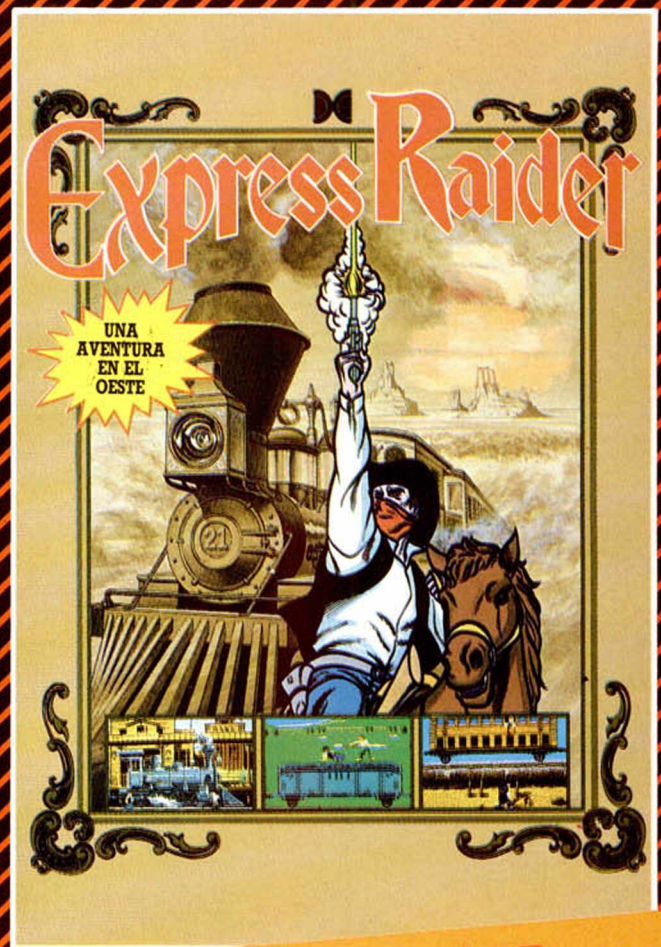
CREAR TU PROPIA RUTINA DE CARGA

¡¡NO PUEDES P



RANA-RAMA

La historia de un mago convertido en rana. Su tarea, encontrar el hechizo que le devuelva su apariencia humana. La prestigiosa revista *Micromanía* ha dicho de este juego: "Un programa de sorprendente originalidad y un índice de adicción elevadísimo." Todo lo que necesitas para pasarlo de miedo.



EXPRESS RAIDER

Como en las clásicas películas del Oeste, estarás en el centro de la acción desde el principio. Asaltos al tren, lucha sobre los vagones, cabalga sobre tu rápido caballo... EXPRESS RAIDER lo tiene todo.

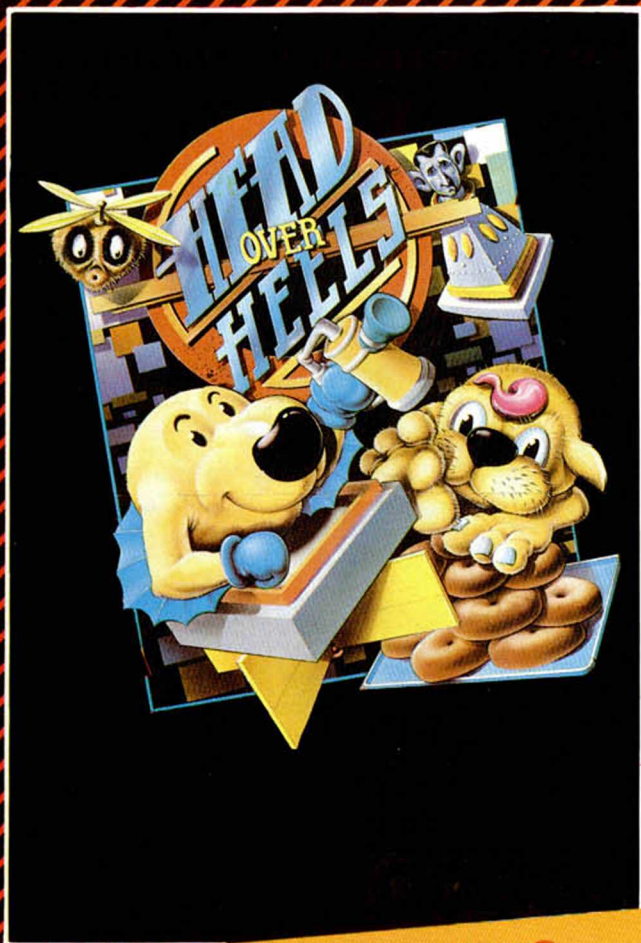
... O TE PERDERIAS LOS MEJORES JUG



DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA:

ERBE SOFTWARE. C/. NUÑEZ MORGADO, 11 - 28036 MADRID. TELÉF. (91) 314 18 04
DELEGACION BARCELONA. C/. VILADOMAT, 114 - TELÉF. (93) 253 55 60.

PERDERTELOS !!



HEAD OVER HEELS

El programa del año en Europa. Los mismos programadores que hicieron BAT-MAN han creado ahora este fabuloso juego mucho más completo aún en gráficos y movimiento. 321 pantallas francamente increíbles han hecho que "HEAD OVER HEELS" haya sorprendido a todos los críticos.



SABOTEUR II

La continuación de uno de los programas de mayor éxito de todos los tiempos. La hermana de nuestro héroe ha de salvarlo de una muerte segura. ¡¡Sólo ella y tú podéis evitarlo!!

JUEGOS DEL MOMENTO

*Ser original
te cuesta
muy poco*

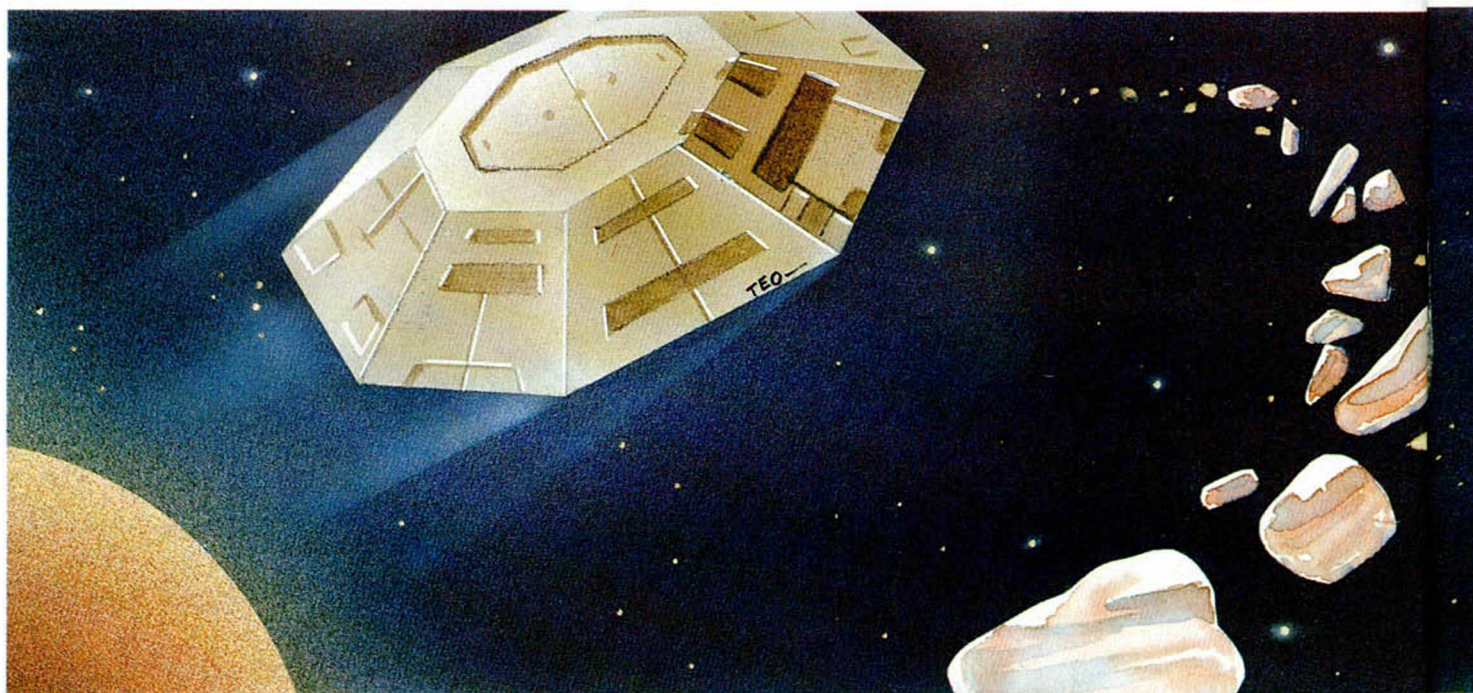
875 ptas.

* DISCO AMSTRAD 2.250 PTAS.

Toda rutina concebida para manejar gráficos y desplazarlos a lo largo y ancho de la pantalla, debe cumplir algunos requisitos básicos y de entre ellos cabe señalar como fundamental, el que sea capaz de detectar si se ha producido un

DETECCIÓN DE CHOQUE

choque entre sprites para poder obrar en consecuencia. Con este artículo se pretende dar una idea clara de las técnicas que podemos utilizar para conseguirlo.



Resulta difícil imaginar una rutina que mueva un cierto número de figuras por la pantalla, sin asignarles unas coordenadas —verticales y horizontales— a partir de las cuales podamos empezar a imprimir los sprites y también, partiendo de ellas, conocer si dos —o más— de estas figuras se han puesto en contacto.

Basándose en esta premisa, se hace evidente la necesidad de una rutina que se encargue de transformar estas coordenadas en la dirección del fichero de imagen —desde 16384d hasta

22527d— a la que correspondan. Ríos de tinta se han vertido a fin de explicar la forma en que éste está organizado en el Spectrum. Organización que, por otra parte, más de uno podría calificar de «caótica» al oír hablar de ella por vez primera. Es de suponer que muchos lectores conocerán el tema en profundidad, pero para aquellos que nunca se han aventurado a internarse en el tortuoso mar de la pantalla del Spectrum, trataremos de explicar brevemente su configuración y la manera en que podemos solventar los problemas que esto nos plantea.

El fichero de imagen, sin los atributos, ocupa 6144d bytes y se encuentra dividido en tres bloques bastante diferenciados entre sí; de tal modo que si queremos pasar de una línea de la pantalla a la inmediatamente inferior, tenemos que sumar el valor 256d a la dirección en cuestión en lugar de 32d, como cabría esperar.

La cuestión se complica todavía más si esa línea está en un tercio diferente al que nos encontramos, con lo cual en vez de 256d tenemos que sumar 2048d a la dirección en la que estemos en dicho momento.

El problema estriba, por lo tanto, en conocer nuestra situación y la cantidad que debemos de sumar, o restar, a esa dirección para ir a donde pretendemos.

La solución la tenemos en la propia ROM del ordenador, concretamente en la posición 22AAh —8874d—

y su nombre es PIXEL-AD.

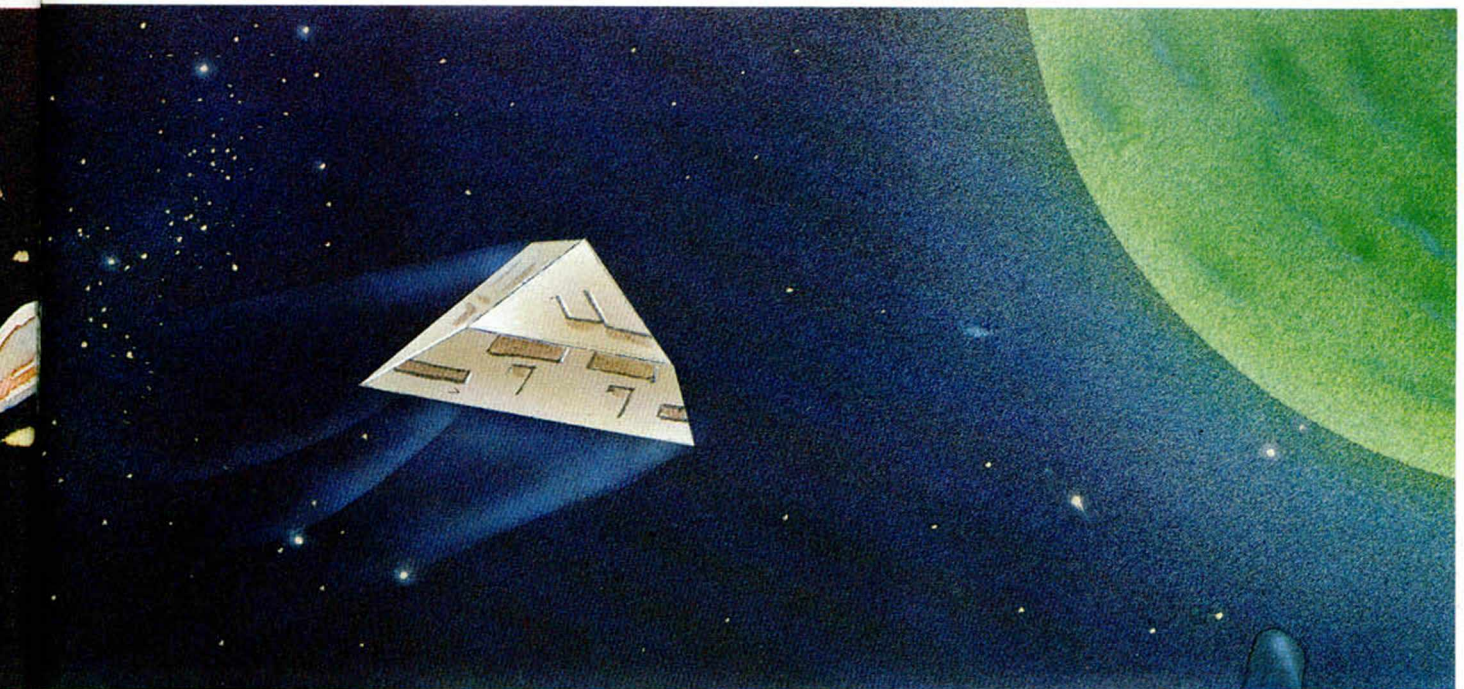
La utilización de esta rutina es sumamente simple: basta llamarla conteniendo en el registro **B** el valor de la coordenada vertical, y en **C** el de la horizontal. La dirección del fichero de imagen correspondiente a esas coordenadas, nos será devuelta en el registro doble **HL**.

Una última «pega» queda por resolver; y es que PIXEL-AD no fue concebida para operar en la parte inferior de la pantalla —donde el Spectrum presenta los mensajes de error y efectúa los INPUTs.

Para arreglar esto, basta cargar el acumulador con el valor 191d y llamar a la rutina dos direcciones más adelante con lo cual, para nosotros, PIXEL-AD va a estar situada en la posición 8876d, teniendo en cuenta que el valor mínimo —0— de la coordenada Y se en-

CHOQUES

Enrique LÓPEZ MARTÍNEZ



contrará en la última línea de la pantalla, mientras que el máximo —191— se localizará en la primera.

Una vez resuelto el problema que supone trabajar directamente con el fichero de imagen, ya podemos plantearnos la tarea de confeccionar nuestra rutina. Y vamos a intentar desarrollar una que mueva tres objetos por la pantalla: un revolver, una bala, y algo a lo que disparar. Una cruz, por ejemplo, podría ser un blanco perfecto, aunque centraremos nuestra atención en la subrutina que se encarga de comprobar si el disparo y la cruz han colisionado.

EL PROGRAMA

La rutina se encuentra profundamente comentada en el listado ensamblador, pero no estará de más recalcar algunos aspectos dignos de consideración. En primer lugar, cabe señalar que los gráficos los vamos a imprimir utilizando XOR —o OVER 1—, con lo cual a la hora de borrarlos bastará con imprimir la figura por encima de la que deseamos hacer desaparecer y el fondo quedará restablecido. Existen dos maneras de imprimir los gráficos empleando esta técnica; la primera de ellas, que es la que va a ser usada por nosotros, consiste en borrar el sprite anterior «de una sola vez» e imprimir el actual del mismo modo. Esto presenta el inconveniente de que, entre el borrado y la impresión del gráfico, transcurre un tiempo durante el cual vamos a tener grandes posibilidades de que el haz de la televisión se encuentre barriendo esa zona de la pantalla y en virtud de esto, conseguiremos obtener un desastroso efecto de parpadeo.

La otra posibilidad es borrar una línea de la figura antigua, imprimir una de la actual, continuar borrando la anterior y así sucesivamente hasta completar la impresión del gráfico. Esta técnica disimula un poco más el parpadeo —muy similar, por cierto, al que tenían la mayoría de los juegos antiguos— que la primera, pero tampoco consigue evitarlo.

Por suerte para nosotros, el microprocesador Z80 dispone de una instrucción llamada halt, que lo introduce a «no hacer nada» —salvo continuar refrescando la RAM— hasta que se produzca una petición de interrupción o un reset, la cual nos va a ayudar a lograr que nuestros gráficos no parpadeen jamás, acentuando el efecto de suavidad en los desplazamientos; aunque no movamos las figuras pixel a pixel.

Para que esta instrucción funcione, ojo, las interrupciones han de estar habilitadas —líneas 200 a 220 del programa—. En caso contrario el ordenador se col-

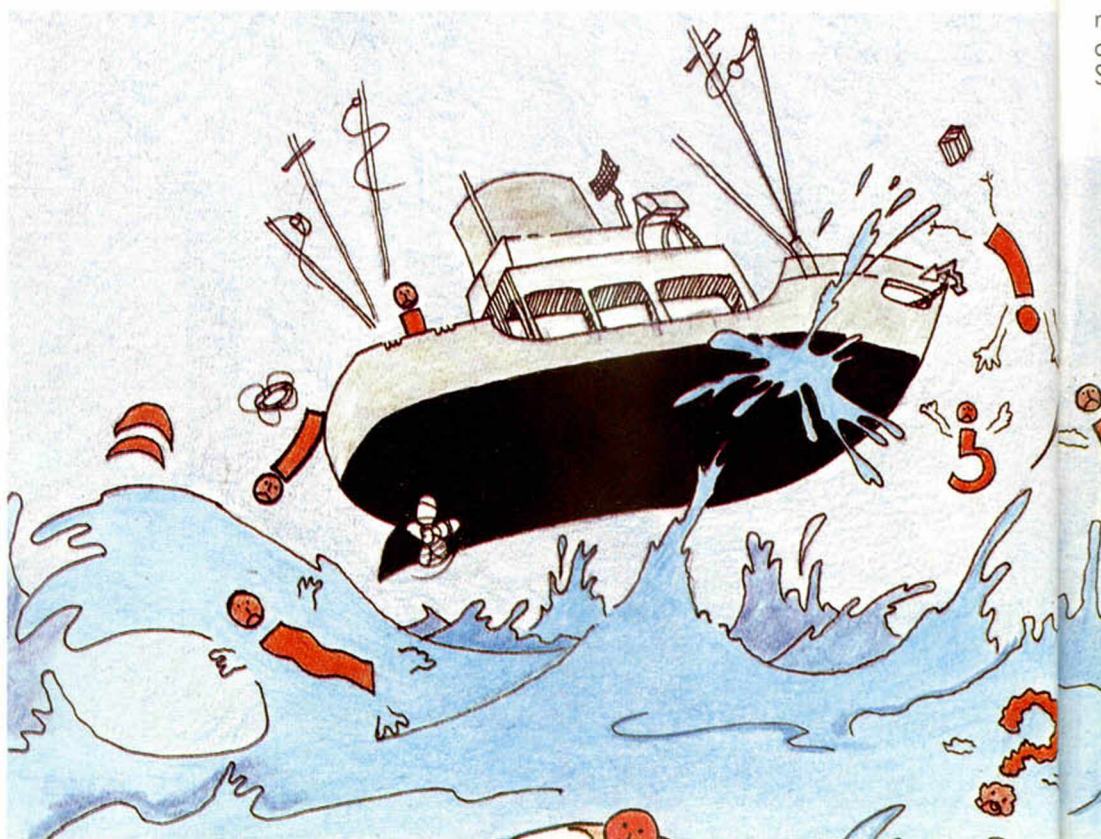
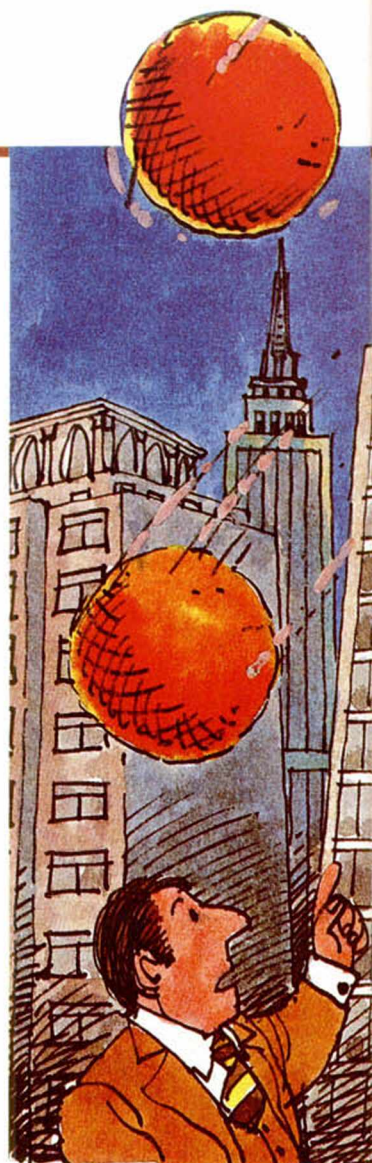
gará de forma irreversible.

Los buenos resultados con la utilización de halt dependen, en la mayoría de los casos, de la experimentación, es decir, si a pesar de todo hay un sprite que se obstina en seguir parpadeando, debemos cambiar la ubicación de halt dentro del programa hasta conseguir que deje de hacerlo.

Desgraciadamente no siempre resulta tan sencilla; en los casos más «rebeldes» suele ser eficaz la utilización de un bucle —antes o después de halt— como el propuesto a continuación y variar el valor de HL hasta lograr que el molesto parpadeo desaparezca por completo:

```

HALT: Paro del Z80
LD HL,500d: Valor inicial del bucle
LOPP DEC HL: Decrementa HL
LD A,H
OR L: ¿Es cero?
JR NZ,LOOP: Si no lo es, continúa decrementando HL
  
```



RESTO DEL PROGRAMA

Continuando con el análisis de la rutina, vamos a intentar explicar con brevedad su funcionamiento. Comienza haciendo una llamada a las subrutinas REVOL y CRUZ —líneas 450 a 650 y 660 a 860 respectivamente— cuya misión es, como su nombre indica, imprimir el revolver y la cruz, partiendo de las coordenadas vertical y horizontal contenidas en el registro doble BC. Esta operación resulta imprescindible puesto que si no lo hiciésemos así, al estar empleando XOR, observaríamos cómo extrañas cosas empezarían a ocurrir en la pantalla o, en otras palabras, la primera impresión de ambos gráficos permanecería sin borrarse.

A continuación, nos encontramos con la etiqueta PRINC, estamos ante el bucle principal del programa —líneas 40 a 250— el cual se encarga de chequear el teclado y llamar a las subrutinas ARRIBA —líneas 260 a 350 ABAJO —líneas 360 a 440— y disparo (DISP) —líneas 1310 a 1400— o salir del programa si SYMBOL SHIFT está pulsada. El retor-

no al Basic lo haremos a través de la subrutina CHOQUE; ésta tiene como objeto restablecer ciertos valores del programa, que más adelante veremos, y habilitar las interrupciones para poder salir al Basic en caso de que se confirme una colisión, por lo cual podemos perfectamente servirnos de ella para este fin.

Seguidamente, hace una llamada a las subrutinas MCRUZ —líneas 1590 a 1870— y MDISP —líneas

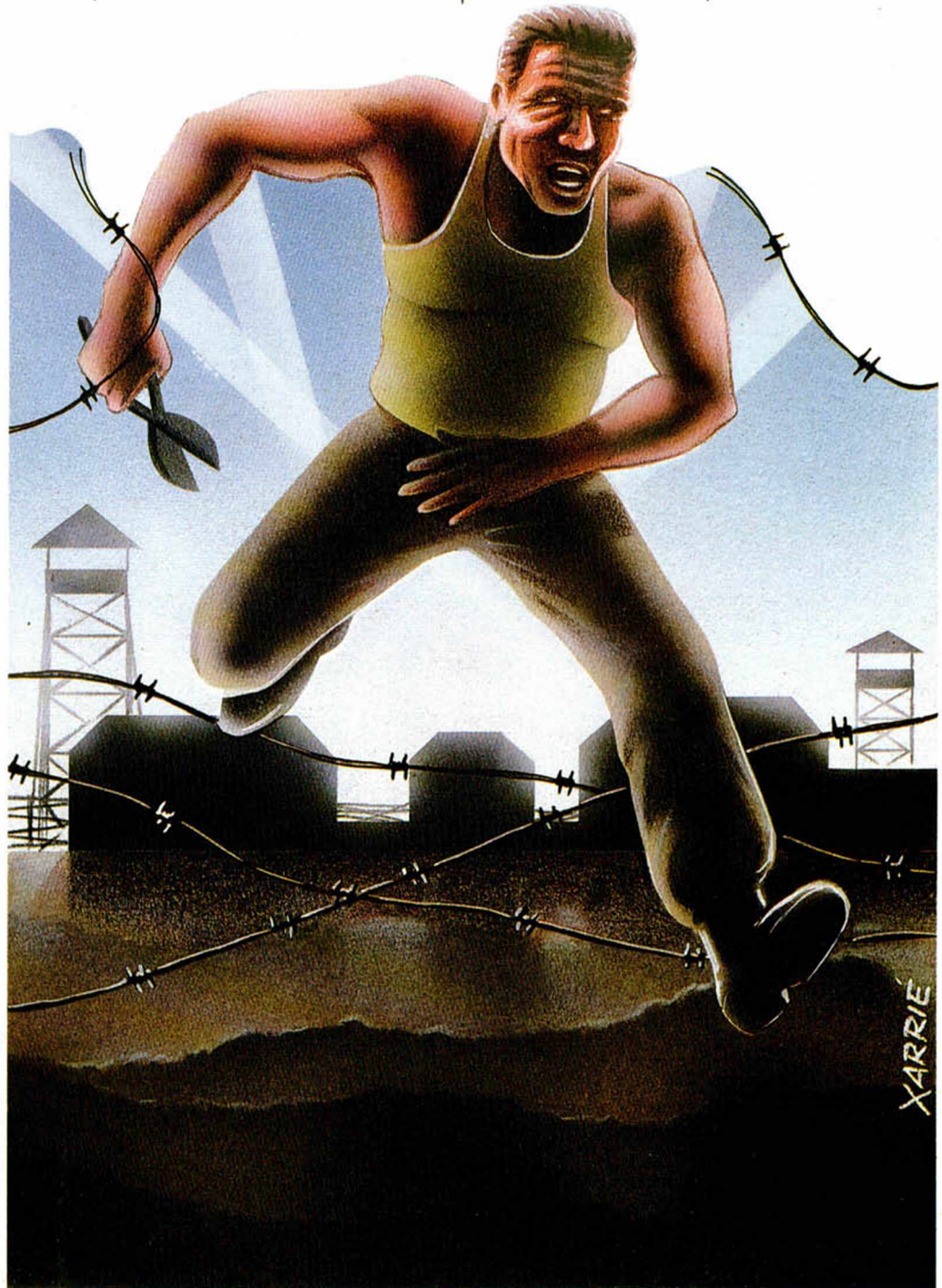
1410 a 1580—, las cuales se encargan del movimiento de la cruz y el disparo, si lo hubiera, respectivamente.

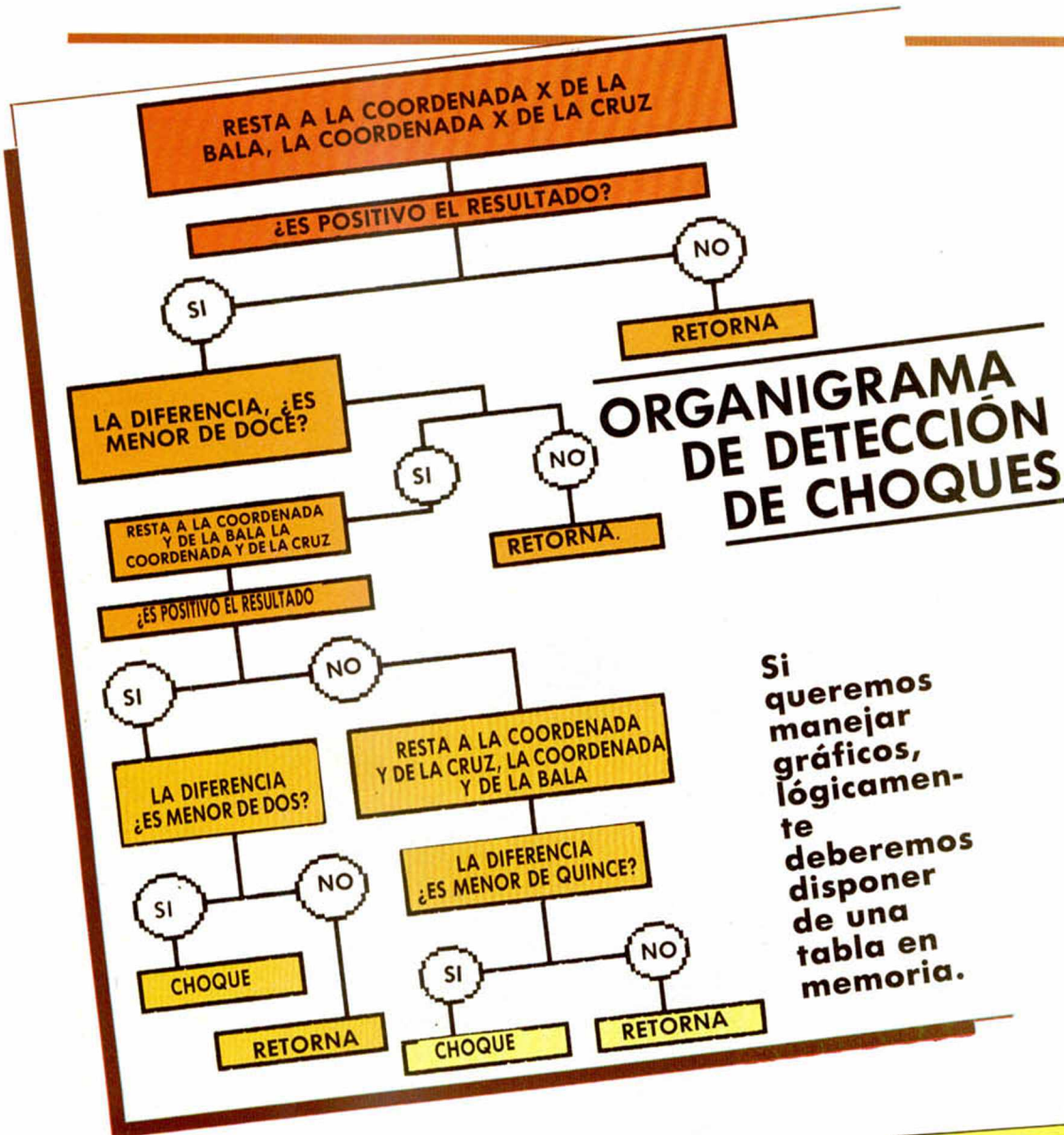
IDEN y DIREC son dos bytes señalizadores utilizados por ellas. El primero indica con un uno la existencia de un disparo en pantalla o con un cero su ausencia. El segundo adopta el valor «cero» si la cruz tiene que moverse hacia arriba o el «uno» si tiene que hacerlo hacia abajo.

Por último salta a la rutina COMPR, la cual si no se ha detectado un choque nos devuelve al bucle principal; en caso contrario retorna al Basic.

COMPROBACIÓN DE CHOQUE

La filosofía seguida en nuestra rutina para detectar una posible colisión entre sprites, responde al siguiente esquema:





Si queremos manejar gráficos, lógicamente deberemos disponer de una tabla en memoria.

Las etiquetas VAR1, VAR2, etc., que pueden observarse en el listado, están referidas a los parámetros verticales y horizontales de cada gráfico y su significado es:

VAR1: Ordenada del revolver.

VAR2: Ordenada de la cruz.

VAR3: Ordenada de la bala.

VAR4: Abscisa de la bala.

VAR5: Abscisa de la cruz.

En nuestro caso, no es posible efectuar un nuevo disparo hasta que el anterior haya desaparecido de la

pantalla, aunque no resulta demasiado complicado el hacer que esto no sea así.

Si queremos manejar gráficos, lógicamente deberemos disponer de una tabla en memoria, que nos informe cuál es la situación de cada uno de ellos —también puede indicarse su fase y color, si los tuviera, o la dirección en que debe desplazarse— a fin de que podamos ir moviéndolos secuencialmente, o llegado el caso, hacer que alguno desaparezca.

No es mala práctica el acceder a una tabla de este tipo, mediante el registro indexado IX —formato LD

IX, TABLA—. De este modo, sería posible tener en (IX+0), la ordenada; en (IX+1), la abscisa; en (IX+2), la fase, etc.

De cualquier manera, no parece demasiado práctico el hacer uso de todo esto en nuestra rutina, cuando solamente pretendemos que una única bala se desplace por la pantalla.

Prosiguiendo con la subrutina COMPR, intentaremos profundizar un poco en ella, a fin de entender su funcionamiento, que dista mucho de ser complicado.

En primer lugar, señalar algo que de por sí ya resulta bastante obvio para que un sprite se mueva hacia la derecha, es preciso incrementar su abscisa tantas unidades como desplazamientos queremos que efectúe —nuestro disparo se mueve de cuatro en cuatro pixels—; del mismo modo, para que lo haga hacia arriba debemos incrementar su ordenada. Y decrementar ambas para que se desplace en sentido contrario.

Partiendo de que el disparo se realiza desde la izquierda de la pantalla, su coordenada horizontal ha

LISTADO 1

```

10 BORDER 0: PAPER 0: CLEAR 39
999: LOAD ""CODE 4e4: LOAD ""COD
E 5e4
20 INK 7: CLS: LET A=INT (RND
*255)+1
30 IF A<=140 OR A=233 THEN GO
TO 20: REM LIMITE INFERIOR Y SUPERIOR
DE LA ABCISIA
40 LET P=INT (A/8): LET P=P*8:
REM HA DE SER MULTIPLO DE 8
50 POKE 40129, P: REM VALOR ALE
ATORIO PARA LA COORDENADA HORIZO
NTAL DE LA CRUZ
60 INK 2: FOR F=0 TO 31: PRINT
AT 4, F: "" : PRINT #1: INK 2: AT
1, F: "" : NEXT F
70 FOR F=5 TO 21: PRINT AT F, 0
: "" : AT F, 31: "" : NEXT F
80 PRINT AT 4, 0: "" : AT 4, 31: ""
90 PRINT #1: INK 2: AT 0, 0: "" :
AT 0, 31: "" : AT 1, 0: "" : AT 1, 31: ""
100 INK 6: PRINT AT 0, 4: ""-PROGR
AMA DE DEMOSTRACION-""
110 PRINT ""0. ARRIBA U. ABAJO
T. DISPARO""
120 PRINT AT 2, 4: ""SYMBOL SHIFT
PARA SALIR""
130 RANDOMIZE USR 40000
140 PRINT INK 5: BRIGHT 1: FLAS
H 1: AT 3, 12: ""OTRA VEZ?""
150 LET A$=INKEY$
160 IF A$=""S"" OR A$=""S"" THEN GO
TO 20
170 IF A$=""N"" OR A$=""N"" THEN ST
OP
180 GO TO 150
  
```

de ser, al menos, igual o un cierto número de veces — nosotros decidimos cuántas— mayor que la de la cruz. En tal caso, la sustracción entre ambas deberá ser un número positivo.

De otro modo, querría decir que la bala —horizontalmente— todavía no ha llegado a la altura de la cruz.

En el supuesto de que el disparo se moviese de derecha a izquierda, sería necesario seguir los mismos pasos, pero a la inversa, es decir, restar de la abscisa de la cruz, la de la bala, mirar si su resultado es positivo y obrar en consecuencia.

Volviendo al caso que nos ocupa, lo siguiente es comprobar cuántas veces es mayor la abscisa del disparo que la de la cruz.

Considerando que esta última ocupa 16 pixels de ancho, si el resultado de la resta fuese, pongamos por caso, dos. Significaría que el disparo se encuentra dos pixels a la derecha de la posición actual que ella ocupe. En nuestro ejemplo, se ha considerado el número 12 como un valor aceptable, por lo que si la diferencia fuese mayor que esta cantidad, se retorna al bucle principal sin hacer más comprobaciones.

En caso contrario, las coordenadas horizontales están ya suficientemente chequeadas, ocupémonos ahora de las verticales.

El proceso a seguir es muy similar al anterior: restamos a la coordenada Y de la bala, la de la cruz, si el resultado es positivo y menor de dos —scans verticales que ocupa el gráfico del disparo— querrá decir que la bala se encuentra impresa, como máximo, dos posiciones por encima del gráfico de la cruz, con lo cual se hace evidente una colisión entre ambos sprites. Si la diferencia fuese mayor de dos, la bala

se encontraría por encima de la cruz, pero sin llegar a entrar en contacto con ella, si es así, podemos regresar sin más miramientos al bucle principal.

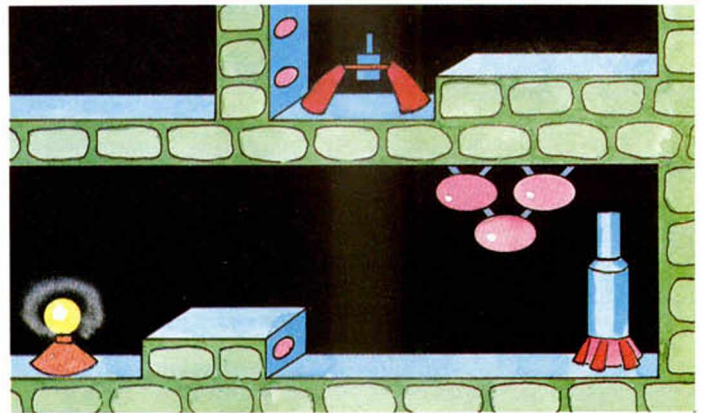
En cambio, si la resta no dio como resultado un número positivo, es fácil llegar a la conclusión de que el disparo se encuentra por debajo de la cruz. Sólo nos queda comprobar si está a menos de 15 —scans verticales que ocupa el gráfico de la cruz, menos uno— posiciones de distancia de ésta; en caso afirmativo se pone de manifiesto la existencia de un choque. De lo contrario se retorna al bucle principal.

Para finalizar, es posible que alguien se pregunte por qué, en el listado Basic la abscisa de la cruz es siempre un número múltiplo de ocho.

La respuesta es muy simple: como no está pensado que se mueva hacia los lados, si intentamos imprimirla en una coordenada horizontal que no sea múltiplo de ocho, el gráfico tenderá a aparecer en una que si lo sea, de lo que se desprende que su posición en pantalla no coincidirá con el valor asignado a su abscisa, con la consiguiente confusión que esto produciría.

Por último, cabe decir que las condiciones impuestas en el programa para que un choque se verifique son absolutamente arbitrarias, es decir, pueden variarse a gusto del consumidor y no existe ningún impedimento para hacerles los retoques necesario hasta conseguir que se adapten a nuestras pretensiones.

Se puede afirmar que el mejor aprendizaje es, sin duda, la práctica y las experiencias personales. Confío en que este artículo haya contribuido en algún modo a ello.



LISTADO 2

DUMP: 40.000 N.º BYTES: 420

LÍNEA	DATOS	CONTROL
1	CD9E9CCD8E9C3EFBDB8FE	1856
2	CB47CC769C3EFBDB8FECB	1741
3	4FCC8B9C3EFBDB8FECB67	1670
4	CC249D3E7FDBFECB4FCA	1543
5	D19DFB76F3CD649DCD3B	1704
6	9DC3A09DCB4FC83A9F9C	1524
7	FE97C8CD9E9C3A9F9C3C	1557
8	329F9C1813CB47C83A9F	1099
9	9CFE17C8CD9E9C3A9F9C	1525
10	3D329F9C06400E081150	615
11	C33E10F5C53EBFCDAC22	1379
12	06031AAE77231310F9C1	840
13	05F13D20EAC906950EE8	1175
14	1180C33E10F5C53EBFC	1318
15	AC2206021AAE77231310	603
16	F9C105F13D20EAC90628	1262
17	0E20C521E19D11A0C306	1036
18	021A77132336002310F7	553
19	79E607280E4721E19D0E	912
20	04CB1E230D20FA10F3C1	1019
21	11E19D3E02F5C53EBFC	1363
22	AC2206021AAE77231310	603
23	F9C105F13D20EAC93AE0	1498
24	9DFE00C03E0132E09D3A	1155
25	9F9CD60232DF9CCDD99C	1543
26	C93AE09DFE00C83AE19C	1533
27	FEF0300FCDDE9C3AE19C	1579
28	C60432E19CCDD99CC99C	1622
29	DE9C3E2032E19CAF320E	1352
30	9DC93ADF90FE0020183A	1167
31	BF9CFE96300EF5C0DBE9C	1609
32	F1C603328F9CCDBE9CC99	1591
33	3E0132DF9DC93ABF90FE	1353
34	1A380EF5CDBE9CF1D603	1350
35	32BF9CCDBE9CC9AF32DF	1597
36	9DC93AE09DFE0028353A	1202
37	C19C473AE19C983828FE	1364
38	0C30273ABF9C473ADF9C	1012
39	983806FE023019180C3A	637
40	DF9C473ABF9C98FE0F30	1324
41	08AF32E09D3E2032E19C	1142
42	FBC9C3469C0000FC00FC	1377

LISTADO 3

DUMP: 50.000 N.º BYTES: 82

LÍNEA	DATOS	CONTROL
1	07FC02D803FF63F00144	1143
2	10014413FF43F0088003	805
3	F0940400A9F800AB0800	988
4	AC8000AA1000ABE000BA	1137
5	00008200007C000003C0	449
6	03C003C003C003C003C0	975
7	FFFFFFFFFFFFFFFF03C0	2235
8	03C003C003C003C003C0	975
9	FCFC0000000000000000	504

LISTADO ENSAMBLADOR

10	ORG 40000		
20	CALL REVOL ;Primera	impresion del revolver	
30	CALL CRUZ ;Y la cru	z, con over 1	
40	PRINC LD A,#FB ;Chequea	la tecla Q	
50	IN A,(#FE)		
60	BIT 0,A ;Esta puls	ada?	
70	CALL Z,ARRIBA ;si 1	o esta, mueve arriba	
80	LD A,#FB ;Chequea	la tecla W	
90	IN A,(#FE)		
100	BIT 1,A ;Esta puls	ada?	
110	CALL Z,ABAJO ;Si lo	esta, mueve abajo	
120	LD A,#FB ;Chequea	la tecla T	
130	IN A,(#FE)		
140	BIT 4,A ;Esta puls	ada?	
150	CALL Z,DISP ;Si lo	esta, dispara	
160	LD A,#7F ;Chequea	Symbol Shift	
170	IN A,(#FE)		
180	BIT 1,A ;Esta puls	ada?	
190	JP Z,CHOQUE ;Si 1	o esta, retorna al Basic	
200	EI ;Sincroniza co	n	
210	HALT ;el barrido		
220	DI ;de la pantalla	a	
230	CALL MCRUZ ;Mueve 1	a cruz y	
240	CALL MDISP ;el disp	aro, si se ha producido	
250	JP COMPR ;Salta a	comprobacion de choque	
260	ARRIBA BIT 1,A ;Esta puls	ada la W?	
270	RET Z ;Retorna si	lo esta	
280	LD A,(VARI) ;La o	rdenada del revolver	
290	CP #97 ;Ha alcanz	ado el limite superior?	
300	RET Z ;En caso afi	rmativo, retorna	
310	CALL REVOL ;Borra f	igura anterior	
320	LD A,(VARI) ;Incr	ementa en uno	
330	INC A ;la ordenada	y la	
340	LD (VARI),A ;vuel	ve a introducir	
350	JR REVOL ;Imprime	la figura en la nueva posicion	
360	ABAJO BIT 0,A ;Esta puls	ada la Q	
370	RET Z ;Retorna si	lo esta	
380	LD A,(VARI) ;La o	rdenada del revolver	
390	CP #17 ;Ha alcanz	ado el limite inferior?	
400	RET Z ;En caso afi	rmativo, retorna	
410	CALL REVOL ;Borra f	igura antigua	
420	LD A,(VARI) ;Toma	la ordenada	
430	DEC A ;para decrem	entarla en uno	
440	LD (VARI),A ;y vo	lver a introducirla	
450	REVOL LD B,#40 ;Ordenad	a del revolver	
460	LD C,#00 ;Abscisa		
470	LD DE,#C350 ;Dire	ccion del grafico del revo	
480	LD A,#10 ;Numero	de scans verticales	
490	IMPRIM PUSH AF		
500	PUSH BC ;Guarda coo	rdenadas	
510	LD A,#BF ;Calcula	la direccion	
520	CALL PIXEL ;del fic	hero de pantalla	
530	LD B,#03 ;e impri	me	
540	IMP0 LD A,(DE) ;mezcla	ndo los	
550	XOR (HL) ;datos de	l grafico con	
560	LD (HL),A ;lo que	haya en pantalla	
570	INC HL		
580	INC DE		
590	DJNZ IMP0		
600	POP BC ;Recupera c	oordenadas y	
610	DEC B ;decrementa	la ordenada	
620	POP AF ;hasta comp	letar la figura	
630	DEC A		
640	JR NZ,IMPRIM		
650	RET		
660	CRUZ LD B,#149 ;Ordenad	a	
670	LD C,#EB ;Abscisa		
680	LD DE,#C300 ;Dire	ccion del grafico de la cruz	
690	LD A,#10 ;Numero	de scans verticales	
700	IMPR1 PUSH AF		
710	PUSH BC ;Guarda coo	rdenadas	
720	LD A,#BF ;calcula	la direccion	
730	CALL PIXEL ;del fic	hero de pantalla	
740	LD B,#02		
750	12 LD A,(DE) ;E impr	ine	
760	XOR (HL) ;del mism	o modo que	
770	LD (HL),A ;la sub	rutina Revol	
780	INC HL		
790	INC DE		
800	DJNZ 12		
810	POP BC ;Recupera c	oordenadas y	
820	DEC B ;decrementa	la ordenada hasta	
830	POP AF ;finalizar	la impresion	
840	DEC A		
850	JR NZ,IMPR1		
860	RET		
870	IDIS LD B,#28 ;Ordenad	a del disparo	
880	LD C,#20 ;Abscisa		
890	PUSH BC ;Guarda coo	rdenadas	
900	LD HL, BufER		
910	LD DE,#C3A0 ;Dire	ccion del grafico del disparo	
920	LD B,#02		
930	METE LD A,(DE) ;Mete 1	os datos en	
940	LD (HL),A ;el buf	er e introduce	
950	INC DE ;un cero de	spues de cada	
960	INC HL ;linea a fi	n de poder	
970	LD (HL),#00 ;rota	rlas si la abscisa	
980	INC HL ;no fuese u	n numero	
990	DJNZ METE ;multiplo	de ocho	
1000	LD A,C		
1010	AND #07 ;Si lo es,	sigue sin	
1020	JR Z,SIGUE ;hacer	ninguna operacion	
1030	LD B,A		
1040	RO1 LD HL, BufER		
1050	LD C,#04		
1060	RO2 RR (HL) ;Rota los	datos del	
1070	INC HL ;bufere para	su	
1080	DEC C ;posterior i	mpresion	
1090	JR NZ,RO2		
1100	DJNZ RO1		
1110	SIGUE POP BC ;recupera c	oordenadas	
1120	LD DE, BufER ;y lo	imprime	
1130	LD A,#02 ;Numero	de scans verticales	
1140	IMDI PUSH AF		
1150	PUSH BC		
1160	LD A,#BF		
1170	CALL PIXEL		
1180	LD B,#02		
1190	ESCR LD A,(DE) ;Imprim	e el grafico	
1200	XOR (HL) ;del disp	aro mezclando	
1210	LD (HL),A ;el con	tenido del	
1220	INC HL ;bufere con	lo que	
1230	INC DE ;haya en pa	ntalla	
1240	DJNZ ESCR		
1250	POP BC		
1260	DEC B		
1270	POP AF		
1280	DEC A		
1290	JR NZ,IMDI		

```

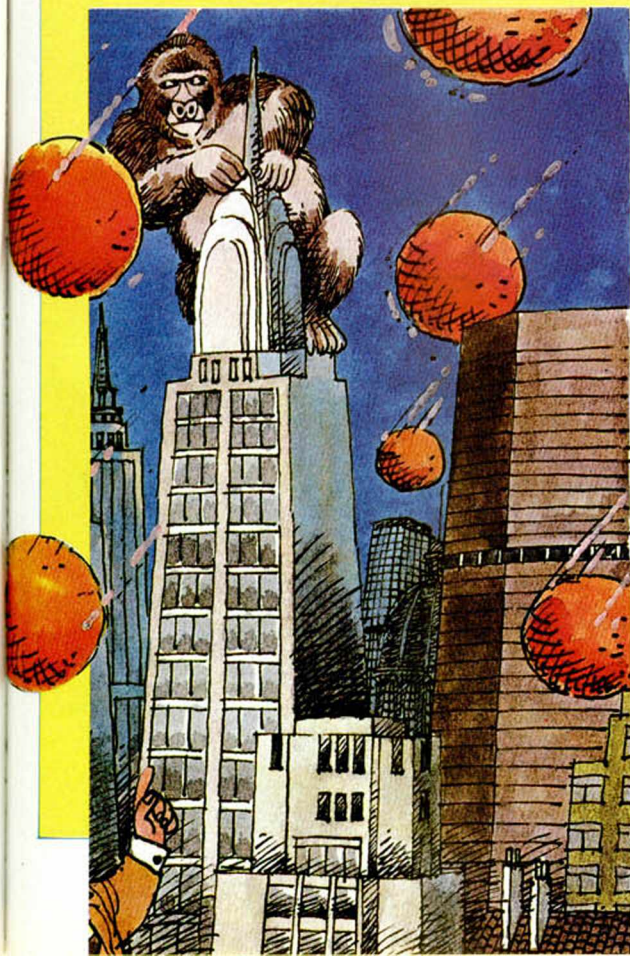
1300 RET
1310 DISP LD A,(IDEN);Hay ya un disparo efectuado?
1320 CP #00;En caso a afirmativo
1330 RET NZ;retorna
1340 LD A,#01;Ahora s i ha de haberlo
1350 LD (IDEN),A; y lo hace constar
1360 LD A,(VAR1);Toma la ordenada del revolver
1370 SUB #02;para decr ementarla en dos e
1380 LD (VAR3),A;intr oducirla en la del disparo
1390 CALL IDIS;Imprime su grafico y
1400 RET;retorna
1410 MDISP LD A,(IDEN);Hay un disparo
1420 CP #00;en pantal la?
1430 RET Z;Retorna si no lo hay
1440 LD A,(VAR4);Su a bscisa ha alcanzado
1450 CP #08;el valor maximo?
1460 JR NC,BORRA;Si e s asi, restablece parametr
1470 CALL IDIS;Borra la figura anterior
1480 LD A,(VAR4);incr ementa su
1490 ADD A,#04;abscisa en cuatro
1500 LD (VAR4),A;unid ades,
1510 CALL IDIS;la impri me en la nueva posicion
1520 RET; y retorna
1530 BORRA CALL IDIS;Borra gr afico del disparo
1540 LD A,#20;restaur a su abscisa
1550 LD (VAR4),A;e in dica que ya
1560 XOR A; puede produ cirse
1570 LD (IDEN),A;un n uevo disparo
1580 RET
1590 MCRUZ LD A,(DIREC);Si el indicador de sentido
1600 CP #00;es un num ero distinto de 0
1610 JR NZ,ABA;la cru z se mueve hacia abajo
1620 LD A,(VAR2);Si s u ordenada ha alcanzado el

```

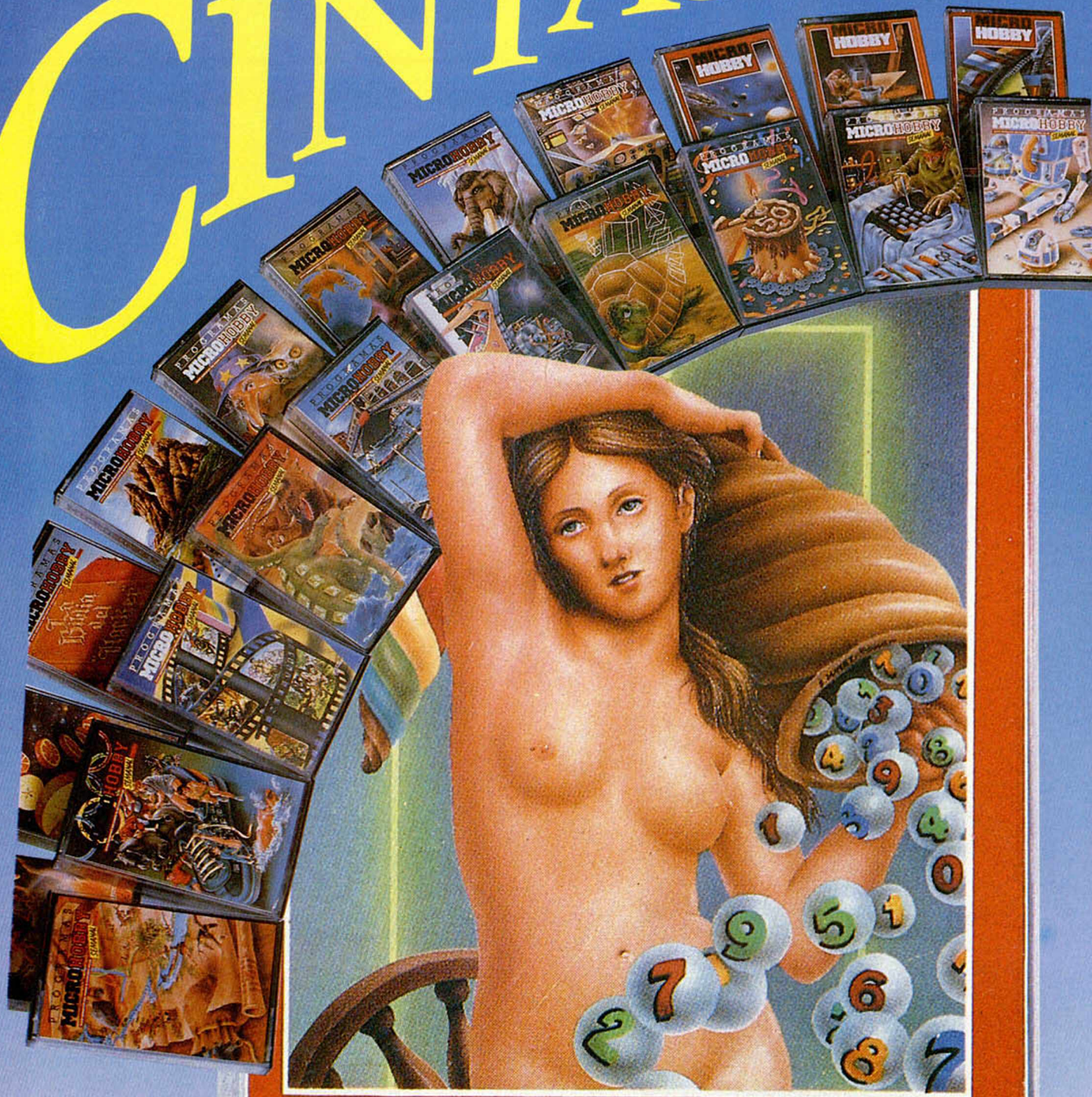
```

1630 CP #96;limite ma ximo, es preciso
1640 JR NC,FIND;canbi ar el sentido
1650 PUSH AF;Guarda ord enada
1660 CALL CRUZ;borra la figura actual
1670 POP AF;recupera o rdenada
1680 ADD A,#03;la incr ementa en tres
1690 LD (VAR2),A;impr ime la cruz
1700 CALL CRUZ;en su nu eva posicion y
1710 RET;retorna
1720 FIND LD A,1;Indica qu e la cruz
1730 LD (DIREC),A;se mueve hacia abajo
1740 RET
1750 ABA LD A,(VAR2);La o rdenada de la cruz
1760 CP #1A;ha alcanz ado el limite inferior?
1770 JR C,FIND2;Si es asi, cambia su sentido
1780 PUSH AF;Guarda la ordenada,
1790 CALL CRUZ;Borra la anterior figura
1800 POP AF;recupera o rdenada
1810 SUB #03;la decrem enta en
1820 LD (VAR2),A;tres unidades
1830 CALL CRUZ;la impri me de nuevo y
1840 RET;retorna
1850 FIND2 XOR A;Ahora la cr uz ha de
1860 LD (DIREC),A;mov erse hacia arriba
1870 RET
1880 *SUBROUTINA DE CHOQUE*
1890 COMP LD A,(IDEN);Hay un disparo
1900 CP #00;en pantal la?
1910 JR Z,VUELVE;Si n o lo hay retorna
1920 LD A,(VAR5);Rest a a la abscisa
1930 LD B,A;del dispa ro
1940 LD A,(VAR4);la d e la cruz
1950 SBC A,B;si el res ultado no es un numero positivo
1960 JR C,VUELVE;reto rna
1970 CP #0C;Si lo es, ha de ser menor de doce
1980 JR NC,VUELVE;En caso contrario retorna
1990 LD A,(VAR2);Rest a a la ordenada
2000 LD B,A;del dispa ro
2010 LD A,(VAR3);la d e la cruz
2020 SBC A,B;Si el res ultado no es positivo
2030 JR C,NO;salta a la subrutina No
2040 CP #02;Si es pos itivo, pero mayor de dos
2050 JR NC,VUELVE;ret orna
2060 JR CHOQUE;si es menor, se detecta un choque
2070 NO LD A,(VAR3);Rest a a la ordenada
2080 LD B,A;de la cru z
2090 LD A,(VAR2);la d e el disparo
2100 SBC A,B;si el res ultado es
2110 CP #0F;menor de quince, se detecta un choque
2120 JR NC,VUELVE;si es mayor retorna
2130 CHOQUE XOR A;Restaura el byte
2140 LD (IDEN),A;iden tificador del
2150 LD A,#20;disparo , repone el valor
2160 LD (VAR4),A;inic ial de su abscisa
2170 EI;habilita las interrupciones
2180 RET; y retorna al Basic
2190 VUELVE JP PRINC;Regresa al bucle principal
2200 DIREC DEFB #0
2210 IDEN DEFB #0
2220 BUFER DEFS #4
2230 PIXEL EQU 8876
2240 VAR1 EQU 40095
2250 VAR2 EQU 40127
2260 VAR3 EQU 40159
2270 VAR4 EQU 40161
2280 VAR5 EQU 40129

```



"LOAD": CINTAS!



Recorta o copia este cupón y envíalo a Hobby Press, S.A. Apartado de Correos nº 8. 28100 Alcobendas (Madrid).

Deseo recibir en mi domicilio las cintas de MICROHOBBY que a continuación indico, al precio de 625 ptas. cada una. Cada cinta lleva grabados los programas publicados por MICROHOBBY durante cuatro números consecutivos (1 al 4, 5 al 8, 9 al 12, etc.).

Números _____ al _____ Números _____ al _____ Números _____ al _____
 Nombre _____ Apellidos _____ Localidad _____ Provincia _____ Fecha de Nacimiento _____
 Domicilio _____ C. Postal _____ Teléfono _____

Formas de pago (Para agilizar tu envío, es importante que indiques el código postal)

Talón bancario adjunto a nombre de Hobby Press, S.A. Giro Postal a nombre de Hobby Press, S.A., nº _____
 Contra reembolso (supone 125 ptas. más de gastos de envío y es válido sólo para España).
 Tarjeta de crédito nº _____ (Sólo para pedidos superiores a 1.500 ptas.)
 Visa Master Card American Express Fecha de caducidad de la tarjeta _____ Nombre del titular (si es distinto) _____ Fecha y firma _____
 (Si lo deseas puedes solicitarlas por teléfono (91) 734 65 00)

MICRO-1

C/. Duque de Sesto, 50. 28009 Madrid (Metro O'Donnell o Goya)
Tel. (91) 275 96 16 - 274 75 02

**SOFTWARE:
POR CADA DOS PROGRAMAS, GRATIS A ELEGIR**
- CASCOS STEREO
- RELOJ DIGITAL + BOLÍGRAFO LACADO
- RELOJ DIGITAL ROBOT O AVIÓN

	PTAS.		PTAS.
FIST II	875	XEVIUS	875
DEEP STRIKE	875	10th FRAME	1.200
SUPER SOCCER	875	LEADERBOARD	1.200
TERRA CREST	875	EXPRESS RAIDER	875
DOUBLE TAKE	875	ACE OF ACES	1.200
SHORT CIRCUIT	875	IMPOSSABALL	875
ARKANOID	875	SIGMA 7	875
UCHI-MATA	875	BAZZOKA BILL	875
INSPECTOR GADGET	875	DRAGON'S LAIR II	875
SHAO LIN'S ROAD	1.750	SHADOW SKIMMER	875
SOFTWARE AMSTRAD DISCO	2.250	(Incluido regalo calculadora)	

SPECTRUM PLUS +
CASCO MÚSICA STEREO
19.800 PTS (incl. IVA).

OFERTAS YOUSTICKS

	PTAS.
QUICK SHOT I	995
QUICK SHOT II	1.195
QUICK SHOT II TURBO	2.695
QUICK SHOT IX	1.995
KONIX (microswitch)	2.595
INTERFACE SPECTRUM	1.195

IMPRESORAS 20% DTO. SOBRE P.V.P.

CABLES E INTERFACES
20% DTO. SOBRE P.V.P.

CADENA MUSICAL 27.900 PTS.
VIDEO VHS AKAI 79.900 PTS.
RADIOCASSETTE STEREO 6.895 PTS.

SOLICITA GRATIS
NUESTRO CATÁLOGO A
TODO COLOR, DE
NUESTROS PRODUCTOS

RATÓN PARA AMSTRAD Y COMMODORE CON SOFTWARE 4.900 PTS.

PEDIDOS CONTRA REEMBOLSO SIN GASTOS

DE ENVÍO (si es inferior a 1.200 ptas. se cargarán

150 ptas). LLAMA POR TELÉFONO. ADELANTAS TRES DÍAS TU PEDIDO TELF. (91) 274 75 02 /

(91) 275 96 16

(Durante las 24 horas)

SERVICIO TÉCNICO REPARACIÓN TARIFA FIJA: 3.600 PTAS.
(incluido provincias sin gastos envío)

CASSETTE ESPECIAL ORDENADOR 3.495 PTAS. Y 3.995 PTAS.

COMPATIBLE PC-IBM 640 K

2 BOCAS 360 K

MONITOR FÓSFORO VERDE

149.900 PTAS. (incluido IVA)

CASSETTE ESPECIAL ORDENADOR
3.495 PTS. Y 3.995 PTS.

COMMODORE 128 54.900

COMMODORE 128 + TECL. MUSICAL.... 57.900

	PTAS.
DISKETTE 3"	695
DISKETTE 5 1/4" DC/DD	190
LÁPIZ ÓPTICO SPECTRUM ..	2.890
LÁPIZ ÓPTICO AMSTRAD ..	2.890
CINTA C-15 SPECTRUM	69
MICRODRIVE	495
ARCHIVADOR DISCO 3"	2.600
RALENTIZADOR DE JUEGOS ..	995

¡¡PRECIOS EXCEPCIONALES PARA TU AMSTRAD!!

Tiendas y Distribuidores, pidan lista de precios al mayor. C/. Galatea, 25 28042 - MADRID telef. (91) 274 75 03

A la hora de adentrarnos en el mundo de la programación resulta imprescindible

Rafael MARQUEZ PARRA

conocer las diferentes técnicas que permiten diseñar el mapeado de un juego. En estas páginas encontraréis una rutina que desarrolla una de estas técnicas.

TECN

MAPEADO

Para facilitar su comprensión vamos a guiarnos por un programa ejemplo. Si observáis detenidamente cada paso, dentro de muy poco podréis diseñar fácilmente vuestros propios mapeados. Hemos elegido para ejemplificar nuestra rutina el mapeado de una casa.

La hemos diseñado de forma que quepa en nueve pantallas del televisor. En cada una podrán verse dos pisos y tendrá tres habitaciones en el plano horizontal; por tanto estará compuesta por 18 lugares distintos. Para hacerlo más real, el último piso lo convertiremos en el tejado de la casa y será una zona donde no se podrá acceder.

Ahora hay que diseñar gráficos para adornar el interior. Deben ser lo más variados posible y podrán estar en más de una habitación.

Cuando ya están dibujados hay que almacenarlos en memoria. Estos gráficos no se almacenan como los GDU, sino por scans, es decir, primero la primera fila de bytes de la primera línea de caracteres que componen el gráfico, después la segunda fila de la primera línea de caracteres y así hasta completar todo el gráfico. (Fig. 1).

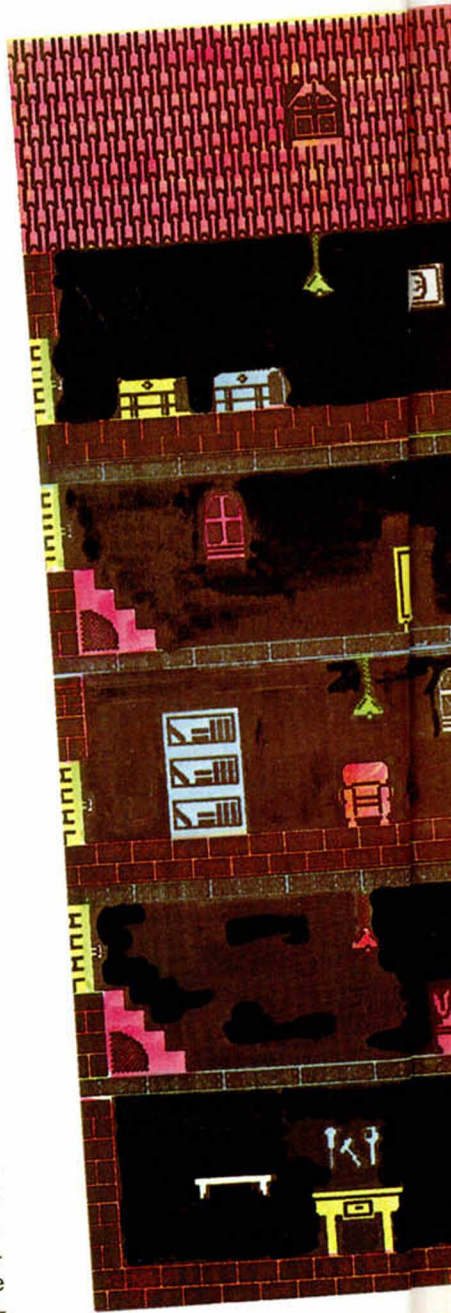
Con todos los gráficos ya almacenados hay que construir una tabla con las características de cada uno. A esta tabla le llamaremos **TABLA DE GRÁFICOS**. Cada gráfico ocupa cuatro bytes de la tabla. El primero indica el número de bits de ancho, el segundo el número de bits (o scans) de alto, el tercero y el cuarto contienen la dirección donde está almacenado. (Fig. 2).

Ahora hay que distribuir los objetos por las habitaciones a nuestro gusto.

Cuando ya lo hayamos hecho, hemos de construir una tabla que nos indique qué gráfico hay en cada pantalla. Le llamaremos **TABLA DE PANTALLA**.

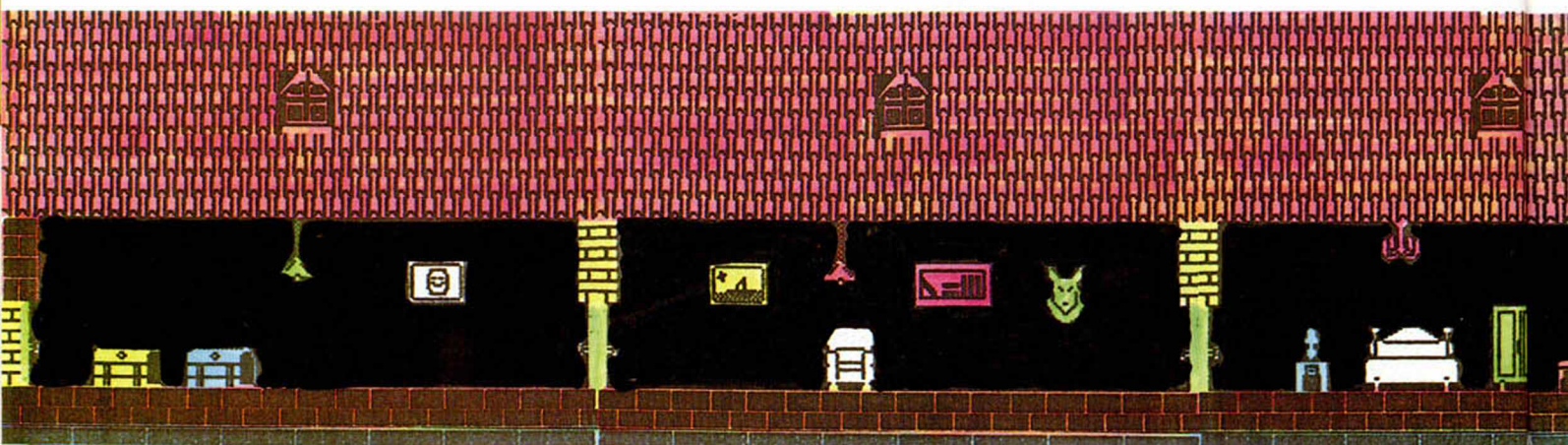
El primer byte de la tabla indica el número de objetos que hay en ella (no se cuentan las paredes, puertas, techo, suelo, escaleras ni el tejado con la ventana que hay en él, ya que éstos reciben un tratamiento diferente).

Después de este byte vienen cuatro más para cada objeto. El primero es el número de identificación del gráfico, que es la posición que ocupa en la **TABLA DE GRÁFICOS**. El segundo byte indica la posición vertical del objeto y el tercero la posición horizontal (estos valores se toman con pantalla en alta resolución; igual que el PLOT y DRAW). El último byte indica el color del gráfico y se halla de la siguiente manera: CO-



ICAS DEL





LOR DE TINTA+COLOR DEL PAPEL*8=BRILLO*64+FLASH*128.

Por tanto, si en una habitación hay una mesa, una silla y una ventana, la TABLA DE PANTALLA tendrá una longitud de 13 bytes, el primero el contador de objetos y cuatro para cada gráfico que haya.

Ya sólo falta hacer una tabla, la **TABLA DE DATOS**. Esta tiene dos bytes por cada pantalla, es decir 18. Estos bytes contienen la dirección donde comienza la TABLA DE PANTALLA de cada una de ellas.

Ahora ya está desarrollando todo lo necesario para empezar a explicar las rutinas que lo realizan.

También haremos que un pequeño personaje pueda visitar todas las habitaciones de la casa.

EXPLICACIÓN DEL PROGRAMA

Empieza en la línea 100 con un salto a SEGUIR. Entre las líneas 120 y 260 están las etiquetas utilizadas por el programa.

Entre las líneas 280 y 700 se encuentra la TABLA DE GRÁFICOS.

Entre las líneas 720 y 00 está la TABLA DE DATOS que contiene la dirección de comienzo de la TABLA DE PANTALLA de cada uno de ellas.

850-870: Papel negro y tinta blanca.

880-910: Se inicializan las coordenadas del muñeco.

920-930: Se pone a 0 la variable SALTO. Estará a 1 si el muñeco tiene que saltar.

940-950: Se inicializa a 1 la variable FORMA. Puede contener un número del 1 al 8. Cada uno corresponde a un gráfico distinto del muñeco.

960-970: Se almacena en PANT el número de habitaciones donde se encuentra el muñeco. Las habitaciones se numeran de izquierda a derecha y de arriba a bajo empezando por 1.

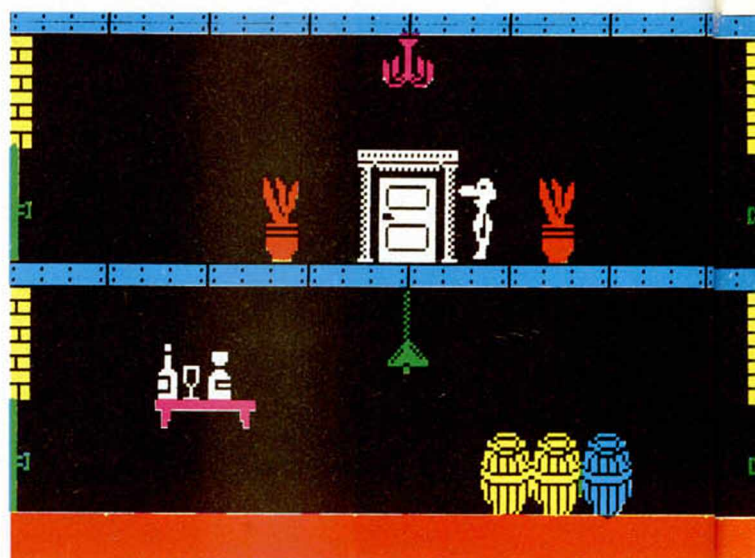


TABLA DE PANTALLA

W	Z0	.	J
0	0F	ZJ	ZF
0	0H	00	0Z
00	0J	HO	HO
0F	00	OH	ON
0H	00	HF	HH
0J	00	00	00
00	00	00	00
00	00	00	00
00	00	00	00

TABLA DE GRÁFICOS

NUMERO DE BYTES ANCHO	NUMERO SCANNS DE ALTO	DIRECCION DEL GRAFICO
1 BYTE	2 BYTE	3 Y 4 BYTE



DATOS FIG. 1

```

0,0,0,0,0,0,0,0,0,0,0,28,0,0,62,0,
0,127,0,0,127,0,0,85,0,0,127,0,0,
109,0,0,127,0,0,127,0,0,62,0,1,
221,128,5,227,224,13,247,176,13,
235,184,14,255,184,31,255,124,31
255,252,63,255,254,61,255,222,1
24,85,206,124,255,142,120,255,20
6,249,255,197,209,255,231,224,12
7,143,113,140,110,113,243,224,33
243,224,1,243,224,1,243,224,1,2
43,224,3,243,240,3,243,240,3,243
240,2,225,208,3,33,48,2,161,80,
3,97,48,2,192,208,3,64,176,2,192
208,5,64,168,10,192,212,21,128,
106,14,0,28,192
    
```

990-1020: CLS y borde azul. Se marca la variable MODO a 1. Si tuviera un 0, la rutina de impresión no imprimiría.

Entre las líneas 1030 y 2730 se imprimen las figuras correspondientes a cada pantalla que no sean objetos, como paredes, puertas, etc. Para imprimirlos se carga en DE la dirección que ocupa el gráfico en la TABLA DE GRÁFICOS y en HL el comienzo de los datos que contienen las coordenadas y el color del gráfico a imprimir.

Después se llama a SITUA que introduce los valores en las etiquetas para que pueda utilizarlos la rutina de impresión e imprime el gráfico.

1030-1050: Se imprime un gráfico del suelo.

1060-1130: Se crea un bucle para que imprima 8 gráficos horizontalmente. Entre 1100 y 1120 se incrementa la coordenada.

1140-1170: Se imprime el gráfico del techo y se llama a SUB1.

1180-1210: Se cambia la coordenada Y para imprimir en la parte superior de la pantalla otro gráfico del techo. Llama a SUB1 y salta CONT1.

1220-1300: Se crea un bucle para imprimir 8

gráficos del techo.

1310-1370: Si el muñeco está en las pantallas 3, 6 ó 9 salta a MURO2.

1380-1400: Se imprime una pared en el piso bajo de la pantalla.

1410-1430: Se imprime una puerta en el piso bajo.

1440-1460: Se comprueba si se está en las dos primeras pantallas y si es así se salta a CONT2.

1470-1490: Imprime una puerta en el piso alto de la pantalla.

1500-1530: Imprime una pared en el piso alto y salta a CONT2.

1540-1560: Aquí se llega si está en la pantalla 3, 6 ó 9. Se imprime un muro.

1570-1590: Si estamos en la pantalla 3 salta a TRES.

1600-1750: Imprime dos puertas, dos paredes y un muro. Salta a CONT2.

1760-1780: Imprime un muro.

1790-1850: Si estamos en la pantalla 1, 4 ó 7 salta a MUR.

1860-1910: Se imprime una pared y una puerta.

1920-1940: Si estamos

en la pantalla 1 ó 2 salta a MUR.

1950-2010: Imprime una pared, una puerta y salta a TEJADO.

2020-2040: Si estamos en la pantalla 2 salta a TEJADO.

2050-2130: Imprime una pared, un muro y una puerta.

2140-2160: Si estamos en la primera pantalla salta a TEJADO.

2170-2250: Imprime una pared, una puerta y un muro.

2260-2280: Si no estamos en las 3 primeras pantallas salta a CONT3.

2290-2310: Imprime el gráfico del tejado.

2320-2460: Crea un bucle para imprimir cinco filas de gráficos que forman el tejado.

2470-2490: Imprime la ventana del tejado.

2500-2550: Si estamos en la pantalla 4 ó 7 salta a ESC, sino a CONT4.

2560-2580: Imprime una escalera.

2590-2640: Si estamos en la pantalla 6 ó 9 va a ESCB, sino a CONT5.

2650-2670: Imprime una escalera.

2680-2730: Si estamos en la pantalla 7 imprime un muro.

En SIG comienza la rutina que lee los datos de las pantallas y crea las habitaciones.

2760: Se carga en A el número de pantalla donde está el muñeco.

2770: Se carga en B para hacer un bucle.

2780-2810: Se calcula la posición de la pantalla en la TABLA DE DATOS.

2820-2860: Se pasa la dirección de la TABLA DE DATOS de la pantalla donde está el registro HL.

2870-2880: Carga en B el número de gráficos a imprimir.

2890: Se guarda el registro B para no perder su contenido.

2900-2930: Se carga en B el número de identificación del primer gráfico.

2940-2990: Halla la posición de los datos del gráfico a imprimir en la tabla de gráficos.

3000: Imprime el gráfico.

3010: Recupera el contador y repite si aún faltan más.

3030-3060: La pantalla ya está completa y se almacena en memoria.

3070: Salta a SEGU2.
3090-3330: Esta rutina

coloca los datos de los gráficos en las etiquetas para que pueda utilizarlos la rutina de impresión.

Carga la anchura del gráfico en ANCHO, la altura en ALTO; el byte bajo de la dirección donde está almacenado en INFER y el alto en SUPER; la coordenada Y en POSY, la X en POSX y los atributos en COLOR. Luego llama a IMPRI que dibuja el gráfico y retorna. Los datos que hay entre las líneas 3360 y 3610 sirven para imprimir las puertas, paredes, etc. El primer byte es la posición Y, el segundo la X y el tercero el color del gráfico. Los datos que hay entre las líneas 3630 y 3700 indican la dirección donde están almacenadas las figuras para crear el movimiento del muñeco.

Las primeras 4 son del movimiento a la derecha y las 4 últimas a la izquierda. La figura 1 y 3 de cada sentido son iguales.

3720-3800: Pone los datos iniciales del muñeco. Llama a BFORM que busca la forma que le corresponde. Después se llama a PCOOR que coloca las coordenadas del muñeco para que pueda utilizarlas la rutina de impresión, y por último lo imprime.

3830-3860: Si se ha pulsado la tecla Z se salta a PULIZ.

LISTADO 1

```
10 LOAD "" CODE 50000
20 LOAD "" CODE 64000
30 LOAD "" CODE 55104
40 LOAD "" CODE 45004
50 RANDOMIZE USR 50000
```

LISTADO 2 (RUTINAS)

```
1 C311C4101840D7101871 880
2 D71018A2D70820D3D720 1130
3 20F4D7181875D81018BE 1102
4 D82008EFD8102010D920 1024
5 1051D9181092D92818C3 976
6 D920083CDA08205DDA18 910
7 107EDA1810AFDA1810E0 1057
8 DA101811DB101842DB20 851
9 1073DB1810B4DB2020E5 1082
10 DB182066DC1010C7DC10 1064
11 18E8DC101819DD18184A 884
12 DD101893DD2010C4DD10 1110
13 1805DE181836DE28187F 766
14 DE1018F8DE201029DF20 1076
15 186ADF1010CBDF1018EC 1087
16 DF10181DE008204EE020 890
17 206FE01818F0E0101839 976
18 E110186AE1CCAFDDAFF2 1613
19 AF07B028B059B07EB097 1292
20 B0C0B03E02CD01163E07 905
21 328D5C32485C3E603292 851
22 E23E7F3291E2AF3294E2 1435
23 3C3293E23E083290E2CD 1178
24 6B0D3E01D3FE32065B21 828
25 9BC61177C3CD2DC60608 1146
26 C5CD00FAC13A045BC620 1228
27 32045B10F1219EC6116F 919
28 C3CD2DC6CD6FC43EAF32 1442
29 055BCD6FC418120608C5 861
30 CD00FAC13A045BC62032 1081
31 045B10F1C93A90E2FE03 1236
32 2835FE062831FE09282D 790
33 2156C61187C3CD2DC621 1145
34 59C611EBC3CD2DC63A90 1384
35 E2FE03385C215CC611EB 1206
36 C3CD2DC6215FC61187C3 1316
37 CD2DC618482189C61173 1044
38 C3CD2DC63A90E2FE0328 1368
39 2F218CC611EBC3CD2DC6 1313
40 2156C61187C3CD2DC621 1145
41 92C611EBC3CD2DC62195 1421
42 061187C3CD2DC6218FC6 1367
43 1173C3CD2DC618092198 993
44 C61173C3CD2DC63A90E2 1401
45 FE012835FE042831FE07 956
46 282D2162C61187C3CD2D 1011
47 C62168C6115FC3CD2DC6 1288
48 3A90E2FE0338142165C6 1093
49 1187C3CD2DC62168C611 1150
50 5FC3CD2DC618443A90E2 1258
51 FE02283D217AC61173C3 1037
52 CD2DC6217DC61187C3CD 1356
53 2DC62180C6115FC3CD2D 1159
54 C63A90E2FE01281B2165 1082
55 C61187C3CD2DC62183C6 1355
56 115FC3CD2DC62186C611 1137
57 73C3CD2DC63A90E2FE04 1444
58 303111D7C32177C6CD2D 1124
59 C60605C50608C5CD00FA 1072
60 C13A045BC62032045B10 737
61 F13A055BD61032055BC1 964
62 10E32174C611CB3CD2D 1255
63 C63A90E2FE042806FE07 1191
64 28021809216EC611EFC3 867
65 CD2DC63A90E2FE062806 1182
66 FE09280218092171C611 699
67 63C3CD2DC63A90E2FE07 1431
68 200921A1C61173C3CD2D 1010
69 C63A90E24711FDC31313 1200
70 10FC1A6F131A677E47C5 947
71 237E2347114FC3131313 615
72 1310FACD2DC6C110EC21 1211
73 004011B88801001BEDB0 842
74 C3B4C6F31A32075B131A 1035
75 32085B131A320A5B131A 390
```

DUMP: 50000
N.º BYTES: 1.472

```

76 320B5B7E32055B237E32 635
77 045B237E32095B5E5CD00 840
78 FAE1FBC95FF8063FF844 1655
79 87F844A7F8065F0006A7 1140
80 00063F00448700448710 491
81 433FD043977802AF0002 855
82 5F00163F00063F0844A7 492
83 08448700163FF0165FF0 893
84 44A7F01687F04487F806 1329
85 5FF0161F00166700283F 616
86 001640D771D740D7A2D7 1285
87 6AE1ECDF6AE11DE03E10 1452
88 32075B3E1832085B3E07 452
89 32095BCDE0C7CDF5C7CD 1632
90 00FA3EFEDBFECB4F2824 1397
91 3EFEDBFECB5728703E7F 1420
92 DBFECB47CCA0C73EBFDB 1782
93 FECB47C83A94E2B7C2AF 1712
94 C7CC1EC818D43A92E23D 1360
95 4F3A91E2D61747C5CDAA 1388
96 22CD05C8C11AFE44CA80 1315
97 C8FE1628C7FE432009CD 1282
98 CE22CDD52DB720BA3A93 1309
99 E23CFE09D4DAC7FE05DC 1657
100 DAC73293E2CDE0C73A92 1672
101 E2D6023292E2CD10C8CD 1490
102 F5C7010500CD3D1F1890 915
103 3A92E2C6104F3A91E2D6 1366
104 1747C5CDAA22CD05C8C1 1303
105 1AFE44CACCC8FE16CADC 1652
106 C6FE43200ACDCE22CDD5 1424
107 2DB7C2DCC63A93E23CFE 1585
108 0538023E0113293E2CDE0 978
109 C73A92E2C6023292E2CD 1456
110 10C8CDF5C7010500CD3D 1137
111 1FC3DCC63A94E2B7C03C 1511
112 3294E23E053295E2C93A 1175
113 95E23D3295E2B7200932 1135
114 94E2CD1EC8C3CC63A91 1609
115 E2C6023291E2CD10C8CD 1473
116 F5C7010500CD3D1FC3CC 1146
117 C6C93E053293E2C93A93 1295
118 E221A2C647232310FC7E 1154
119 320A5B237E320B5B5C93A 723
120 92E232045B3A91E23205 1001
121 5BCD00FAC97C0F0F0FE6 1146
122 03F658575DC9F321B888 1314
123 11004001001BEDB0FBC9 974
124 3A92E24F3A91E2D61847 1247
125 C5CDAA22CD05C8C11AFE 1489
126 28C8FE16C8FE432008CD 1282
127 CE22CDD52DB7C03A92E2 1508
128 C60F4F3A91E2D61847C5 1227
129 CDA22CD05C8C11AFE28 1332
130 C8FE16C8FE432008CDCE 1448
131 22CDD52DB7C03A91E2D6 1515
132 023291E2CD10C8CDF5C7 1493
133 010500CD3D1F189E3A90 687
134 E2FE012814FE042810FE 1109
135 07280C3D3290E23EE832 884
136 92E2C335C43A91E2FE5F 1594
137 38153A90E2D6033290E2 1142
138 3E373291E23E123292E2 1040
139 C335C43A90E2C6033290 1267
140 E23E123292E23E9F3291 1144
141 E2C335C43A90E2FE0628 1398
142 10FE09280C3C3290E23E 873
143 0A3292E2C335C43A91E2 1305
144 FE5F38133E573291E23E 1056
145 DE3292E2CD10C8CDF5C7 1714
146 C3CCC63E7F3291E23EDE 1491
147 3292E2CD10C8CDF5C7C3 1687
148 CCC60000000000000000 402
    
```

3870-3900: Si se ha pulsado la X se salta a PULDE.
 3910-3940: Si se ha pulsado Space se llama a PSALT.
 3950-3980: Si se ha pulsado ENTER vuelve al BASIC.
 3990: Carga en A el valor de SALT.
 4000-4010: Si no es 0 es que el muñeco ha de estar subiendo porque se ha pulsado la tecla de salto y pasa a SUBIR.

4020: Si es 0 hay que comprobar si tiene que bajar porque no hay suelo debajo suyo, y se hace una llamada a BAJAR.

4030: Cierra el bucle sobre TECLAS.

PULIZ. Aquí se llega si se ha pulsado para mover a la izquierda.

4060-4120: Se sacan las coordenadas del muñeco y se actualizan para comprobar qué hay a su izquierda. Se guardan las nuevas coordenadas en BC y éste en la pila.

4130: Se llama a una rutina de la ROM. Esta rutina necesita las coordenadas en el registro BC y a la salida da en HL la dirección del punto que señalan las coordenadas.

LISTADO 3 (IMPRIMIR)

DUMP: 50000 N.º BYTES: 238

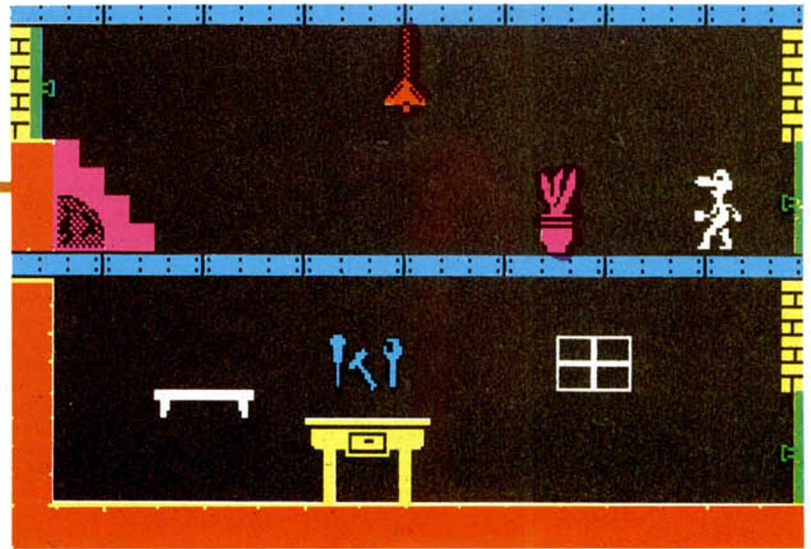
LÍNEA	DATOS	CONTROL
1	F3DD21045BED4B045BCD	1204
2	AA22320E5BDD340A0100	643
3	01DD86033D38050CD608	715
4	18F9ED430C5B5D7C0F0F	927
5	0FE603F65857D5ED5B0A	1220
6	5BEB3A085B08D53A075B	860
7	32105B3A0E5B320F5BDD	697
8	CB024620080600D360D	695
9	FA18064623DD350D080E	607
10	081ADD350B2807070DD	889
11	350B20F9C82017DD350C	714
12	2817DD350D20064623DD	431
13	360D080D20EA12131A0E	536
14	0818E318B00D2804070D	871
15	20FC12D108FE01283108	1165
16	E5EB7CE607FE07280324	1383
17	181E7DE6E0FEE02808DD	830
18	340911E006A7ED52180C	843
19	7CFE572807DD34091120	1019
20	0019EBE1083D20BFE111	641
21	2000DDCB024620053A12	565
22	5B18077E32125B3A095B	1006
23	DD4E09DD4608E5772310	1242
24	FCE1190D20F3FBC90000	

4140: Se llama a BATTRIB. Esta rutina necesita tener en HL una dirección del archivo de pantalla y a la salida devuelve la dirección de atributos en el registro DE de la posición apuntada por HL.

4160: Carga en A el contenido de DE, es decir, el color del carácter que hay a la izquierda del muñeco.

4170-4180: Si este valor es 68, es que hay una puerta. Entonces se salta a PUEIZ.

4180-4200: Si es 22 es que hay un muro, en-



01	00007FFE781E7BDE7A5E	1092
02	7BDE700E700E700E700E	849
03	700E700EFFDEDEDEDEDE	1617
04	DEDEDEDEDEDEDEDEDEDE	2220
05	DEDEDEDEDEDE2D2D2D2C	2154
06	CCCCCDEDEDEDEDEDEDE	2211
07	EDEDEDEDEDEDEDEDEDE	2370
08	EDEDEDEDEDEDE2D2D2D	1794
09	20CCCCCCEDEDEDEDE00	1809
10	FFFFFFFFFFFFFFFFFFFFFF	2550
11	FFFF000000003EFBEFBC	1250
12	3EF00FBC000FF0002F3F	870
13	FCF42EFFFF7401FCFF80	1803
14	3BFAFFDC3BF6FFDC03F7	1814
15	7FC02BF7BDD42BF7B1D4	1689
16	03B7CDC03B3BEDDC3B5B	1308
17	DDDC036BFD02B73FBD4	1617
18	2B77FBD4037BF7C03BB7	1432
19	EFDC3BCFD0DC0017D817	1430
20	0817D800003F7C3F7C00	829
21	007BEE7BEE0000DF7DDF	1293
22	7D0000FBEFFBEF000000	1105
23	01C003E07A60FFF07FF0	1500
24	01F007C0038001800180	829
25	03F002C8E6E4F9EEE7E6	1851
26	03C007800E601EE01860	814
27	1870383078F000000001	601
28	C003E07A60FFF07FF001	1500
29	F007C003800180018003	831
30	C00638EBE4F3CEE3E603	1626
31	E000E007600F600C600C	782
32	3838187880000000F0F	406
33	0F0F0F0F0F0F0F0F0F	150
34	0F0F0F0F0F0F0F0F0F	1014
35	0F0F0F0F0F0F0F0F0F	150
36	00FF000000FF000000FF	765
37	000000FF000000FF0000	510
38	00FF000000FF000000FF	765
39	000000FFFFFF0000FFFF	1020
40	00FFFF0000FFFFFF0000	1275
41	FF0000FFFFFF00000000	1275
42	00FFFF000083FFFF00D4	1363
43	FFFF00AA3FFF00C55FFF	1545
44	00A0A7FF00D513FF00A2	1231
45	8BF00CS45FF0082A1FF	1461
46	FFCS44FFFF8282AFFFD1	2009
47	14FFFFA2AAFFFC555FF	1909
48	FFAAAAFFFFFFF0000000	2125
49	3E00005F0000BF00005F	443
50	0000BF00005F0000BF00	477
51	005F0000BF00005F7FF8	756
52	BF7FFC5F7FFC5F7FFC5F	1709
53	7FFC5F80005FAAA8BFD5	1535
54	545FFFFC5F7FFC5F7FFC	1986
55	BFFFFC30003078007878	1154
56	00780000001C001C001C	204
57	001C001C001C00180018	132
58	0018001800180017F817	366
59	F818001FF81818181818	671
60	18100810081008100810	136
61	080001C003E07A60FFF0	1141
62	7FF001F007C003800180	1067
63	01800340034007600760	469
64	06600240024003C001C0	622
65	01800180038007800000	524
66	00000000000000000000	0

FORMA y se incrementa.
 4290-4300: Si es mayor que mueve se llama a CAMB1.
 4310-4320: Si es menor que 5 se llama a CAMB1.

- 4330-4340: Se almacena la nueva forma y se llama a BFORM.
- 4350-4370: Decrementa la coordenada X del muñeco y lo almacena de nuevo en COORX.
- 4380: Se llama a REST que imprime la pantalla que había sido almacenada.
- 4390: Se llama a PCOOR.

LISTADO 5 (DATOS)

DUMP: 45000
 N.º BYTES: 285

1	04142F2806142F500513	288
2	5F7804104FB007050F4F	596
3	3006135F600216376007	451
4	1D4F88031E4FC0040521	590
5	372805185F5003203748	461
6	07173F80040737A80208	465
7	1A9F480317879806104F	684
8	30051D3F30051D2F3005	327
9	135F80041B3778021A57	563
10	68070C127F20052A7F48	626
11	0718A770030C7F680607	569
12	7FA0071A94C8041E4F38	837
13	07293748022447700623	437
14	376806063790021E4FA8	649
15	0709297F480518A77804	576
16	207F7006217FA0021787	757
17	C8071A57200712374805	501
18	135F68031C3788060613	471
19	A77802127FA8030D3F30	729
20	07114F68050C37600608	392
21	4FB0070A127F480218A7	682
22	780316877007127FA002	706
23	19373805193748061937	379
24	5806135F7804154FB807	623
25	0D3FB0030A0F97300629	526
26	7FA80218A778040C7F68	759
27	07067F98021097B0060B	654
28	4F30071C375804135F78	543
29	06142F80030000000000	204

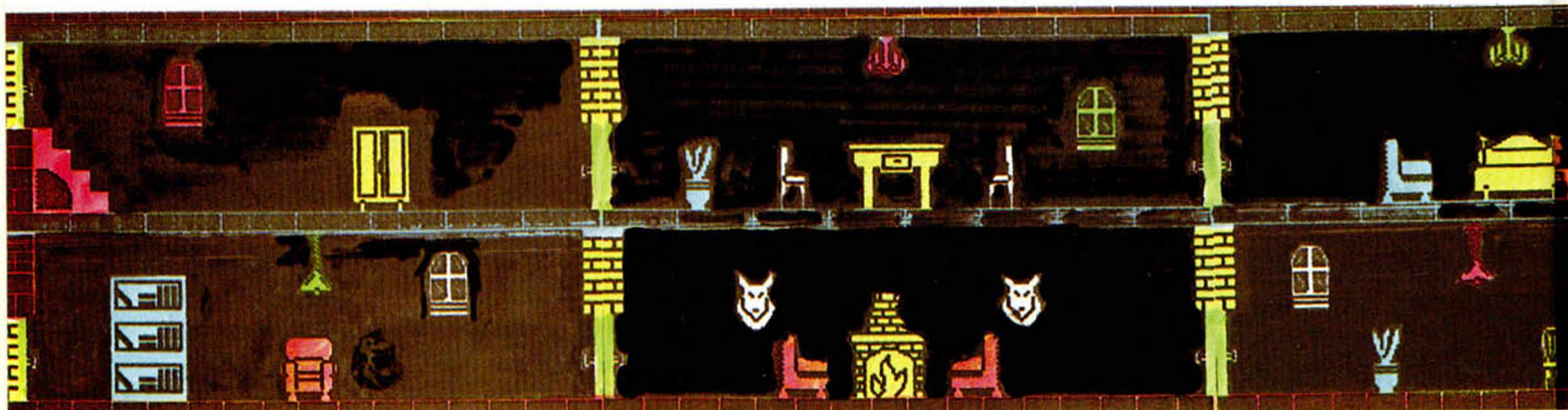
LISTADO 6 (ALMACÉN)

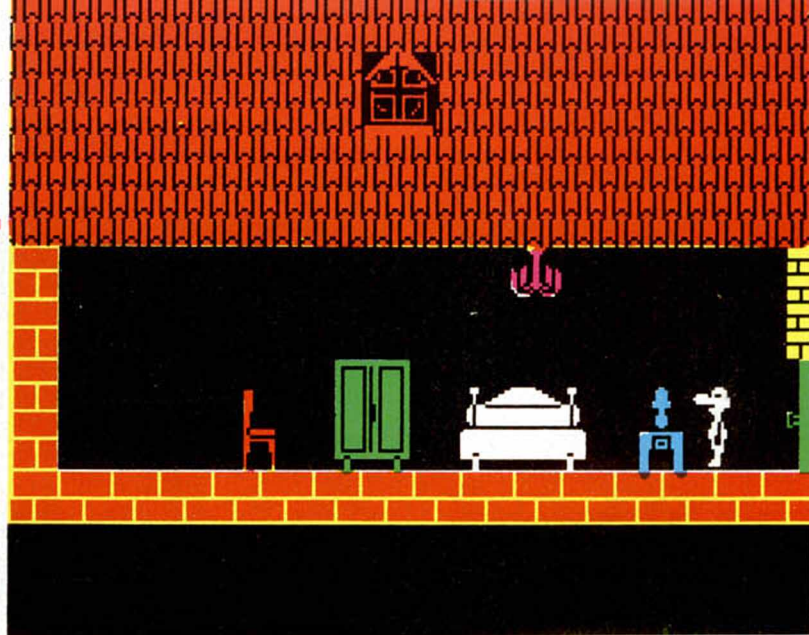
```

9000 INPUT "cargar grafico en GO
U (s/n) ?";a$
9010 IF a$="s" THEN LOAD ""CODE
9020 INPUT "caracteres de alto ?";al
      INPUT "caracteres de ancho
      ?";an
9024 LET d=65368: LET d2=d
9030 INPUT "direccion donde lo v
as almacenar";de: LET dir=de
9040 FOR e=1 TO al*8
9050 FOR g=d TO d+an*8-1 STEP 8
9060 POKE de,PEEK g
9070 LET de=de+1
9080 NEXT g
9090 IF e/8=INT (e/8) THEN LET d
=de-7+(8*an): LET d2=d: GO TO 912
9110 LET d=d+1: LET d2=d
9120 NEXT e
9125 PRINT AT 10,10;"COMIENZO :
";dir;AT 12,10;"LONGITUD : ";an
*al*8
9130 INPUT "QUIERES GRABARLO ? "
;a$
9140 IF a$="s" THEN GO SUB 9200
9145 IF a$="n" THEN GO SUB 9220
9150 INPUT "continuar ?";a$
9160 IF a$="s" THEN RUN
9170 STOP "nombre ?";n$: INPUT
"COMIENZO ?";c: INPUT "LONGUIT
UD ?";l
9210 SAVE n$CODE c,l;"BYTE BAJO :
9220 PRINT AT 14,10;"BYTE BAJO :
";dir-256*INT (dir/256)
9230 PRINT AT 16,10;"BYTE ALTO :
";INT (dir/256)
9240 RETURN
  
```

- 4400-4410: Se hace una pausa llamando a una rutina de la ROM y con el valor de espera en BC.
- 4420: Se salta a VOLV.
- 4450-4510: Se calculan las coordenadas que hay a la derecha del muñeco.
- 4520-4530: Se halla el color que hay en esas coordenadas.

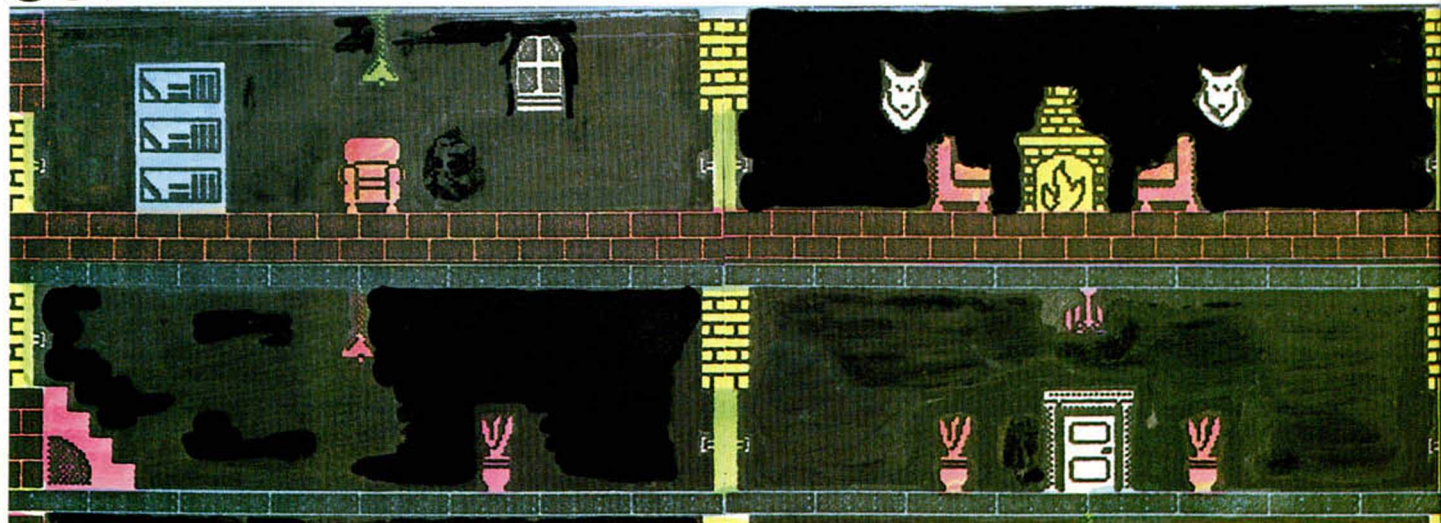
- 4560-4570: Si hay una puerta salta a PUEDE.
- 4580-4590: Si hay un muro salta a VOLV.
- 4600-4610: Si no hay una escalera salta a CONT9.
- 4620-4650: Si ha chocado con una escalera se salta a VOLV.
- 4660-4690: Si el valor de FORMA es menor que 5 salta a CONT6.
- 4700-4710: Se carga 1 en A y se almacena en FORMA.
- 4720: Se llama a BFORM.
- 4730-4750: Se incrementa la coordenada X en dos y se almacena.
- 4760: Imprime la pantalla.
- 4770: Se llama a PCOOR.
- 4780-4790: Se realiza una pausa.
- 4800: Salto a VOLV.
- 4830-4850: Aquí se llega si se ha pulsado la tecla de salto. Si el muñeco ya estaba saltando, SALTO contendrá un 1 y se retorna.
- 4860-4870: Pone SALTO a 1.
- 4880-4900: Se carga en CONT el número de veces que habrá que incrementar la coordenada Y del muñeco. Se retorna.
- SUBIR: Aquí se llega si SALTO contiene 1.
- 4930-4950: Decrementa el contador de veces que hay que hacer subir al muñeco.
- 4960-4970: Si no es 0 salta a CONT7.
- 4980-4990: Si no es 0 se pone SALTO a 0 y se llama a BAJAR.
- 5000: Se vuelve a TECLAS.
- 5010-5030: Incrementa la coordenada Y del muñeco y se almacena.
- 5040: Recupera la pantalla.
- 5050: Llama a PCOOR.
- 5060-5070: Hace una pausa.
- 5080: Vuelve a teclas.
- 5120-5140: Pone FORMA a 5 y retorna.
- 5160-5210: Halla la dirección donde se encuentra el gráfico de la forma del muñeco que hay que imprimir.
- 5220-5270: La almaceña en las etiquetas que utiliza la rutina de impresión y retorna.
- 5290-5340: Coloca las coordenadas del muñeco en las etiquetas que utiliza la rutina de impresión, lo imprime y retorna.
- 5360-5440: Halla la dirección del archivo de atributos de una dirección del archivo de pantalla.
- 5460-5580: Calcula las coordenadas que hay debajo del muñeco en su parte izquierda.
- 5590-5620: Halla el color de estas coordenadas.
- 5630-5670: Si es un techo o un suelo se retorna, ya que el muñeco no debe caer.
- 5680-5690: Si no es el color de una escalera salta a CONT12.
- 5700-5730: Si está tocando la escalera retorna.
- 5740-5940: Realiza lo mismo pero para las coordenadas que hay debajo del muñeco y a su derecha.
- 5950-5970: Decrementa la coordenada Y y se almacena en COORY.
- 5980-6020: Imprime pantalla, llama a PCOOR, hace una pausa y cierra el bucle sobre BAJAR.
- 6050-6110: Si está en la pantalla 1, 407 salta a CONT10.
- 6120-6130: Decrementa el número de pantallas





- y lo almacena en PANT.
- 6140-6160: Actualiza las coordenadas del muñeco y se salta a EMP.
- 6170-6190: Si el muñeco está en el piso de abajo salta a CONT11.
- 6200-6220: Disminuye el número de pantalla en 3.
- 6230-6270: Actualiza las coordenadas del muñeco y salta a EMP.
- 6280-6350: Aumenta en 3 el número de pantalla, se actualizan las coordenadas del muñeco y salta a EMP.
- 6380-6420: Si está en la pantalla 6 ó 9 salta a CONT13.
- 6430-6470: Aumenta el número de pantalla, se actualizan las coordenadas del muñeco y se vuelve a EMP.
- 6480-6500: Si el muñeco está en el piso de abajo salta a CONT5.
- 6510-6570: Actualiza las coordenadas, recupera la pantalla, llama a PCOOR y se vuelve a TECLAS.
- 6580-6640: Actualiza las coordenadas, llama a REST, llama a PCOOR y vuelve a TECLAS.

10	*****	510	DEFB 16,16,199,220	1010	OUT (254),A
20	*	520	DEFB 16,24,232,220	1020	LD (MOD0),A
30	* RUTINA DE MAPEADO *	530	DEFB 16,24,25,221	1030	LD HL,SUEL
40	*	540	DEFB 24,24,74,221	1040	LD DE,SUELO
50	*****	550	DEFB 16,24,147,221	1050	CALL SITUA
60		560	DEFB 32,16,196,221	1060	LD B,8
70	ORG 50000	570	DEFB 16,24,5,222	1070	REP1 PUSH BC
80	ENT 50000	580	VENTAN DEFB 24,24,54,222	1080	CALL IMPRI
90		590	DEFB 40,24,127,222	1090	POP BC
100	JP SEGUIR	600	DEFB 16,24,248,222	1100	LD A,(POSX)
110		610	TEJAD DEFB 32,16,41,223	1110	ADD A,32
120	IMPRI EQU 64000	620	DEFB 32,24,186,223	1120	LD (POSX),A
130	PANT EQU 50000	630	DEFB 16,16,283,223	1130	DJNZ REP1
140	COORY EQU 50001	640	DEFB 16,24,236,223	1140	LD HL,TECH
150	COORX EQU 50002	650	DEFB 16,24,29,224	1150	LD DE,TECHO
160	FORMA EQU 50003	660	PUERTA DEFB 8,32,78,224	1160	CALL SITUA
170	SALTO EQU 50004	670	ESCAL DEFB 32,32,111,224	1170	CALL SUB1
180	CONT EQU 50005	680	DEFB 24,24,240,224	1180	LD A,175
190	POSX EQU 23300	690	DEFB 16,24,57,225	1190	LD (POSY),A
200	POSY EQU 23301	700	DEFB 16,24,186,225	1200	CALL SUB1
210	MOD0 EQU 23302	710		1210	JR CONT1
220	ANCHO EQU 23303	720	DPAN DEFB 284,175	1220	LD B,8
230	ALTO EQU 23304	730	DEFB 221,175	1230	REP2 PUSH BC
240	COLOR EQU 23305	740	DEFB 242,175	1240	CALL IMPRI
250	INFER EQU 23306	750	DEFB 7,176	1250	POP BC
260	SUPER EQU 23307	760	DEFB 40,176	1260	LD A,(POSX)
270		770	DEFB 89,176	1270	ADD A,32
280	TABLA DEFB 16,24,64,215	780	DEFB 126,176	1280	LD (POSX),A
290	DEFB 16,24,113,215	790	DEFB 151,176	1290	DJNZ REP2
300	DEFB 16,24,162,215	800	DEFB 192,176	1300	RET
310	PUERT2 DEFB 8,32,211,215	810		1310	CONT1 LD A,(PANT)
320	ESCAL2 DEFB 32,32,244,215	820		1320	CP 3
330	DEFB 24,24,117,216	830	SEGUIR LD A,2	1330	JR 2,MURO2
340	DEFB 16,24,190,216	840	CALL #1601	1340	CP 6
350	TECHO DEFB 32,8,239,216	850	LD A,7	1350	JR 2,MURO2
360	MURO DEFB 16,32,16,217	860	LD (23693),A	1360	CP 9
370	SUELO DEFB 32,16,81,217	870	LD (23624),A	1370	JR 2,MURO2
380	DEFB 24,16,146,217	880	LD A,96	1380	LD HL,PAR1
390	DEFB 40,24,195,217	890	LD (COORX),A	1390	LD DE,PARED
400	DEFB 32,8,60,218	900	LD A,127	1400	CALL SITUA
410	PARED DEFB 8,32,93,218	910	LD (COORY),A	1410	LD HL,PUE1
420	DEFB 24,16,126,218	920	XOR A	1420	LD DE,PUERTA
430	DEFB 24,16,175,218	930	LD (SALTO),A	1430	CALL SITUA
440	DEFB 24,16,224,218	940	INC A	1440	LD A,(PANT)
450	DEFB 16,24,17,219	950	LD (FORMA),A	1450	CP 3
460	DEFB 16,24,66,219	960	LD A,8	1460	JR C,CONT2
470	DEFB 32,16,115,219	970	LD (PANT),A	1470	LD HL,PUE2
480	DEFB 24,16,180,219	980		1480	LD DE,PUERTA
490	DEFB 32,32,229,219	990	EMP CALL 3435	1490	CALL SITUA
500	DEFB 24,32,182,220	1000	LD A,1	1500	LD HL,PAR2



1510	LD DE,PARED	2010	JR TEJADO	2510	CP 4	3010	POP BC	3510	PUE6	DEFB 167,8,68
1520	CALL SITUA	2020	MUR LD A,(PANT)	2520	JR Z,ESC	3020	DJNZ OTRA	3520	MUR2	DEFB 135,8,22
1530	JR CONT2	2030	CP 2	2530	CP 7	3030	LD HL,16384	3530	MUR3	DEFB 63,240,22
1540	MUR02 LD HL,MUR3	2040	JR Z,TEJADO	2540	JR Z,ESC	3040	LD DE,35000	3540	PUE7	DEFB 95,240,68
1550	LD DE,MURO	2050	LD HL,MUR1	2550	JR CONT4	3050	LD BC,6912	3550	MUR4	DEFB 167,240,22
1560	CALL SITUA	2060	LD DE,MURO	2560	ESC LD HL,ESC1	3060	LDIR	3560	PUE8	DEFB 135,240,68
1570	LD A,(PANT)	2070	CALL SITUA	2570	LD DE,ESCAL	3070	JP SEGU2	3570	PAR6	DEFB 135,240,6
1580	CP 3	2080	LD HL,PAR5	2580	CALL SITUA	3080		3580	MUR5	DEFB 95,240,22
1590	JR Z,TRES	2090	LD DE,PARED	2590	CONT4 LD A,(PANT)	3090	SITUA DI	3590	SUEL	DEFB 31,8,22
1600	LD HL,PUE7	2100	CALL SITUA	2600	CP 6	3100	LD A,(DE)	3600	TECH	DEFB 103,8,40
1610	LD DE,PUERTA	2110	LD HL,PUE5	2610	JR Z,ESCB	3110	LD (ANCHO),A	3610	MUR6	DEFB 63,8,22
1620	CALL SITUA	2120	LD DE,PUERT2	2620	CP 9	3120	INC DE	3620		
1630	LD HL,PAR1	2130	CALL SITUA	2630	JR Z,ESCB	3130	LD A,(DE)	3630	FIGURA	DEFB 64,215
1640	LD DE,PARED	2140	LD A,(PANT)	2640	JR CONT5	3140	LD (ALTO),A	3640		DEFB 113,215
1650	CALL SITUA	2150	CP 1	2650	ESCB LD HL,ESC2	3150	INC DE	3650		DEFB 64,215
1660	LD HL,PUE8	2160	JR Z,TEJADO	2660	LD DE,ESCAL2	3160	LD A,(DE)	3660		DEFB 162,215
1670	LD DE,PUERTA	2170	LD HL,PAR4	2670	CALL SITUA	3170	LD (INFER),A	3670		DEFB 106,225
1680	CALL SITUA	2180	LD DE,PARED	2680	CONT5 LD A,(PANT)	3180	INC DE	3680		DEFB 236,223
1690	LD HL,PAR6	2190	CALL SITUA	2690	CP 7	3190	LD A,(DE)	3690		DEFB 106,225
1700	LD DE,PARED	2200	LD HL,PUE6	2700	JR NZ,SIG	3200	LD (SUPER),A	3700		DEFB 29,224
1710	CALL SITUA	2210	LD DE,PUERT2	2710	LD HL,MUR6	3210	LD A,(HL)	3710		
1720	LD HL,MUR4	2220	CALL SITUA	2720	LD DE,MURO	3220	LD (POSY),A	3720	SEGU2	LD A,16
1730	LD DE,MURO	2230	LD HL,MUR2	2730	CALL SITUA	3230	INC HL	3730		LD (ANCHO),A
1740	CALL SITUA	2240	LD DE,MURO	2740		3240	LD A,(HL)	3740		LD A,24
1750	JR CONT2	2250	CALL SITUA	2750		3250	LD (POSX),A	3750		LD (ALTO),A
1760	TRES LD HL,MUR5	2260	TEJADO LD A,(PANT)	2760	SIG LD A,(PANT)	3260	INC HL	3760		LD A,7
1770	LD DE,MURO	2270	CP 4	2770	LD B,A	3270	LD A,(HL)	3770		LD (COLOR),A
1780	CALL SITUA	2280	JR NC,CONT3	2780	LD DE,DPAN-2	3280	LD (COLOR),A	3780		CALL BFORM
1790	CONT2 LD A,(PANT)	2290	LD DE,TEJAD	2790	BUC1 INC DE	3290	PUSH HL	3790		CALL PCOOR
1800	CP 1	2300	LD HL,TEJ	2800	INC DE	3300	CALL IMPR1	3800		CALL IMPR1
1810	JR Z,MUR	2310	CALL SITUA	2810	DJNZ BUC1	3310	POP HL	3810		
1820	CP 4	2320	LD B,5	2820	LD A,(DE)	3320	EI	3820		
1830	JR Z,MUR	2330	REP3 PUSH BC	2830	LD L,A	3330	RET	3830	TECLAS	LD A,#FE
1840	CP 7	2340	LD B,8	2840	INC DE	3340		3840		IN A,(#FE)
1850	JR Z,MUR	2350	REP4 PUSH BC	2850	LD A,(DE)	3350		3850		BIT 1,A
1860	LD HL,PAR3	2360	CALL IMPR1	2860	LD H,A	3360	PAR1	3860		JR Z,PULIZ
1870	LD DE,PARED	2370	POP BC	2870	LD A,(HL)	3370	PUE1	3870		LD A,#FE
1880	CALL SITUA	2380	LD A,(POSX)	2880	LD B,A	3380	PUE2	3880		IN A,(#FE)
1890	LD HL,PUE3	2390	ADD A,32	2890	OTRA PUSH BC	3390	PAR2	3890		BIT 2,A
1900	LD DE,PUERT2	2400	LD (POSX),A	2900	INC HL	3400	PAR3	3900		JR Z,PULDE
1910	CALL SITUA	2410	DJNZ REP4	2910	LD A,(HL)	3410	PAR4	3910	VOLV	LD A,#7F
1920	LD A,(PANT)	2420	LD A,(POSY)	2920	INC HL	3420	PUE3	3920		IN A,(#FE)
1930	CP 3	2430	SUB 16	2930	LD B,A	3430	PUE4	3930		BIT 0,A
1940	JR C,MUR	2440	LD (POSY),A	2940	LD DE,TABLA-4	3440	ESC1	3940		CALL Z,PALT
1950	LD HL,PAR4	2450	POP BC	2950	BUC2 INC DE	3450	ESC2	3950		LD A,#BF
1960	LD DE,PARED	2460	DJNZ REP3	2960	INC DE	3460	VENT	3960		IN A,(#FE)
1970	CALL SITUA	2470	LD HL,VENT	2970	INC DE	3470	TEJ	3970		BIT 0,A
1980	LD HL,PUE4	2480	LD DE,VENTAN	2980	INC DE	3480	MUR1	3980		RET Z
1990	LD DE,PUERT2	2490	CALL SITUA	2990	DJNZ BUC2	3490	PAR5	3990		LD A,(SALTO)
2000	CALL SITUA	2500	CONT3 LD A,(PANT)	3000	CALL SITUA	3500	PUE5	4000		OR A

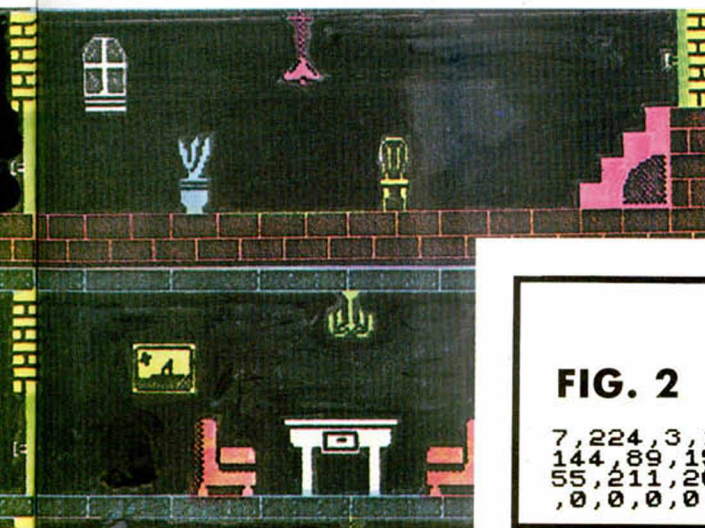
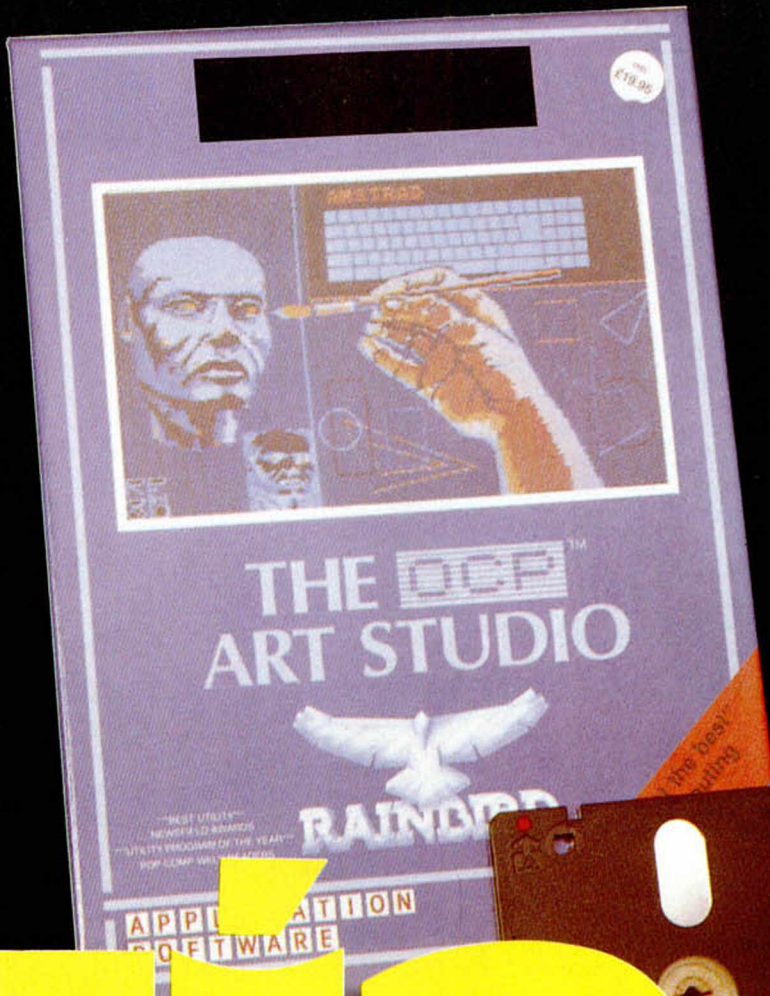


FIG. 2

7,224,3,192,1,128,1,128,1,128,9,
144,89,153,217,155,217,155,217,1,
55,211,203,205,179,66,66,60,60,0,
0,0,0,0

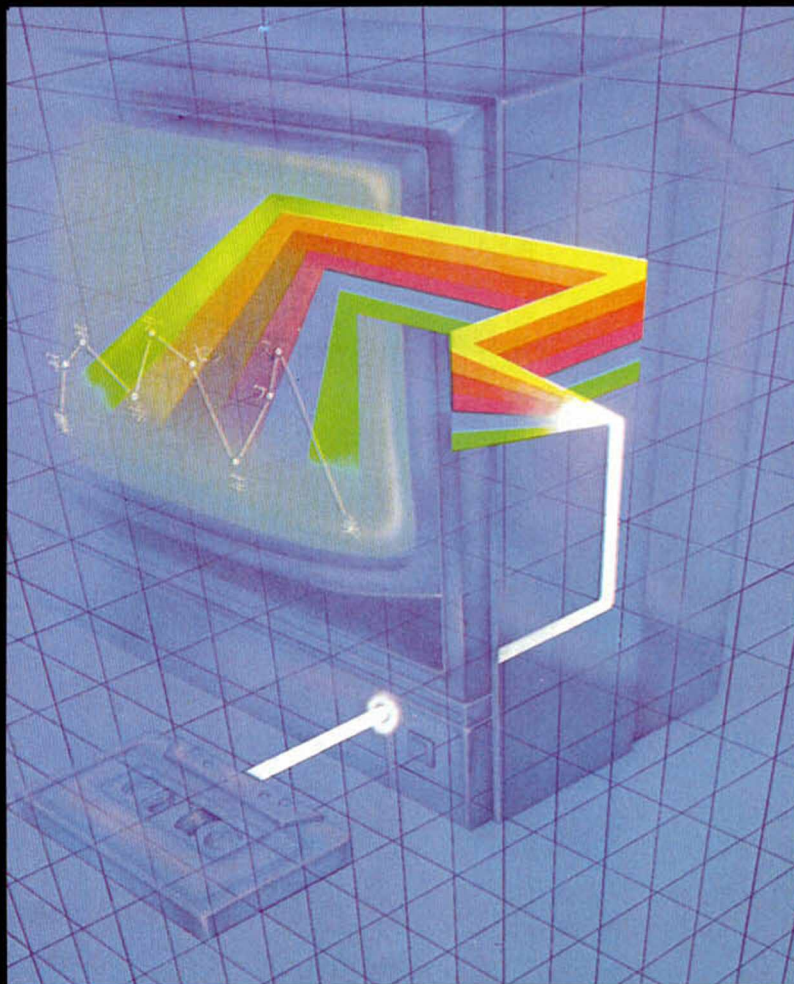
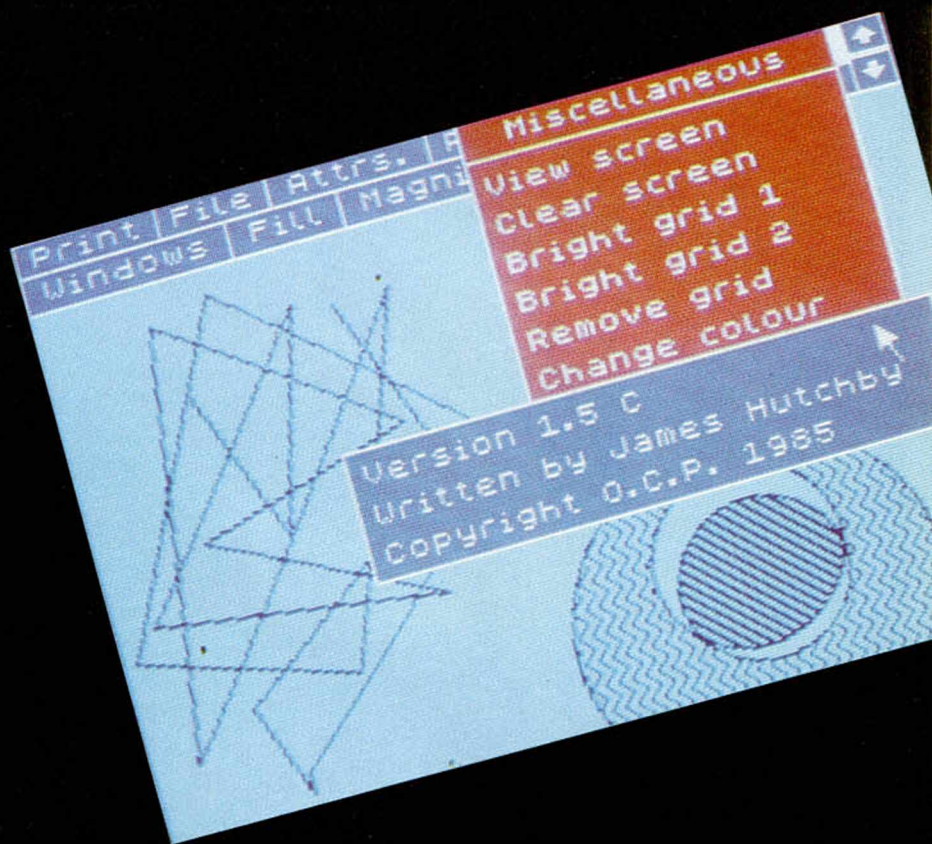
4810 JP NZ,SUBIR	4510 PUSH BC	5010 CONT7 LD A,(COORY)	5510 EI	6810 CALL #1F3D
4820 CALL Z,BAJAR	4520 CALL 8874	5020 ADD A,2	5520 RET	6820 JR BAJAR
4830 JR TECLAS	4530 CALL BATRI	5030 LD (COORY),A	5530 BAJAR LD A,(COORDX)	6830
4840	4540 POP BC	5040 CALL REST	5540 LD C,A	6840
4850	4550 LD A,(DE)	5050 CALL PCOOR	5550 LD A,(COORY)	6850 PUE12 LD A,(PANT)
4860 PUL12 LD A,(COORDX)	4560 CP 68	5060 LD BC,5	5560 SUB 24	6860 CP 1
4870 DEC A	4570 JP Z,PUEDE	5070 CALL #1F3D	5570 LD B,A	6870 JR 2,CONT10
4880 LD C,A	4580 CP 22	5080 JP TECLAS	5580 PUSH BC	6880 CP 4
4890 LD A,(COORDY)	4590 JP Z,VOLV	5090 RET	5600 CALL 8874	6890 JR 2,CONT10
4100 SUB 23	4600 CP 67	5100	5610 CALL BATRI	6100 CP 7
4110 LD B,A	4610 JR NZ,CONT9	5110	5620 POP BC	6110 JR 2,CONT10
4120 PUSH BC	4620 CALL 8910	5120 CAMB1 LD A,5	5630 LD A,(DE)	6120 DEC A
4130 CALL 8874	4630 CALL 11733	5130 LD (FORMA),A	5640 CP 40	6130 LD (PANT),A
4140 CALL BATRI	4640 OR A	5140 RET	5650 RET Z	6140 LD A,232
4150 POP BC	4650 JP NZ,VOLV	5150	5660 CP 22	6150 LD (COORDX),A
4160 LD A,(DE)	4660 CONT9 LD A,(FORMA)	5160 BFORM LD A,(FORMA)	5670 RET Z	6160 JP EMP
4170 CP 68	4670 INC A	5170 LD HL,FIGURA-2	5680 CP 67	6170 CONT10 LD A,(COORY)
4180 JP Z,PUE12	4680 CP 5	5180 LD B,A	5690 JR NZ,CONT12	6180 CP 95
4190 CP 22	4690 JR C,CONT6	5190 BUC3 INC HL	5700 CALL 8910	6190 JR C,CONT11
4200 JR Z,VOLV	4700 LD A,1	5200 INC HL	5710 CALL 11733	6200 LD A,(PANT)
4210 CP 67	4710 CONT6 LD (FORMA),A	5210 DJNZ BUC3	5720 OR A	6210 SUB 3
4220 JR NZ,CONT8	4720 CALL BFORM	5220 LD A,(HL)	5730 RET NZ	6220 LD (PANT),A
4230 CALL 8910	4730 LD A,(COORDX)	5230 LD (INFER),A	5740 CONT12 LD A,(COORDX)	6230 LD A,55
4240 CALL 11733	4740 ADD A,2	5240 INC HL	5750 ADD A,15	6240 LD (COORDY),A
4250 OR A	4750 LD (COORDX),A	5250 LD A,(HL)	5760 LD C,A	6250 LD A,18
4260 JR NZ,VOLV	4760 CALL REST	5260 LD (SUPER),A	5770 LD A,(COORY)	6260 LD (COORDX),A
4270 CONT8 LD A,(FORMA)	4770 CALL PCOOR	5270 RET	5780 SUB 24	6270 LD A,18
4280 INC A	4780 LD BC,5	5280	5790 LD B,A	6280 CONT11 LD A,(PANT)
4290 CP 9	4790 CALL #1F3D	5290 PCOOR LD A,(COORDX)	5800 PUSH BC	6290 ADD A,3
4300 CALL NC,CAMB1	4800 JP VOLV	5300 LD (POSX),A	5810 CALL 8874	6300 LD (PANT),A
4310 CP 5	4810	5310 LD A,(COORY)	5820 CALL BATRI	6310 LD A,18
4320 CALL C,CAMB1	4820	5320 LD (POSY),A	5830 POP BC	6320 LD (COORDX),A
4330 LD (FORMA),A	4830 PSALT LD A,(SALTO)	5330 CALL IMPRI	5840 LD A,(DE)	6330 LD A,159
4340 CALL BFORM	4840 OR A	5340 RET	5850 CP 40	6340 LD (COORDY),A
4350 LD A,(COORDX)	4850 RET NZ	5350	5860 RET Z	6350 JP EMP
4360 SUB 2	4860 INC A	5360 BATRI LD A,H	5870 CP 22	6360
4370 LD (COORDX),A	4870 LD (SALTO),A	5370 RRCA	5880 RET Z	6370 PUEDE LD A,(PANT)
4380 CALL REST	4880 LD A,5	5380 RRCA	5890 CP 67	6380 CP 6
4390 CALL PCOOR	4890 LD (CONT),A	5390 RRCA	5900 JR NZ,CONT14	6400 JR 2,CONT13
4400 LD BC,5	4900 RET	5400 AND 3	5910 CALL 8910	6410 CP 9
4410 CALL #1F3D	4910	5410 OR #58	5920 CALL 11733	6420 JR 2,CONT13
4420 JR VOLV	4920	5420 LD D,A	5930 OR A	6430 INC A
4430	4930 SUBIR LD A,(CONT)	5430 LD E,L	5940 RET NZ	6440 LD (PANT),A
4440	4940 DEC A	5440 RET	5950 CONT14 LD A,(COORY)	6450 LD A,10
4450 PULDE LD A,(COORDX)	4950 LD (CONT),A	5450	5960 SUB 2	6460 LD (COORDX),A
4460 ADD A,16	4960 OR A	5460 REST DI	5970 LD (COORDY),A	6470 JP EMP
4470 LD C,A	4970 JR NZ,CONT7	5470 LD HL,35000	5980 CALL REST	6480 CONT13 LD A,(COORY)
4480 LD A,(COORDY)	4980 LD (SALTO),A	5480 LD DE,16384	5990 CALL PCOOR	6490 CP 95
4490 SUB 23	4990 CALL BAJAR	5490 LD BC,6912	6000 LD BC,5	6500 JR C,CONT15
4500 LD B,A	5000 JP TECLAS	5500 LDIR		6510 LD A,87



guía de utilidades gráficas

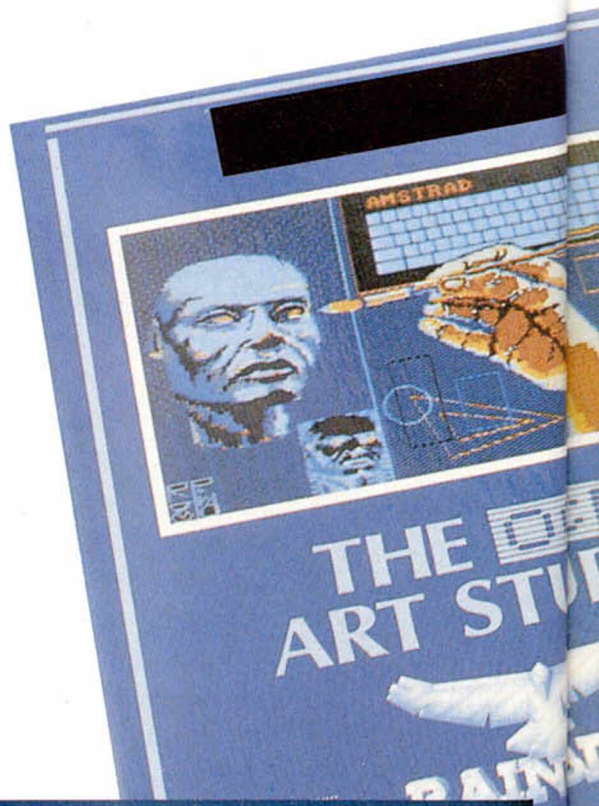
En alguna ocasión hemos comentado los diferentes programas que sobre gráficos se encuentran en el mercado. En estas páginas encontraréis, a modo de guía, el análisis de las características de cada uno. Desfilarán diseñadores gráficos, ampliaciones de memoria y ensambladores; es decir todas las herramientas que facilitarán nuestro trabajo. Por razones obvias, algunos de sobra conocidos por los usuarios de Spectrum ocuparán un lugar mínimo, dedicando mayor espacio a programas de reciente aparición que no han sido sometidos hasta ahora al minucioso examen que pasaron sus precededoras.

Hemos eliminado de esta guía aquellos programas que no han superado el nivel de calidad establecido. Creemos que esta selección beneficiará tanto a los muchos asiduos a los programas de dibujo, como a todos aquellos que se inicien en este mundo, sustituyendo el lápiz y el papel por él, sin duda, más cómodo diseño a través del ordenador.



ES

diseñadores gráficos



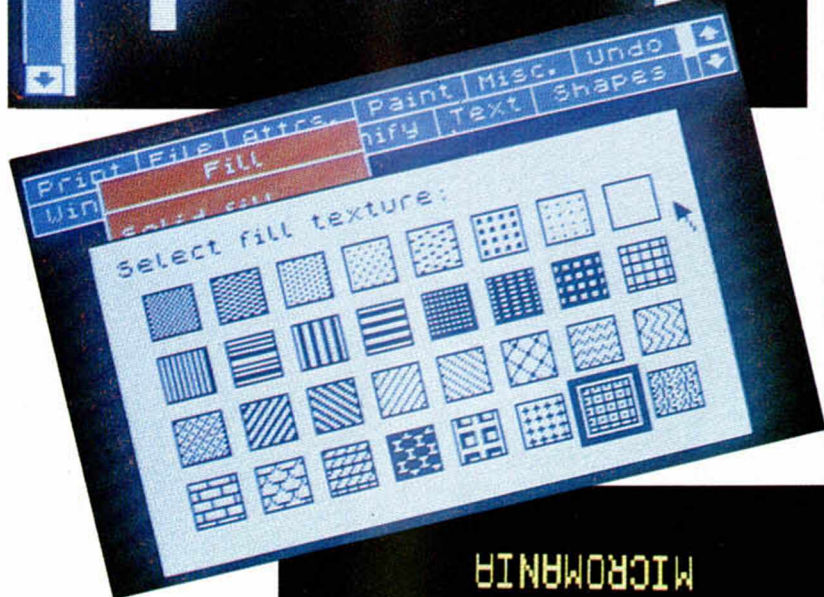
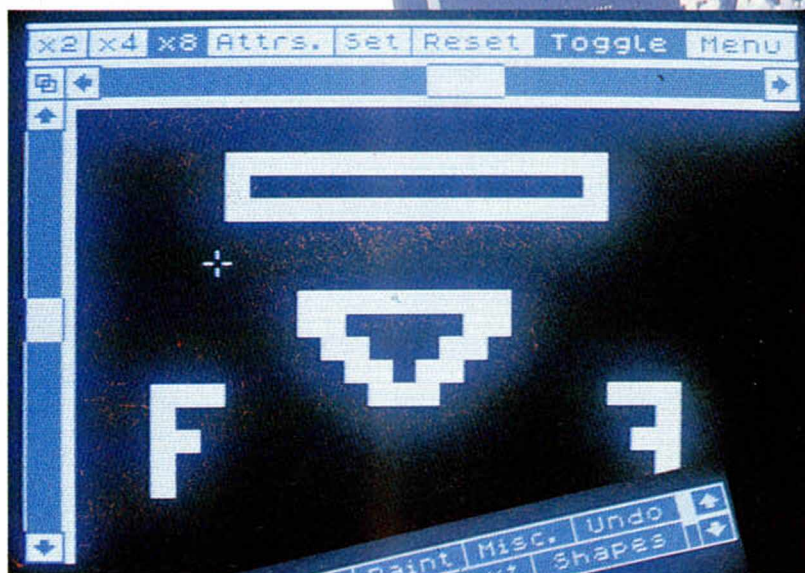
"THE ART STUDIO"

«The Art Studio» fue calificado en su momento como la mejor utilidad de dibujo para Spectrum situándola pareja al potente «MacPaint» del Macintosh. Entre sus peculiaridades destacan el sencillo manejo por iconos y la posibilidad de emplear ratón, joystick o recurrir al teclado. Además de ser el único programa con menú de impresoras.

Consta de un menú principal con once opciones diferentes, de las que se generan submenús específicos. La forma de acceder a ellos es trasladar un icono sobre la opción deseada y dejar volar nuestra imaginación combinando las amplias posibilidades de este programa de dibujo.

Las opciones del menú principal son:

PRINT, permite sacar una copia por impresora de la pantalla, eligiendo el tamaño y justificando a nuestro antojo a derecha o izquierda.



MICROMANIA
MICROMANIA
MICROMANIA
MICROMANIA



FILE, se utiliza para volcar en pantalla los gráficos salvados en cinta. Permite además de salvar, cargar y verificar archivos, o mezclar dos pantallas.

ATTRS, sirve para modificar los atributos con los que trabajemos, cambiando color de tinta, papel o borde y detallando brillo, flash, inverse, etc.

PAINT, podremos elegir para dibujar entre lápiz, spray y brocha, modificando la configuración dentro de un submenú de dieciseis especificaciones.

MISC, incluye un variado submenú formado por siete opciones que permiten desde borrar la pantalla, crear un rejilla de los atributos u observar la pantalla completa.

UNDO, elimina la última operación realizada en la pantalla.

WINDOW, permite delimitar ventanas y trabajar con ellas aprovechándonos de su utilidad, empleando las dieciseis variantes del menú.

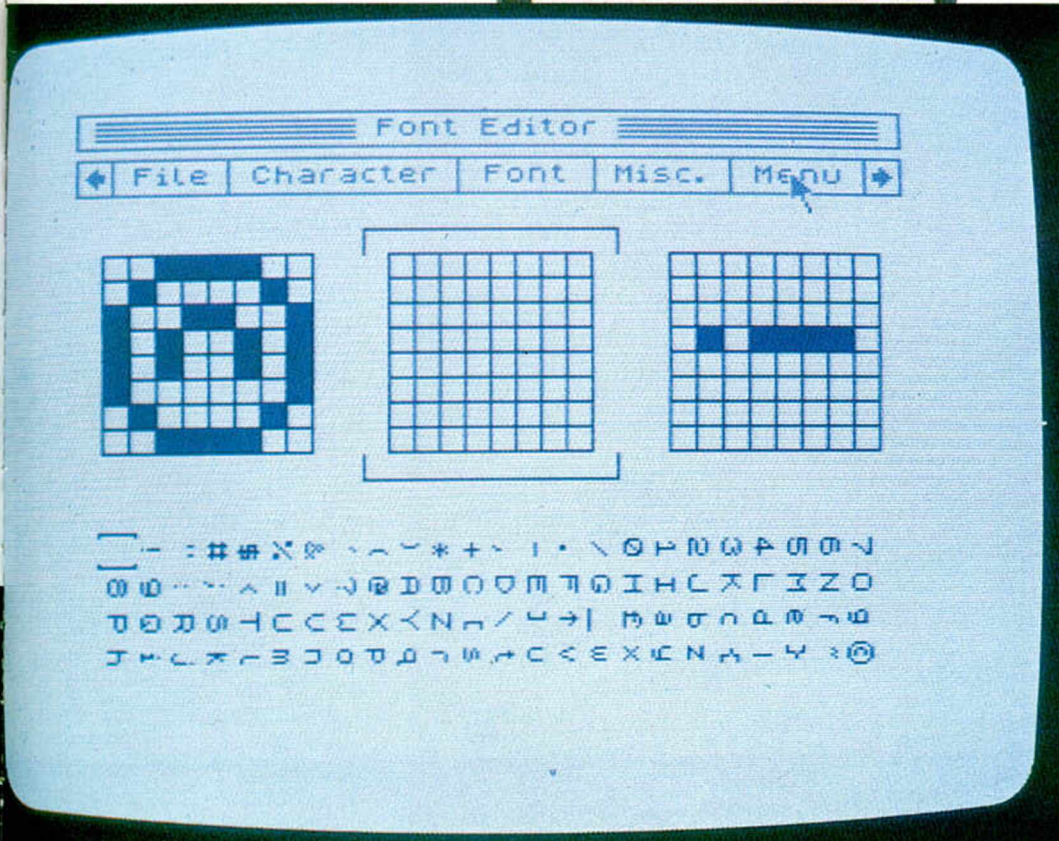
FILL, rellena con color o con cualquiera de las 64 tramas una superficie previamente delimitada.

MAGNIFY, para ampliar una zona de la pantalla entre dos y ocho veces, y trabajar en ella.

TEXT, introduce texto en pantalla; permite modificar tamaño y grosor de los caracteres.

SHAPES, sirve para crear figuras.

Como veis estamos ante un programa muy completo, de fácil manejo, que cumple a la perfección su objetivo prioritario, emular a Miguel Ángel con un sofisticado método.

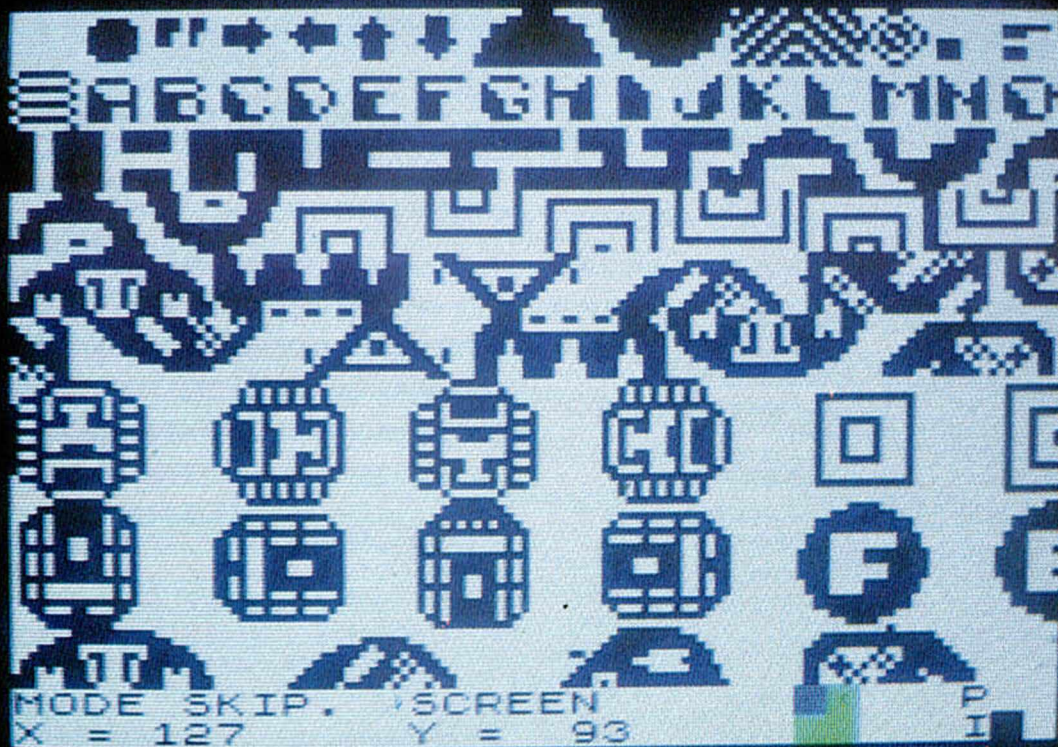


*diseñadores
gráficos*



"MELBOURNE DRAW"

Éste fue uno de los primeros programas de dibujo que llegaron para Spectrum. Introdujo en su momento interesantes novedades, como trabajar con toda la pantalla o ampliar una parte de ella. Su manejo se realiza a través del teclado o del joystick, situando el cursor y la pantalla en los diferentes modos que combinados permiten el acceso a variadas opciones. La información aparece en las dos líneas inferiores de pantalla, pudiendo ser desplazadas para poder trabajar en estas líneas. El cursor puede estar en cuatro modos: normal, dibujo, borra e invierte. A su vez el cursor puede cuatriplicarse dando como resultado cursores en alta y baja resolución, el de texto y el de scroll. Incluye también la posibilidad de ampliar un detalle de la pantalla o manejar una extensa gama de colores y atributos. Un programa de fácil manejo, con grandes posibilidades.



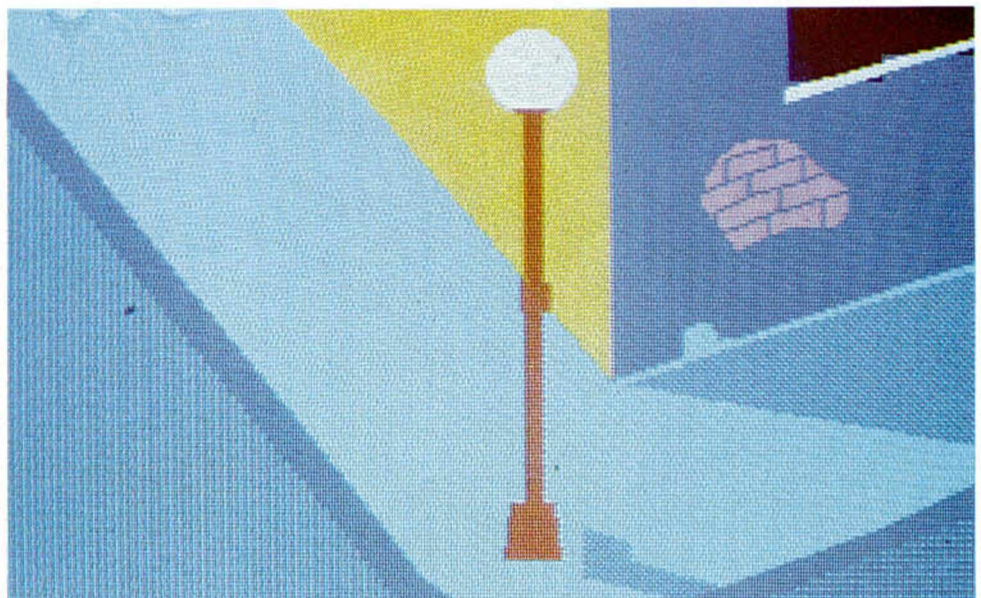
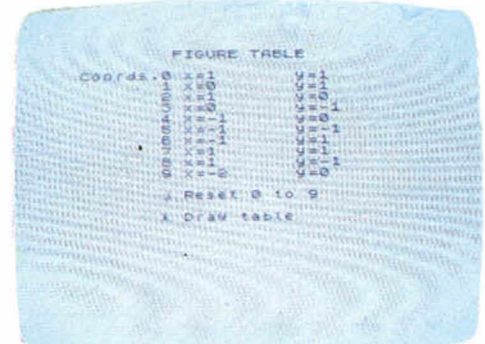


"LEONARDO"

«Leonardo» es un programa de dibujo para realizar pantallas, gráficos definidos por el usuario y elementos gráficos, variando su tamaño desde un pixel a toda la pantalla.

El programa presenta tres opciones en el menú principal: creación de dibujos, salvar y cargar gráficos. Para manejar el programa podemos emplear un joystick o el teclado. Si elegimos la opción de creación de dibujos en pantalla aparecerá el cursor que nos permitirá dibujar. Dentro de esta opción encontraremos una extensa variedad de posibilidades a las que accederemos mediante determinadas teclas. Existen dos modos de cursor: el modo pixel, moverá el cursor pixel a pixel y su tamaño es de un pixel; y el modo carácter, lo mueve de carácter a carácter y su tamaño es de 8x8 pixels. Para poder dibujar es preciso poner el cursor en modo plot. Es imprescindible en muchos momentos conocer el modo en el que nos encontramos o la localización exacta del cursor, para ello diferentes teclas nos irán suministrando la información a través de una ventana en pantalla.

«Leonardo» tiene una larga lista de comandos que permiten desde invertir, borrar, rellenar y realizar cualquier diseño geométrico a una amplia variedad de posibilidades. Un programa interesante aunque resulte en algunos momentos lioso por la gran variedad de teclas a utilizar.

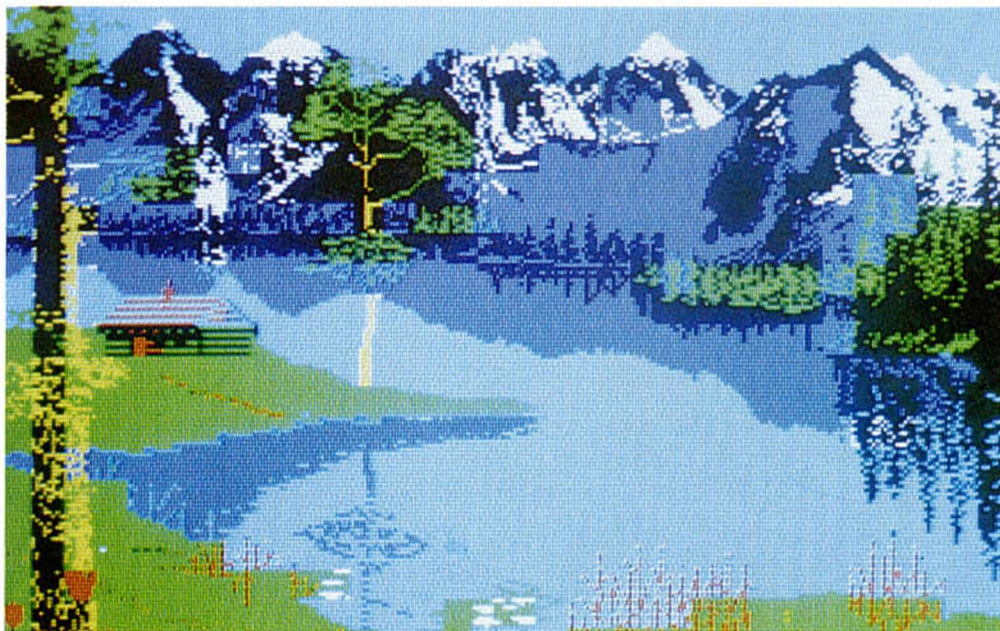


"THE ARTIST"

Este programa se maneja a través del teclado mediante el clásico sistema de menús, combinados con dos cursores que si lo deseamos podrán trabajar simultáneamente en pantalla. «The Artist» consta de tres menús que se representan en la parte inferior de la pantalla, que a su vez abarcan un amplio número de opciones.

Con el primer menú podremos invertir, desplazar o almacenar en memoria los gráficos, preseleccionando de antemano el tamaño de la brocha y la trama. Permite también borrar la última operación realizada en pantalla e insertar texto.

El segundo menú presen-



ta diez opciones diferentes; tal vez la más interesante es la de poder manipular una pantalla cargada previamente, con la consiguiente ampliación o reducción. También desde este menú

podremos crear figuras geométricas y recurrir a la opción arc para dibujar arcos que serán delimitados gracias a una tecla. Podremos rellenar también las figuras con gran rapidez.

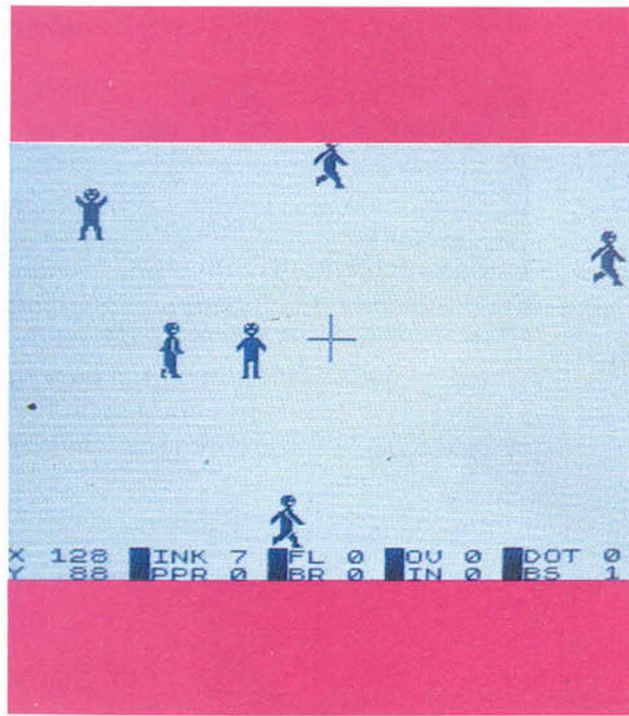
De chip a chip

“Sábado Chip”, de 17 a 19 h.

"DRAWER"

Este programa no presenta grandes diferencias con el resto de los programas analizados manejados desde un menú en la parte inferior de la pantalla. Como os indicamos el manejo es semejante, un elevado número de teclas van dando paso a las diferentes opciones de que consta.

El cursor puede estar en cuatro modos: pintar, invertir, xor y el modo desplaza, que responden al planteamiento inicial de cualquier programa de dibujo. Permite también realizar scrolls en pantalla o introducir textos. La ampliación de una parte de la pantalla es un detalle a tener en cuenta, ya que facilita considerablemente



nuestro trabajo. Al igual que la posibilidad de manejar dos parrillas guía para distinguir la posición de los caracteres. Presenta también la opción de Fill, que permite rellenar eligiendo entre diez tramas diferentes.

«Drawer» admite la posibilidad de almacenar en memoria la pantalla sobre la que trabajamos. Un dato interesante que hace de «Drawer» un buen programa, que aunque no se caracteriza precisamente por su sencillo manejo, no llega a ser excesivamente complicado, permitiendo realizar nuevas experimentaciones en el campo del diseño.

Chip Pestilo Cope

Todos los sábados, de 5 a 7 de la tarde, en "Sábado Chip". Dirigido por Antonio Rua. Presentado por José Luis Arriaza, hecho una computadora. Dedicado en cuerpo y alma al ordenador, y a la informática. Haciendo radio chip... estilo Cope.



Cadena Cope

RADIO POPULAR



... de chip a chip

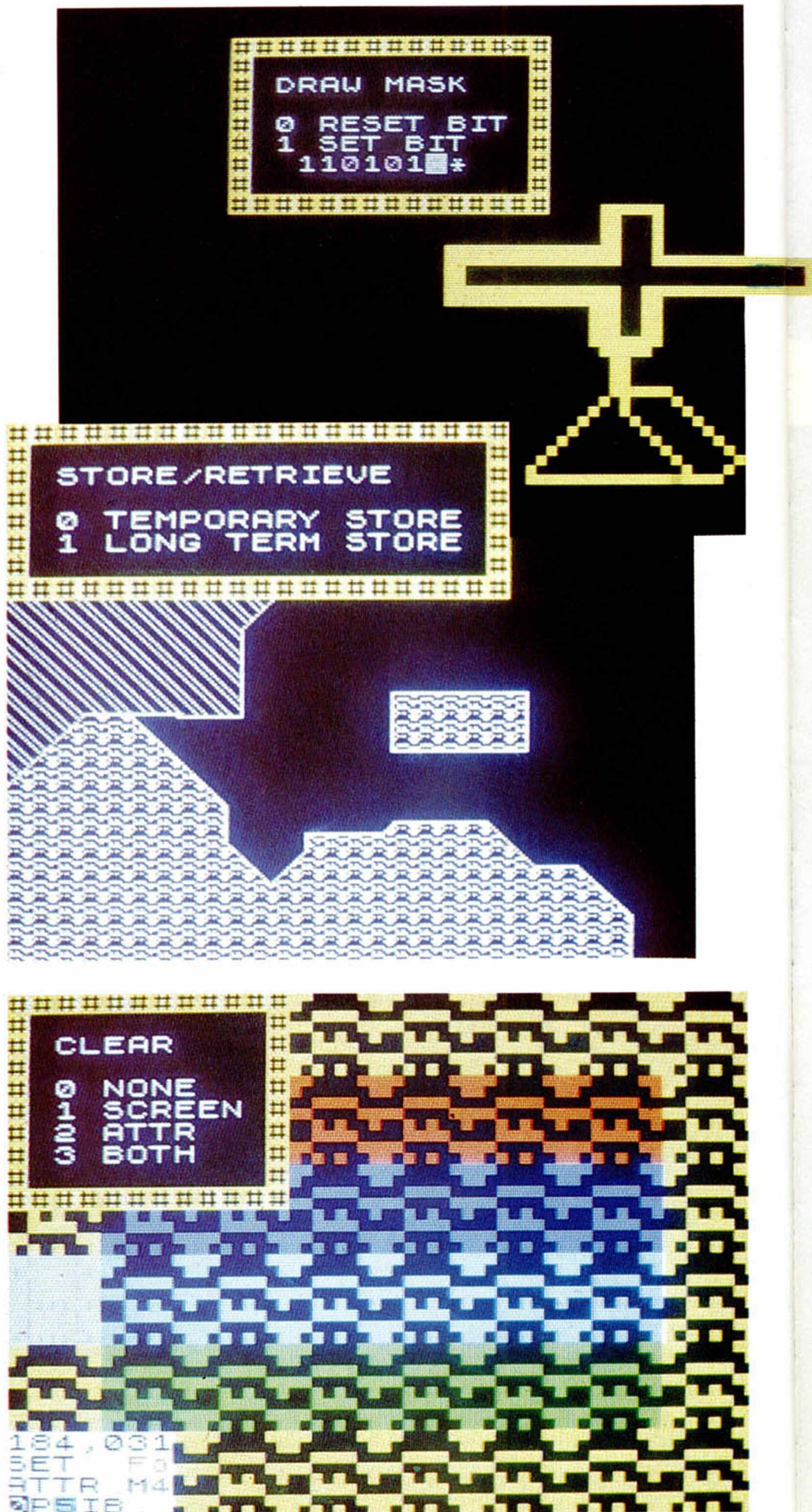
diseñadores gráficos

"PAINT PLUS"

Todos los usuarios de Spectrum interesados en alguna medida en los programas de diseño gráfico conocerán el «Paint Box», uno de los primeros programas que aparecieron. «Paint Plus», basado en este programa, perfecciona muchas de las opciones de su predecesor.

Presenta un menú con cuatro opciones. El editor de UDG posee cuatro bancos gráficos; permite manejar los gráficos realizando rotaciones, inversiones y ampliaciones de pantalla entre otras cosas. La segunda opción es el Plotter, o modo de alta resolución, permite el manejo de la pantalla punto por punto; así como salvar y cargar pantallas, borrar la última operación y tiene cinco tramas diferentes de fill. La opción Screen planer posibilita la introducción de textos y gráficos, así como cambiar los colores de los caracteres. La última opción, Organiser, se presenta en un programa adicional que sirve para almacenar un máximo de cinco pantallas y grabarlas en un fichero.

«Paint Plus» dispone de una página de ayuda que solucionará en parte el problema de memorizar un elevado número de teclas. Sin embargo no permite trabajar en las dos líneas inferiores de la pantalla, ya que no admite la posibilidad de desplazar el menú, ni almacenar figuras.



"DYNAMIC GRAPHICS"

Dynamic Graphics es un paquete de diseño formado por tres programas que te permitirán experimentar en el campo del dibujo desde tu Spectrum.

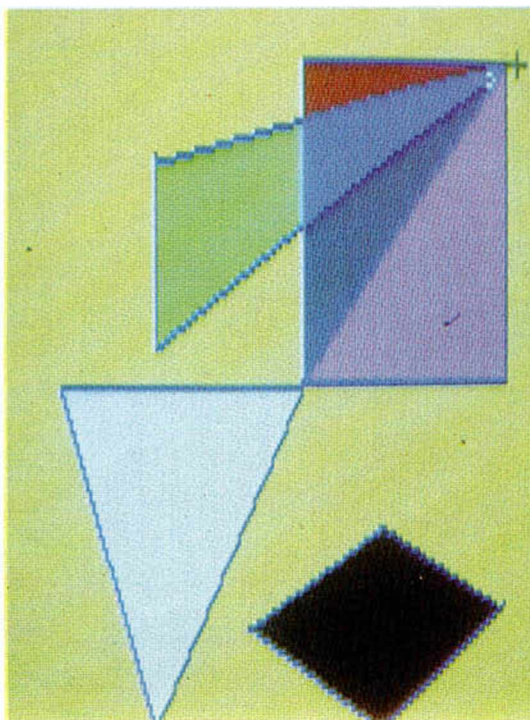
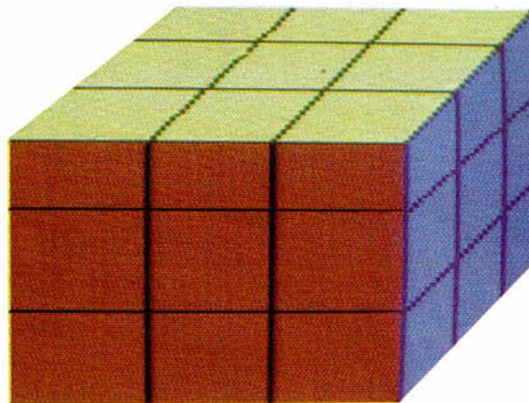
Los tres programas se complementan para dar como resultado una útil herramienta de gran calidad. El primer programa es un diseñador de sprites. El segundo permite la creación de subrutinas de dibujo para incorporarlas a los programas del usuario; y el tercero llamado «Drawmaster» que trata de la creación de dibujos.

En el primer programa encontraremos cuatro menús que permiten el acceso a más de veinticinco comandos. Las posibilidades son muchas, podremos desde borrar la ventana, invertirla, modificarla, insertar un carácter, utilizar un carácter como un UDG, dividir la ventana o grabar en cinta los dibujos.

El segundo programa dispone de cinco comandos que permiten cargar en cinta los dibujos realizados con el primer programa, grabar la subrutina, autodestruir el programa, ir al Basic y crear la subrutina.

El programa «Drawmaster», especialmente creado para dibujar, permite que el cursor esté en cuatro modos diferentes: draw, erase, over y trans. Podremos delimitar el número de pixels por el que queremos que se mueva el cursor, jugar con los comandos para modificar los atributos, rellenar con tinta una parte del dibujo y cargar y salvar en cinta las pantallas.

Un paquete interesante, en el que aunque el programa de dibujo es algo limitado, comparado con otros programas del mercado, sin embargo combinados forman una interesante utilidad gráfica.



PERIFÉRICOS

Todo diseñador gráfico debe cumplir dos objetivos. Por una parte permitir desde el ordenador la creación de diseños dependiendo sus posibilidades del software elegido, y por otro lado la medida en que facilita su realización al usuario. Esta última razón hace que muchos de los programas de dibujo admitan que a la hora de manejarse la sustitución del teclado o el joystick por periféricos especialmente creados para hacer más cómodo su uso. Sin duda son los ratones y los lápices ópticos los que se ajustan a este cometido.

AMX MOUSE y STAR MOUSE son los dos ratones compatibles con Spectrum. Permiten el manejo a través de iconos de las opciones del software que lo admitan. Su fácil manejo evita tener que recurrir constantemente a hojas de consulta para su utilización.

Los lápices ópticos permiten trabajar sobre la pantalla en lugar del papel. Destacan el lápiz óptico Inves Trónica y lápiz óptico DK Tronics.

AMPLIACIONES DEL BASIC

"BETA BASIC"

«Beta Basic» es una ampliación del Basic del Spectrum que añade considerables ventajas a la hora de programar y trabajar con el ordenador. Añade 26 nuevas instrucciones y diez funciones. Además se introducen mejoras en algunos comandos y se han añadido algunos rasgos como cursor parpadeante y un break para Código Máquina. Las nuevas instrucciones se obtienen de forma sencilla desde el modo de gráficos, sustituyendo los gráficos por los comandos.

Las instrucciones más importantes son entre otras:

ALTER, para manipular los atributos de la pantalla. Permite cambiar la pantalla a una combinación de atributos, o por ejemplo crear un gráfico utilizando tinta del mismo color que el papel y luego modificar la tinta.

AUTO, pone en funcionamiento una numeración automática de líneas.

BREAK, especial para C/M.

CLOCK, controla la operación de un reloj de 24 horas con alarma.

DEF PROC, END PROC, PROC, estos tres comandos sirven para llamar un proceso, definirlo con un nombre y terminarlo.

DELETE, borra todas las líneas de un bloque delimitado del programa.

DO y LOOP, junto con las calificaciones **whilw** y **until** proporciona una estructura de control ventajosa respecto al Basic originario del Spectrum.

DPOKE, significa doble poke, utilizando la dirección especificada y la siguiente.

EDIT, permite editar un número de línea.

GET, para leer el teclado sin usar **ENTER**, de forma similar a **INKEY \$**.

KEYWORDS, controla la utilización de las funciones de Beta Basic, como tokens o comandos introducidos carácter a carácter.

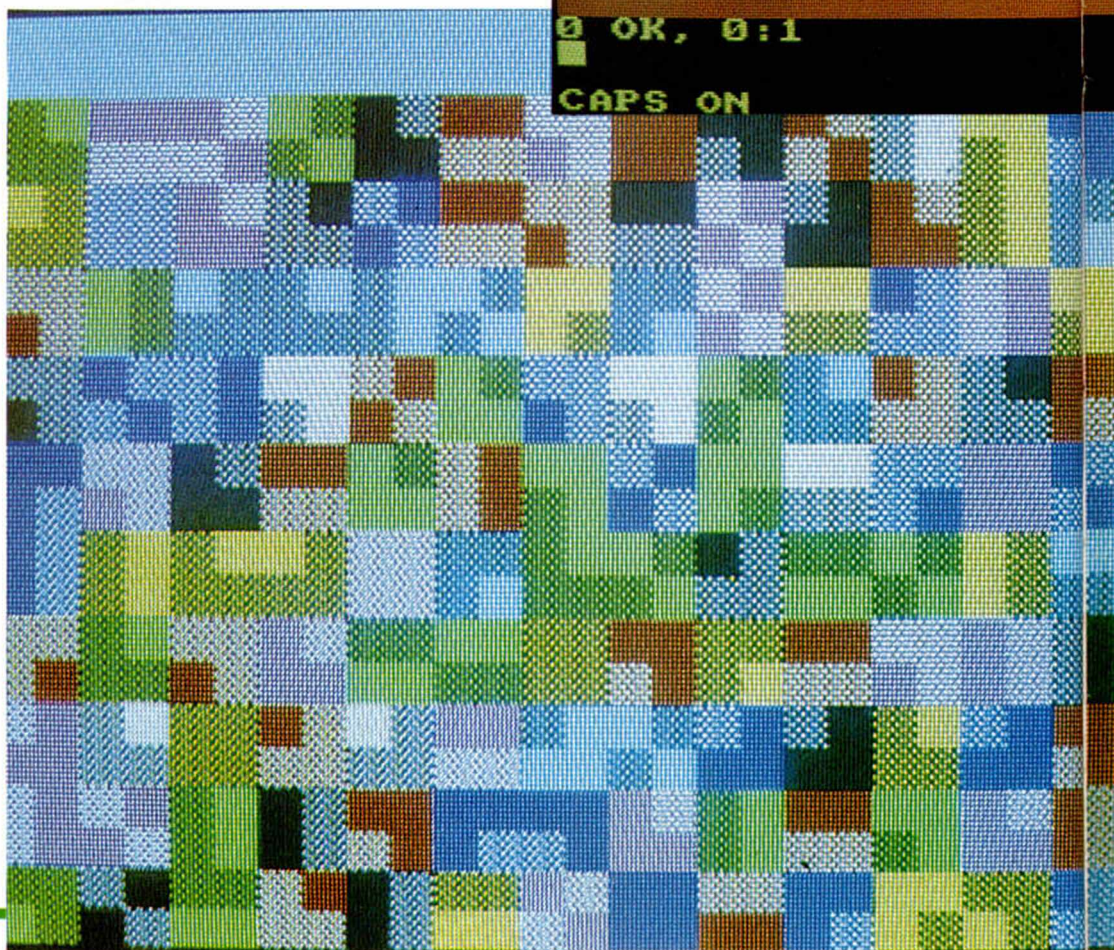
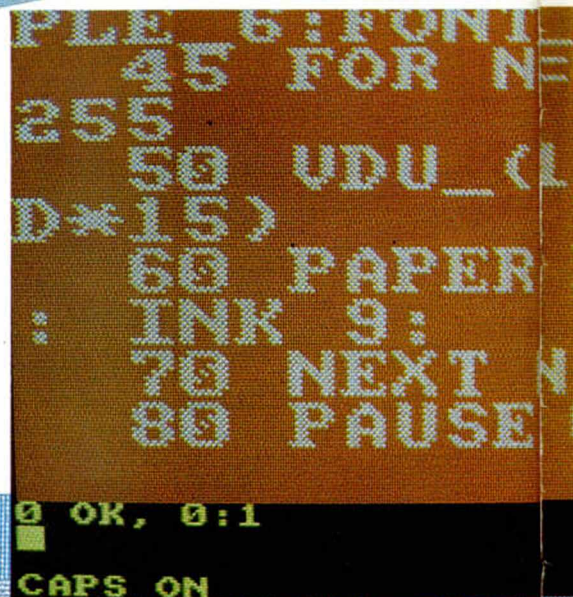
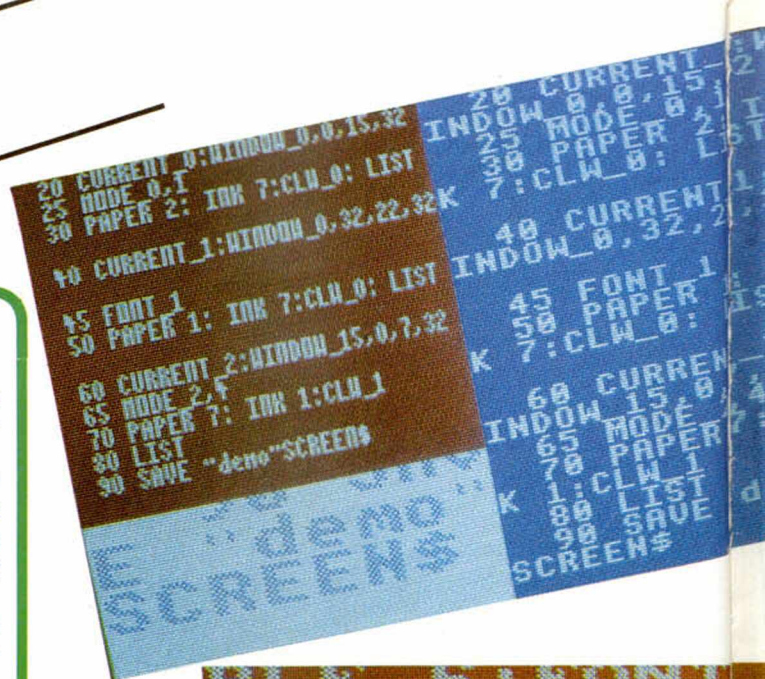
ON, permite hacer un **GOTO** o **GOSUB** a un número de línea particular, dependiendo de una situación dada.

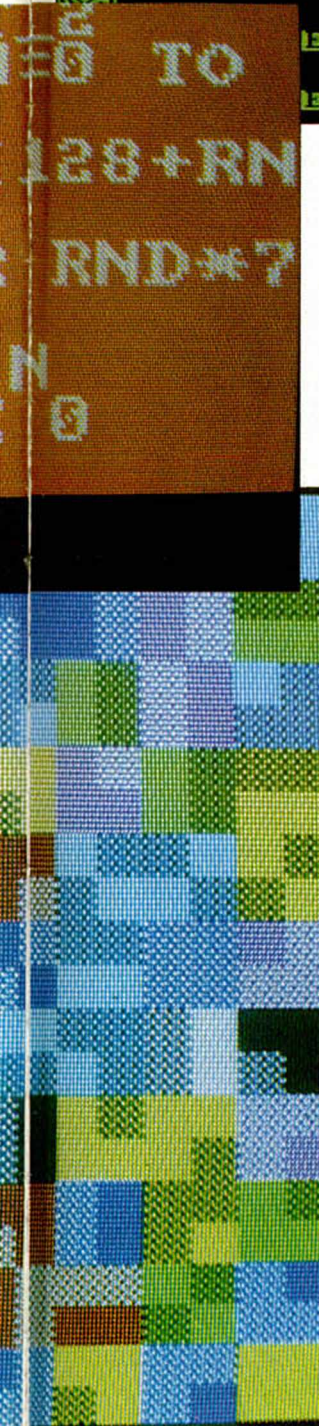
PLOT, para trazar una cadena de hasta 32 caracteres al igual que los pixels usuales.

RENUM, para renumerar las líneas de un programa.

SORT, ordena cadenas, números o letras en orden ascendente o descendente.

TRACE, permite la depuración de un programa en Basic.





E LOS REGIST

LOS REGIST

LOS REGIST

LOS REGIST

ESTADO DEL ESTADO DE LOS REGIST

ESTADO DE LOS REGIST

ESTADO DE LOS REGIST

"MEGABASIC"

Ampliar el sistema operativo de cualquier ordenador, supone que aumenten considerablemente las posibilidades de los mismos a la hora de programar. «Megabasic», como bien indica su nombre, incorpora al Basic del Spectrum nuevos comandos que se introducirán tecleando directamente el nombre; diferencia respecto a otros programas que contenían en las teclas originarias los nuevos comandos.

«Megabasic» tiene un sistema de edición muy completo que incluye la posibilidad de desplazar el cursor horizontal y verticalmente, borrar líneas, copiar algunas para transpasarlas a la línea de edición para facilitar la corrección de los programas. El aspecto que más nos interesa en el tratamiento de gráficos es la amplia gama de comandos referentes a ventanas, borrado, scrolls con diferentes venta-

nas informativas, etc. También incluye tres juegos de caracteres en cuatro tamaños distintos que hacen posible 12 tipos de letra diferentes. El tratamiento de gráficos y atributos ha sido potenciado con comandos para alterar los valores de los atributos y almacenar o modificar bloques de gráficos. También permite el llamamiento a subrutinas previamente definidas. «Megabasic» proporciona, al mismo tiempo, dos modos distintos para producir sonidos.

«Megabasic» contiene, además de la ampliación, un segundo programa denominado «Megaspectrum Sprites» especialmente diseñado para el tratamiento de gráficos en pantalla. Podrán definirse formas, colores, desplazamientos y velocidad de animación, visualizar las distintas secuencias de imágenes para la edición y corrección de las mismas.

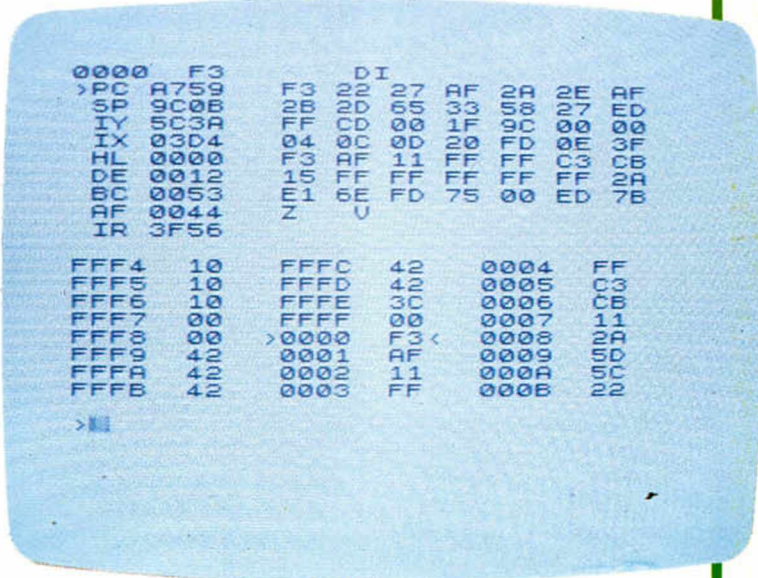
Las ventajas de esta potente utilidad son considerables dejándonos libres una vez almacenados los datos de unos 20 K de memoria.

"GENS Y MONS"

Estos dos programas de la compañía Hisoft son dos herramientas imprescindibles para todos aquellos que se inicien en el mundo de la programación.

«Gens» es un poderoso ensamblador del Z80, el microprocesador del Spectrum; su principal característica es la diferencia de manejo respecto a otros ensambladores disponibles, que le convierten en un programa profesional por su extensión y sus amplias posibilidades.

Un ensamblador, básicamente es una herramienta que permite programar en Código Máquina de forma muy sencilla, trabajando de modo complementario con el relocizable desensamblador Mons. Sustituye el árido sistema binario, por una serie de nemónicos, que constituyen el código fuente; el propio programa traslada a números este código, que tras ensamblarlo se convierte en el código



objeto. El «Gens» es también totalmente reubicable, es decir, podemos hacer que funcione instalándolo en cualquier zona de la memoria; esta es la ventaja que le sitúa por encima de otros ensambladores, ya que evita la posibilidad de que coincidan en una misma dirección el programa sobre el que trabajamos y el ensamblador o el desensamblador.

El «Gens» se caracteriza por la rapidez en la ejecución, y añade otras ventajas, como incluir un ensamblado condicional, muchos comandos del ensamblador y una tabla de símbolos binaria.

Muchas son las opciones de este ensamblador, pero como analizar detenidamente cada una de ellas supondría escribir de nuevo un libro de instrucciones, como el que acompaña a ambos programas, nos limitaremos a señalar que algunas de ellas son: ensamblaje rápido, producir un listado de la tabla de símbolos, no generar ningún código objeto y dirigir el listado del ensamblador a la impresora.

```

>A
Table size:
Options:
*HISOFT GEN53M ASSEMBLER*
  ZX SPECTRUM
Copyright © HISOFT 1983
All rights reserved
Pass 1 errors: 00
Pass 2 errors: 00
Table used: 13 from 100
>B
    
```

"THE CODE MACHINE"

Todos los usuarios interesados en la programación en Código Máquina conocerán sin duda el popular paquete monitor/ensamblador Editas, de la casa Gremlins. Este paquete ha sido actualizado por esta misma compañía y ha dado lugar a la nueva versión 3.1, que recibe el nombre global de «The Code Machine», pudiendo ser adaptados ambos programas a casi todos los modelos de interface de impresora.

El editor ensamblador trabaja en modo 40 columnas divididos en distintos campos. El primero permite introducir los comandos y reenumerar las líneas; el segundo para la utilización de etiquetas; el tercero destinado a los nemónicos y el último para operandos. Los co-

THE CODE MACHINE

DISCOVER THE KEY TO SPECTRUM MACHINE CODING

DISCOVERY SOFTWARE

Alpha House, 10 Colver Street, Sheffield S1 4FS
Tel: (0742) 753423

WHEN YOU HAVE MADE THE DISCOVERY NOTHING ELSE COMES CLOSE

mandos de edición más utilizados son: LIST, EDIT, AUTO, NEW, RETURN, RENUM y DELETE.

El programa monitor nos

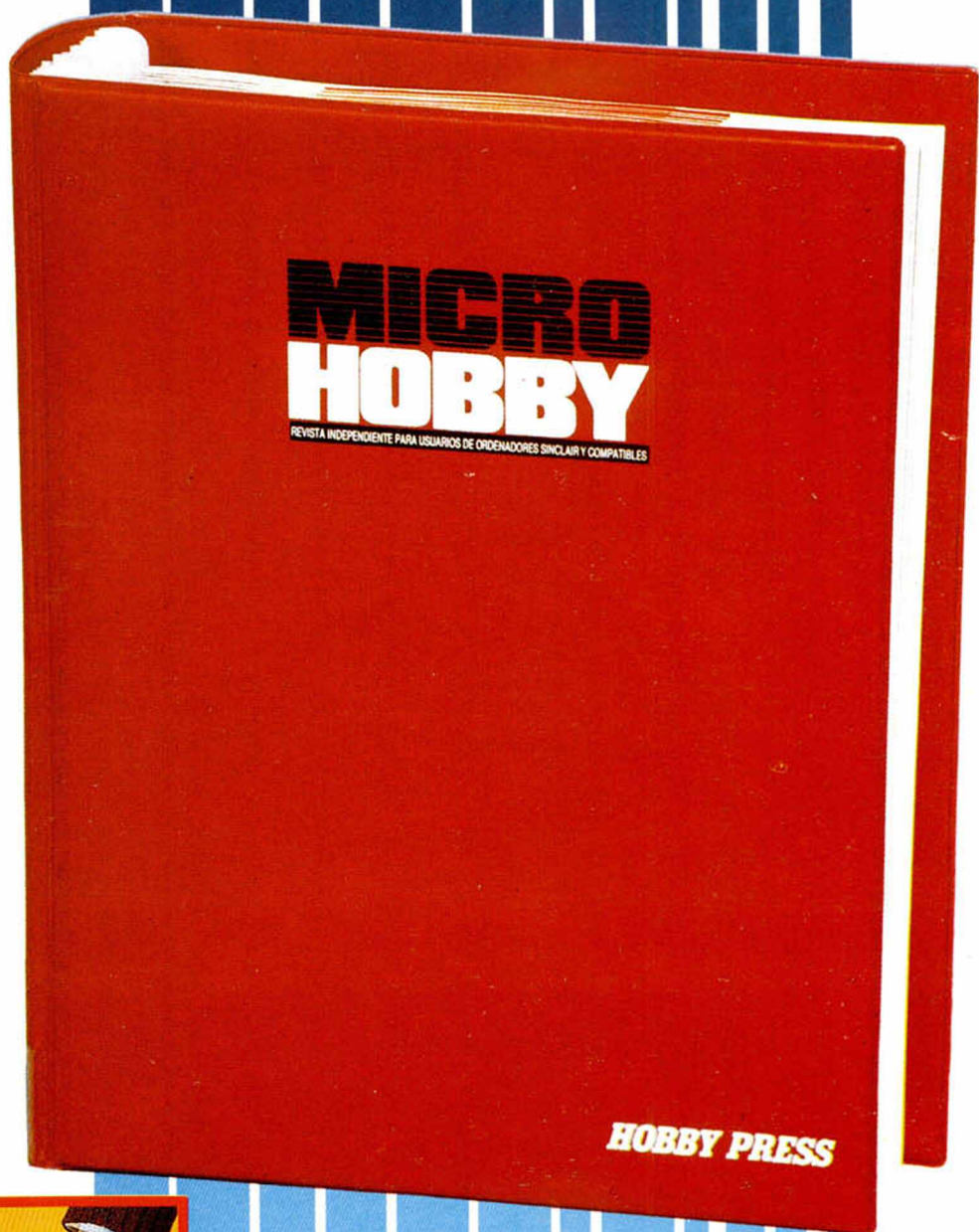
permitirá estudiar y trabajar sobre nuestras propias rutinas reubicarlas o analizar las rutinas de otros programas.

Dos herramientas imprescindibles para adentrarnos en el mundo de la programación de una forma mucho más asequible.

COLECCIONA MICROHOBBY!

850 ptas.

Para solicitar
tus tapas,
llámanos
al tel. (91)
734 65 00



No necesita encuadernación,

gracias a un sencillo
sistema de fijación
que permite además
extraer cada revista
cuantas veces sea necesario.

AUTO VOLTAJE

NONAMED

SPECTRUM • MSX
AMSTRAD

GAME OVER

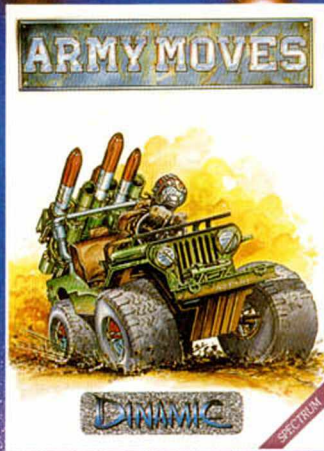
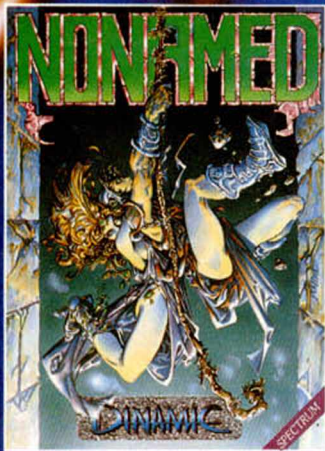
SPECTRUM
AMSTRAD

ARMY MOVES

SPECTRUM • MSX
AMSTRAD • CBM 64

DUSTIN

SPECTRUM
AMSTRAD



875 PTS. CADA UNO, NUEVO PRECIO DINAMIC

¡¡INCREIBLE!!
LOS 4 JUEGOS EN UN
DISCO AMSTRAD
SOLO: 2.750 pts.

DINAMIC SOFTWARE. Plaza de España, 18.
Torre de Madrid, 29-1. 28008 Madrid.
Pedidos contra reembolso (de lunes a viernes,
de 10 a 2 y de 4 a 8 horas): Teléfono (91) 248 78 87.
Tiendas y Distribuidores: Teléfono (91) 447 34 10.

