150 ptas.

miCOMPUTER 41

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

mi COMPUTER CURSO PRACTICO

DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IV - Fascículo 41

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-005-8 (tomo 4)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 248410
Impreso en España - Printed in Spain - Octubre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

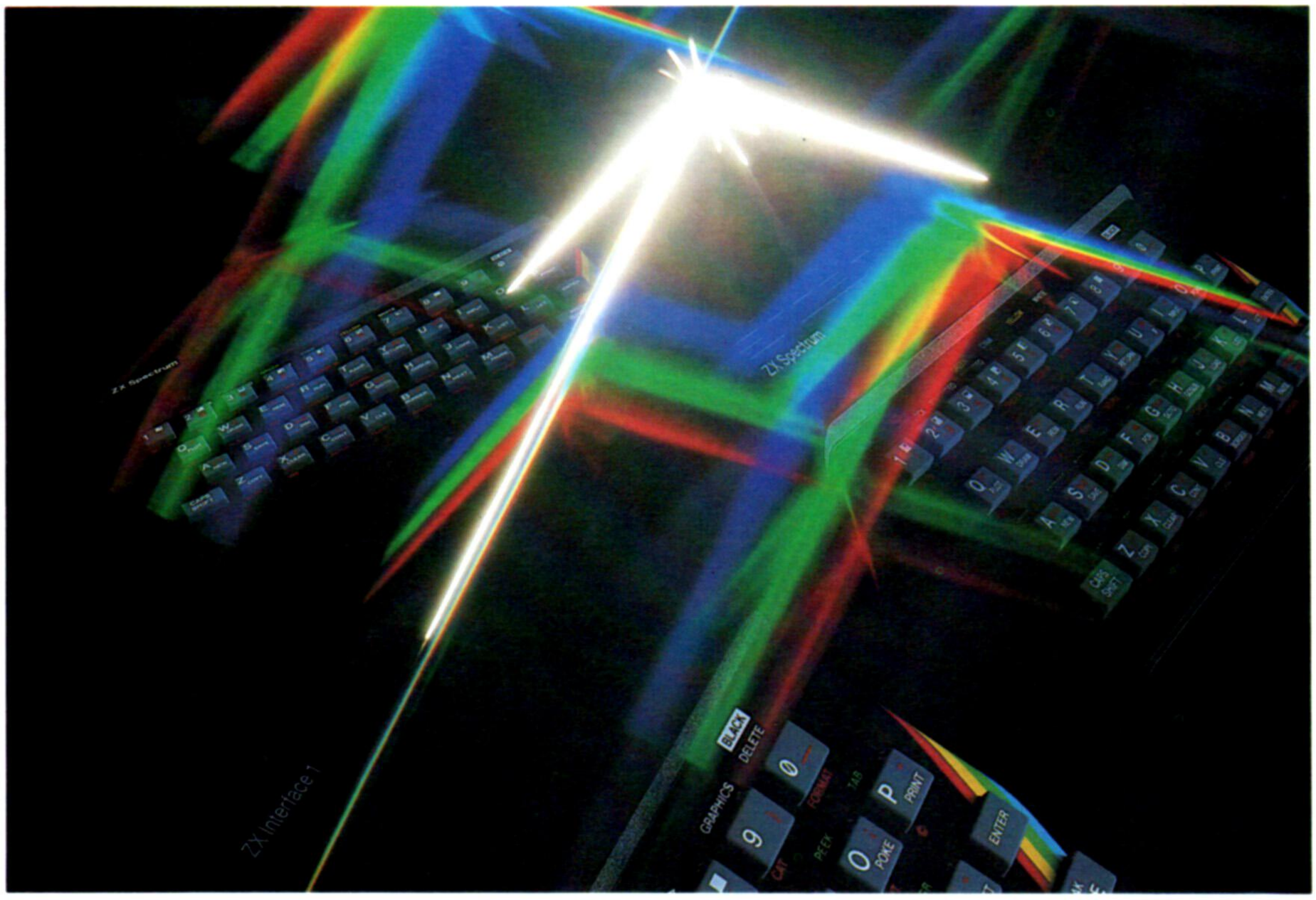
Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.

Editorial Delta, S.A.

Líneas de transmisión

En este capítulo veremos cómo se organiza una red de información utilizando un grupo de ordenadores personales



Paul Chave

Interface 1
Las redes no implican necesariamente kilómetros de cables y equipos costosos: los Spectrum equipados con la Interface 1 se pueden comunicar entre sí y compartir microdrives y una impresora, conformando un sistema de precio muy reducido

Una red es un sistema de ordenadores enlazados entre sí para compartir datos y equipos. Sin embargo, cada ordenador posee su propio sistema operativo y sus propios "protocolos" (procedimientos, reglas de formateado, etc.) para la comunicación con el mundo exterior. Debido a estos problemas de compatibilidad, las estaciones individuales o *nodos* de una red deben estar siempre constituidos por ordenadores similares: todos Spectrum, o todos Apple, por ejemplo.

Para poder realizar el análisis, vamos a suponer que hay un grupo de personas que poseen cinco ordenadores y desean conectarlos a una misma impresora. Nuestro grupo ha de ser capaz de enviarle información a la impresora desde cualquiera de los nodos. ¿Qué sucede si dos o más nodos tienen texto para imprimir al mismo tiempo? Y, aún más importante, ¿qué pasa si el nodo 3 tiene texto para imprimir pero necesita seguir trabajando mientras la impresora está en funcionamiento? Para resolver estos problemas debemos instalar un sexto ordenador, llamado *manejador de impresión*. El papel de esta máquina consiste en controlar el flujo de datos hacia la impresora y, por lo tanto, no se puede destinar a ningún otro fin. El manejador de impresión almacenará los documentos por orden de prioridad. Una vez que un nodo ha enviado su texto al manejador de impresión, el nodo puede realizar otros trabajos.

El empleo de una máquina exclusiva que actúe como manejador es esencial para una red, porque es a través del manejador que se puede compartir la información. Además de un manejador de impresión, algunas aplicaciones de red requieren un *manejador de archivos* para manipular las unidades de disco compartidas y controlar el flujo de información de nodo a nodo.

El siguiente paso consiste en crear un enlace entre los ordenadores-nodos. Esto se hace tendiendo un cable (ya sea un cable de dos hilos trenzados o un cable coaxial) de máquina a máquina. Aunque existen varias disposiciones posibles para los nodos y la estación del manejador —configuraciones en forma de estrella *(star)* o de anillo *(ring)*—, el concepto es esencialmente el mismo, de modo que describiremos el proceso de manera general. Para efectuar la conexión se suele requerir una interface especial para red para cada nodo. Dicha interface puede ser una conexión RS232 simple o una placa de circuito impreso conectable. Además, la estación del manejador requiere una unidad de almacenamiento con capacidad suficiente para manipular toda la carga de trabajo. La estación del manejador requiere, asimismo, suficiente RAM para administrar la red.

Es el software que utiliza un ordenador lo que determina que una máquina realice correctamente su función, y esto es particularmente cierto aplicado a redes. Antes que nada debe haber una "capa" de software sobre el sistema operativo de la máquina que establezca la conexión entre cada nodo y la red. Conocida como *software para red*, esta capa

coloca la estación del manejador al mando de las operaciones especializadas de manejo de archivos y el manejo de impresión, y también le proporciona el control sobre el flujo de datos dentro de la red. Además de establecer esta cadena de mando, el software para red informa al ordenador sobre cada nodo que está en una red, que hay un manejador instalador y cuántos otros nodos hay. Por último, el software para red les proporciona a los ordenadores de los nodos el protocolo que éstos precisan indefectiblemente al objeto de poder comunicarse con el resto del sistema.

Una vez establecida la capa de la red, los nodos individuales deben tener un programa, o un conjunto de programas, para que sus propias aplicaciones reconozcan la red y sepan cómo comunicarse con la misma. Éste es un software escrito especialmente para aplicaciones de red, y se puede ejecutar desde una cassette o una unidad de disco en el nodo, o a través del manejador de archivos. El software sólo es tan complicado como la operación. Si el nodo 1 de nuestra red está ejecutando un programa de tratamiento de textos y está enviando los resultados a la impresora independientemente de los otros nodos, la única modificación requerida para un procesador de textos estándar es la inclusión de protocolos para red. Por otra parte, si los nodos 2 y 4 han de utilizar los mismos datos, y ser capaces de ver los resultados de cada uno, las cosas se vuelven más complicadas. En este caso, el software de aplicaciones (ya sea un procesador de textos, hoja electrónica, base de datos o incluso un juego) y el hardware del sistema deben tener la capacidad de trabajar en multitarea. Dicho en otras palabras, la CPU ha de ser capaz de manejar más de una tarea al mismo tiempo, y debe tener, asimismo, la capacidad de gestionar las comunicaciones que reciba procedentes de al menos otras dos CPU simultáneamente.

Tony Sleep

Experiencia compartida
Esta escuela se equipó con 16 micros BBC Micro con pantallas en color, una impresora, una unidad de disco doble y Econet (la LAN de Acorn para el BBC Micro), invirtiendo menos dinero del que hubiera costado instalar una unidad de disco y una impresora para cada ordenador. (Las siglas LAN corresponden a *Local Area Network*: red de área local.) La velocidad de la red es tal que cada puesto de trabajo parece tener el uso exclusivo de la unidad de disco, incluso con 30 alumnos trabajando; los terminales disponen de una "cola de red" para usar la impresora. El Econet le permite al profesor de informática proporcionar experiencia práctica a todos sus alumnos regularmente; la comunicación interterminal es de gran valor cuando la red se utiliza para la enseñanza de asignaturas

La empresa que se ha mostrado más activa e interesada en poner las redes al alcance de los usuarios de ordenadores personales es, sin duda, Sinclair Research. La unidad para accesorios Interface 1 del Spectrum lleva incorporada una interface para red. Esta unidad se está vendiendo bien porque es necesaria para usar los microdrives con el Spectrum. Es probable que cuando se hayan vendido suficientes unidades de la Interface 1 se ponga en venta software apropiado para hacer uso de la interface para red.

El último micro de Sinclair, el QL, lleva incorporada como estándar una interface para red similar, que debería ser compatible con la versión para el Spectrum. Aunque esta interface es bastante primaria, la popularidad de estas máquinas debería hacer que valiera la pena producir software de redes para ellas. Los programas para juegos son los primeros y obvios candidatos. Más allá de ellos, las posibles aplicaciones para usuarios de ordenadores personales pueden considerarse, lamentablemente, más bien limitadas.

Antes de que la conexión de ordenadores en red sea auténticamente viable hay varios elementos que son necesarios. El más simple de todos, por supuesto, es que debe haber al menos dos micros a conectar entre sí. En segundo lugar, los micros deben estar lo bastante próximos el uno del otro para que se puedan unir entre sí mediante cables. Esto significa que han de estar en el mismo edifico. Por último, debe haber suficiente "tráfico" como para que se justifique el uso de una red. Esto significa que es necesario que existan usuarios que se intercambien datos muchas veces al día, o bien que deseen compartir equipos caros (tales como impresoras o unidades de disco) como fórmula para amortizar el costo de la red.

Si sólo se hubiera de enviar por la red una pequeña cantidad de datos, sería más fácil que un usuario se los pasara al otro en cinta o disco. Del mismo modo, si la red sólo está formada por unos pocos micros, sería más barato suministrarle a cada uno de ellos su propia impresora y unidad de disco, en vez de invertir en el costo adicional de ordenadores para la gestión de la red.

Por consiguiente, aparte de los juegos, los únicos casos prácticos para conectar micros personales en red se dan en pequeñas empresas y en los centros de enseñanza. El Sinclair QL ha hecho que el costo de las redes disminuya hasta un nivel en el que resulta rentable proporcionarle un ordenador al personal que no necesita utilizar los ordenadores de forma intensiva. Son muchas las personas que antes de que haya transcurrido mucho tiempo se encontrarán con ordenadores conectados en red encima del escritorio de su oficina.

Caja del reloj
Genera las señales de temporización que sincronizan todas las comunicaciones de la red

EDITOR

Manejador de archivos
La función de este micro es dirigir la red. Los usuarios poseen sus propias contraseñas de red y directorios de disco particulares. También los usuarios pueden disponer de un directorio público de archivos

Impresora
A la misma se accede desde cualquier estación de la red, y se controla mediante una ROM *background* localizada en una de las estaciones, dejándola libre para el trabajo normal

EDITOR DE PRODUCCIÓN

Acoplador acústico
Para la información llegada vía línea telefónica a la única unidad de disco a través de un terminal fuera de línea que luego se conecta a la red y le transmite la información recibida

CAJA DE CONEXIÓN

SUBEDITOR

Terminador
Termina electrónicamente las terminaciones del bus de datos para impedir la degradación de las señales

Identificación del terminal
Cada terminal posee un número de identificación de red exclusivo (entre 001 y 254) establecido mediante interruptores internos

Acceso de terminal
Todos los terminales pueden enviar o recibir mensajes a o desde otro terminal

REDACTOR

REDACTOR

Prensa

La red se está utilizando para producir un periódico imaginario. Los redactores emplean sus terminales como procesadores de textos y guardan su copia en una unidad de disco común. El equipo de producción puede contemplar las pantallas de los redactores en sus propios terminales en cualquier momento y luego editar la copia terminada desde los archivos en disco. El micro del editor controla la red, de modo que la copia acabada se puede leer desde la impresora. Cabe recibir copias desde el ordenador de otra escuela por el acoplador acústico

Steve Cross

Todo bajo control

Veamos cómo se obtienen efectos especiales sobre papel y cómo se realiza la conexión entre ordenador e impresora

ÉLITE
ABCDEFGHIJKLMNOP
QRSTUVWXYZabcdef
ghijklmnopqrstuv
wxyz0123456789 !
"£$%&'()*+,-./:;
<=>?@[\]^_`{|}~

PICA
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
£$%&'()*+,-./
:;<=>?@[\]^_`
{|}~

CURSIVA PICA ENFATIZADA
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
#$%&'()+,-./*
:;<=>?@[¥]^_`
{|}~

ÉLITE ALARGADA
ABCDEFGH
IJKLMNOP
QRSTUVWX
YZabcdef
ghijklmn
opqrstuv
wxyz0123
456789 !
"£$%&'()
*+,-./:;
<=>?@[\]
^_`{|}~

PICA CONDENSADA
DE PERCUSIÓN DOBLE
ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 !"£$%&
'()*+,-./:;<=>?@[\]^_`{
|}~

Sólo con puntos
Las impresoras matriciales ofrecen una gama de tipos de letras como pica, élite y cursiva, y diversos estilos de tipos, como condensados, alargados y enfatizados. Los ejemplos que vemos aquí fueron obtenidos con una Epson FX-80

Una impresora matricial puede hacer mucho más que producir simplemente listados de programas. Una rápida ojeada al manual de usuario de la impresora le enseñará que se pueden producir sobre el papel diversos "efectos especiales". Incluso las impresoras matriciales más baratas le permitirán modificar el tamaño de los caracteres impresos sobre el papel. Normalmente, el texto se imprime en 80 caracteres por línea, pero este número se puede aumentar seleccionando la modalidad "impresión condensada" (que utiliza caracteres más pequeños) o disminuir seleccionando "impresión alargada". De modo similar, se puede alterar el espaciado entre líneas. Un espaciado grande de cuatro líneas por pulgada, por ejemplo, se podría reducir a, digamos, ocho líneas por pulgada, lo que proporciona un listado de mayor densidad.

La impresora que vamos a examinar en profundidad en este capítulo, la Epson FX-80 es un buen ejemplo de máquina con una amplia gama de posibilidades de impresión. La modalidad "enfatizada", que imprime el texto en negrita, y la modalidad alternativa, que cambia de los tipos de letras normales a los caracteres en cursiva, son dos de sus facilidades estándar. Pero quizá su característica más interesante sea su capacidad para cambiar cualquiera de los caracteres almacenados en la memoria de la impresora, una facilidad sumamente útil para los alfabetos extranjeros o para la impresión de símbolos científicos. Antes de pasar a investigar cómo se producen estas configuraciones, consideremos cómo lleva a cabo una impresora la sencilla tarea de imprimir el listado de un programa.

La forma en que un ordenador le "habla" a una impresora varía de una máquina a otra. El Dragon, por ejemplo, utiliza una variante de la instrucción LIST (LLIST) para indicar a la impresora que produzca el listado de un programa. Otras máquinas exigen la apertura de "canales" o "corrientes" *(streams)* para tener acceso a la impresora. Como el procedimiento exacto varía tanto, lo mejor es consultar en el manual de usuario de su ordenador (es poco probable que en este caso el manual de la impresora sea de alguna utilidad).

Habiendo establecido la comunicación entre las dos máquinas, puede que su primer listado sea decepcionante. Los problemas más comunes son que todo el texto se imprima en una línea negra indescifrable, o que haya líneas en blanco entre cada línea del programa. La explicación para estos dos fallos radica en la diferencia entre un carácter de "salto de línea" y un carácter de "retorno de carro". Después de que su ordenador le envía una línea de texto a la impresora, también le hace llegar un carácter de retorno de carro, que mueve el cabezal de impresión otra vez al margen izquierdo, listo para imprimir una nueva línea. Algunos ordenadores también envían un carácter de salto de línea para mover el papel una línea hacia arriba; otros dan por sentado que es la propia impresora la que hace esto de forma automática. Para complicar las cosas todavía más, la mayoría de las impresoras posee un interruptor interno que decide si éstas generan sus propios saltos de línea o no. Si se plantea alguno de estos problemas, busque este interruptor (consultando el manual de la impresora) y colóquelo en la posición alternativa.

Aparte de producir listados de programas, una impresora también se puede utilizar como un dispositivo de salida: en vez de visualizar los caracteres en la pantalla, éstos se pueden imprimir en papel. Nuevamente, el procedimiento exacto para hacer esto varía de un ordenador a otro; la instrucción estándar en BASIC es LPRINT y ésta la emplean el Spectrum y el Oric. En el Commodore 64, OPEN1,4 seguida de PRINT#1,"HOLA" imprimirá la palabra "HOLA". En un micro Dragon, la misma tarea se realiza utilizando PRINT#−2,"HOLA". El BBC Micro emplea VDU2 seguido de PRINT "HOLA" y la instrucción VDU3. Los ejemplos de programación que proporcionamos aquí utilizan LPRINT, de modo que quizá deba ser modificada para su máquina.

Etiquetas de direcciones

```
10 LPRINT "SR. JOSE GOMEZ"
20 LPRINT "C/ DEL PEZ, 9"
30 LPRINT "CIUDAD"
40 LPRINT "ABC 123"
50 FOR I = 1 TO 7
60 LPRINT
70 NEXT I
100 GOTO 10
```

Este listado es un programa sencillo para producir etiquetas de direcciones. Éstas se pueden comprar en un rollo con agujeros para rueda dentada en ambos lados, de modo que se puedan utilizar con el arrastre de tracción de la impresora. Dado que no emplea códigos de control especiales, el programa funcionará en impresoras de cualquier marca. Tal como está, imprime el mismo nombre y la misma dirección una y otra vez. Quizá usted desee modificarlo con el fin de que pueda entrar distintos nombres y direcciones, o incluso leerlos de un archivo de datos. El bucle FOR...NEXT entre las líneas 50 y 70 imprime siete líneas en blanco y sirve para posicionar el cabezal de impresión justo al comienzo de cada etiqueta. Tal vez sea necesario ajustar el número exacto de líneas en blanco para su máquina.

Nuestro programa es bastante adecuado simplemente para imprimir etiquetas, pero para imprimir algo más complicado, como una factura o un membrete, vamos a tener que aplicar algunos de los efectos especiales que hemos mencionado antes. Éstos se producen enviando a la impresora códigos de control además de los caracteres del texto.

Junto con tener un código para cada carácter del teclado, el juego de caracteres ASCII (véase p. 557) posee un grupo de caracteres "invisibles" que no imprimen nada ni en pantalla ni sobre papel. Son estos códigos los que se utilizan para activar los efectos especiales de la impresora: en el juego ASCII estándar hay cuatro códigos (17, 18, 19 y 20) que están reservados como órdenes para control de periféricos. Lamentablemente, el juego de caracteres ASCII no posee suficientes caracteres de control reservados para las setenta y pico posibilidades de una Epson FX-80, y para superar este inconveniente la mayoría de los efectos se producen enviando a la impresora *códigos de escape*. Éstos se componen de dos o más códigos de carácter, empezando con un carácter ESC (código ASCII 27). Por ejemplo, para activar la facilidad de espaciado proporcional en una Epson, se envía ESC-p, o sea, el carácter Escape seguido del carácter "p" minúscula.

En BASIC esto se escribe de la siguiente forma:

```
LPRINT CHR$(27);"p"
```

Normalmente el carácter EC no se puede generar mediante la pulsación de la tecla Escape en el teclado y, por consiguiente, se emplea la función CHR$.

En el BBC se utilizaría:

```
VDU2
VDU1,27,1,112
VDU3
```

La instrucción VDU2 activa la impresora; VDU1 significa "enviar el siguiente carácter sólo a la impresora" (PRINT enviaría el siguiente carácter ESC también a la pantalla, con resultados no deseados). VDU3 desactiva la impresora.

Estas secuencias de instrucciones se aplican sólo a la Epson FX-80. Si se enviara el mismo código a una impresora diferente, o bien no tendría ningún efecto, o realizaría algo inesperado, o haría que la impresora se "colgara" (es decir, se negara a responder al ordenador).

Creación de una factura

Nuestro segundo listado ilustra el uso de algunas de las facilidades de la Epson para crear el membrete de una factura como la que podría emplear un pequeño garaje. Los códigos que hemos aplicado aquí son los que se utilizan para la Epson FX-80. La gama de impresoras Epson es una de las más populares; tanto es así que otros fabricantes hacen modelos "compatibles con Epson". No obstante, si su impresora es incompatible, deberá modificar los códigos de control.

```
999 REM MEMBRETE DE FACTURA
1000 LPRINT CHR$(12)
1010 LPRINT CHR$(14);TAB(11);"MOTORES PLIM, S.A."
1020 LPRINT CHR$(13);CHR$(13)
1030 LPRINT CHR$(27);"E";
1040 LPRINT TAB(36);
1050 LPRINT CHR$(27);"-";CHR$(1);
1060 LPRINT "FACTURA";
1070 LPRINT CHR$(27)"-";CHR$(0);
1080 LPRINT CHR$(27);"F";
1090 LPRINT CHR$(13);CHR$(13);CHR$(13)
1100 REM IMPRESION DETALLES FACTURA
```

Para empezar, la línea 1000 le envía a la impresora el carácter cuyo código es 12. Éste es el carácter de "salto de página", que le indica a la impresora que mueva el papel al comienzo de una nueva hoja. Luego tenemos el código ASCII 14; éste es el carácter denominado *shift out* (SO) y en la Epson hace que todo el texto subsiguiente se imprima en letras alargadas. En nuestro programa se utiliza para el membrete, obteniéndose el nombre del garaje en letras grandes. La función TAB se emplea para centrar el membrete.

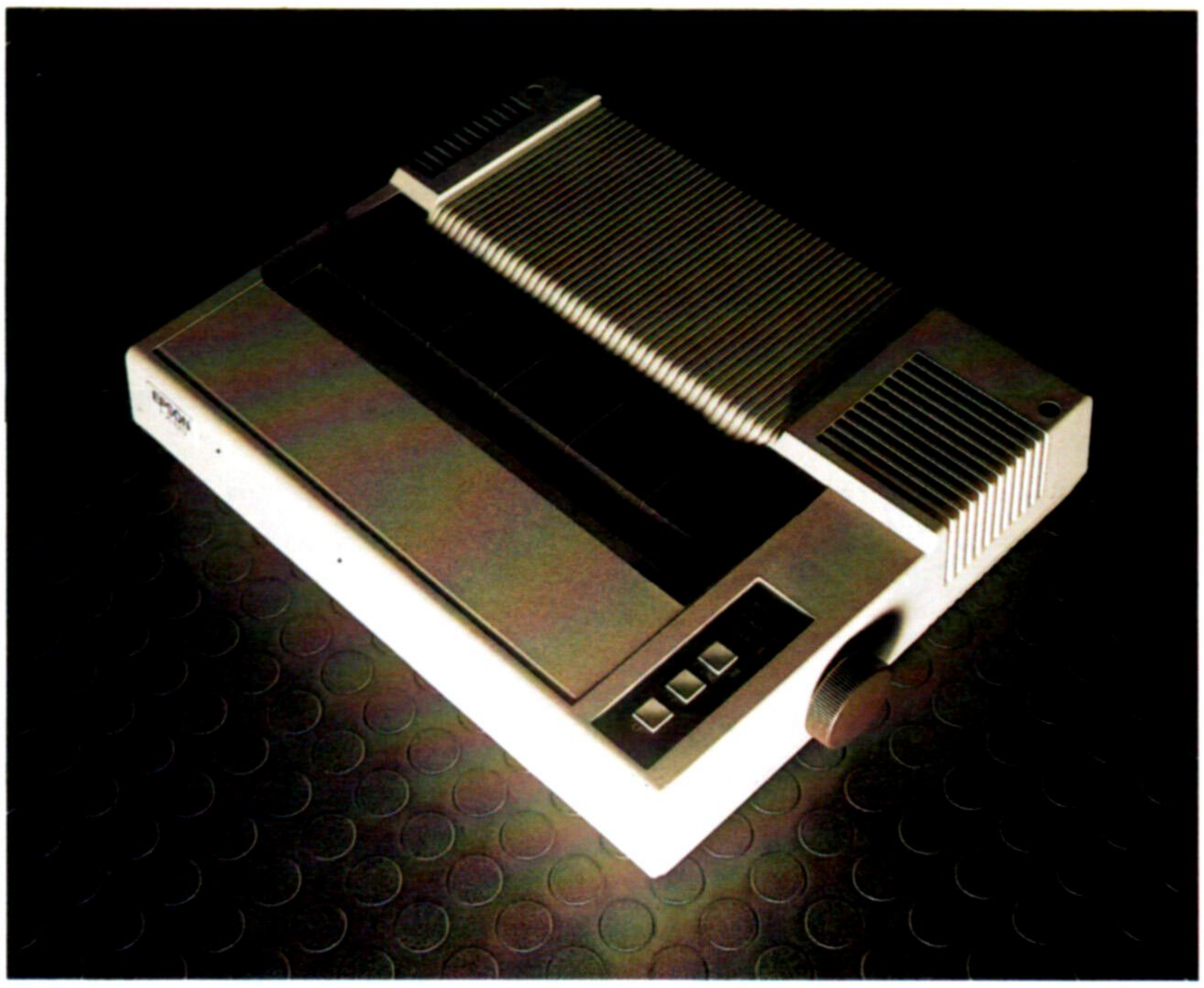
CHR$(13) es el carácter de retorno de carro, que al ser enviado a la impresora produce una línea en blanco. Entre las líneas 1020 y 1090 se emplea varias veces para espaciar la parte superior de la factura ESC-E. En la línea 1030, activa la modalidad enfatizada, y todo el texto a continuación se imprime en negrita (que se obtiene imprimiendo varias veces la misma letra). La línea 1050 activa la característica de "subrayado" y la línea 1070 la desactiva, después de imprimir y subrayar la palabra "factura". ESC-F desactiva la modalidad enfatizada. El listado tendrá el siguiente aspecto:

```
MOTORES PLIM, S.A.

FACTURA
```

Aquí sólo hemos reseñado la parte inicial del programa; un programa de facturación completo incluirá instrucciones para imprimir los detalles del cliente: nombre, marca del coche, cantidad adeudada, etc. Estos detalles se habrán obtenido a partir de una serie de preguntas realizadas al principio del programa y las respuestas se habrán almacenado en variables dentro de éste.

Los dos programas que hemos ofrecido son ejemplos sencillos de los tipos de usos alternativos que se pueden dar a una impresora matricial. En la actualidad son muchas las personas que están explorando la utilización de una impresora para algo más que para hacer listados de programas. De hecho, programando la impresora, el usuario puede disfrutar tanto como cuando programa el propio ordenador.



Ian McKinnell

Epson FX-80
Aunque cara, la Epson FX-80 es una impresora popular entre los propietarios de micros personales y de gestión. La FX-80 tiene un cabezal de nueve agujas y una velocidad de impresión máxima de 160 caracteres por segundo. La mayoría de los paquetes de software (tratamiento de textos, p. ej.) admite impresoras Epson o tipo Epson

Entrega inmediata

Un sistema de correspondencia para un micro necesita una impresora, un paquete para tratamiento de textos y un programa de base de datos

El correo informatizado puede llegar muchísimo más allá de la simple impresión de etiquetas de dirección. En realidad, si todo lo que necesitáramos fuera alguna forma de colocar direcciones en sobres, el tratamiento de textos no sería necesario. Un paquete de base de datos sería suficiente, dado que preparar un archivo de nombres y direcciones es muy fácil en una base de datos estándar. A pesar de que inicialmente cada registro de nombre y dirección habría que entrarlo campo por campo, una vez que todo el sistema estuviera preparado podríamos utilizar el paquete de base de datos, una y otra vez, para generar todas las etiquetas precisas.

Dado que los paquetes de base de datos, por poco sofisticados que sean, poseen facilidades para informes, sería posible seleccionar las etiquetas a imprimir de acuerdo a cierto criterio: por ejemplo, todas las direcciones que lleven Barcelona en el campo de la ciudad. Éste se encargaría de seleccionar y crear las etiquetas para los sobres, pero aún dejaría en manos del usuario la realización de la otra mitad de un trabajo tedioso. Si, por ejemplo, se deseara enviar a un conjunto determinado de clientes una carta normalizada, también sería necesario digitar los nombres y las direcciones en el interior de cada una de las cartas.

Además, aunque cada cliente tuviera escrito su propio nombre y dirección en la cabecera de la carta, el texto de ésta permanecería anónimo e impersonal. No tiene mucho sentido, por ejemplo, dejar agujeros vacíos en el texto para rellenarlos luego con el nombre de cada cliente, porque los nombres tienen distintas longitudes. Se tendrían que dejar agujeros lo suficientemente largos como para dar cabida al nombre más largo que se deseara incluir, y la carta "personalizada" resultante tendría un aspecto desastroso. De modo que personalizar la carta implicaría volver a digitarla tantas veces como nombres de distinta longitud hubiera en la lista de correspondencia.

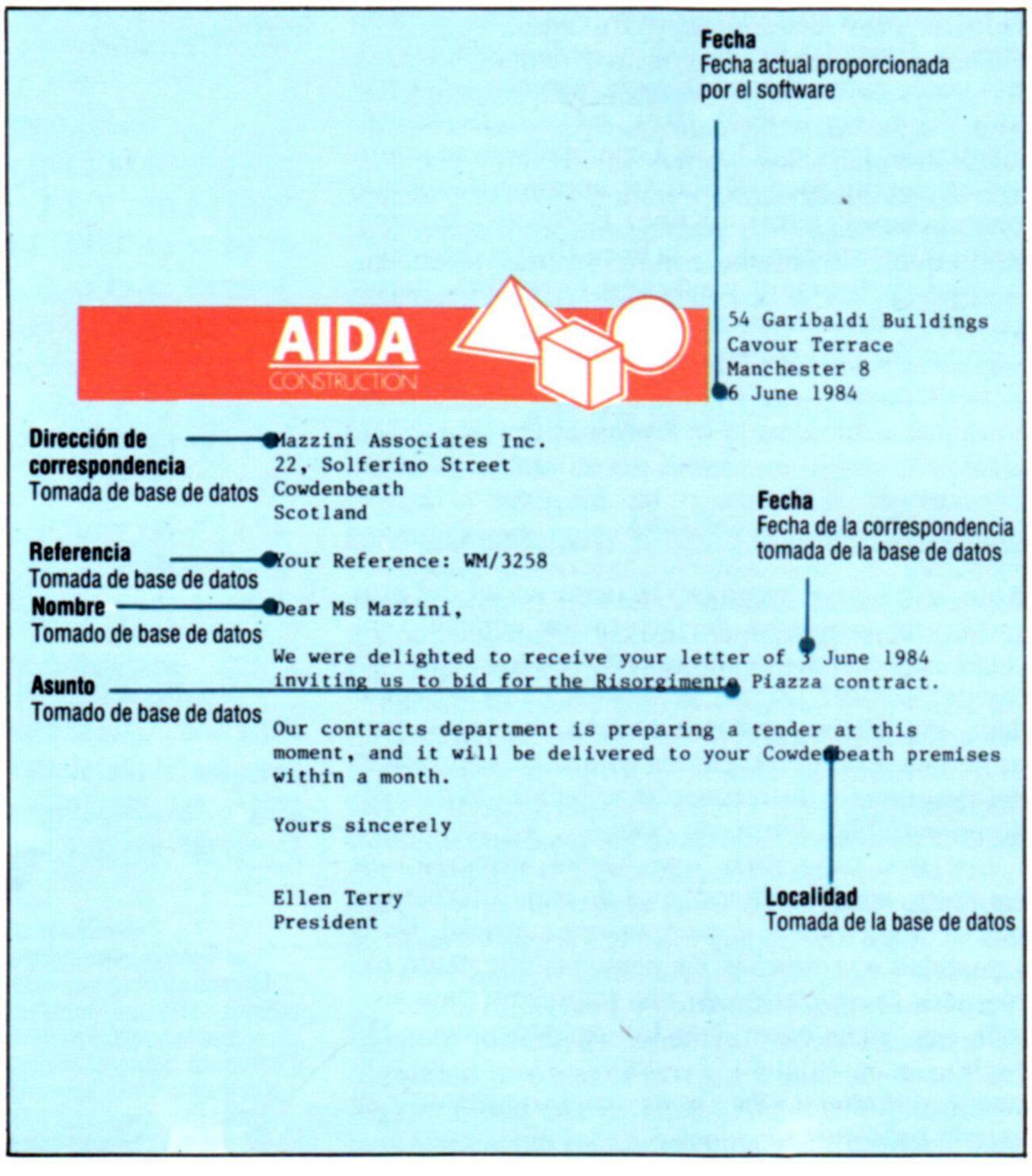
La solución ideal para este problema consiste en acomodar el sistema de modo que el usuario pueda componer una carta estándar empleando todas las facilidades de un procesador de textos, y después dejar que el ordenador extraiga automáticamente los detalles correspondientes a cada cliente de un archivo de base de datos, para agregarle a cada carta los detalles personalizados.

Existen varias maneras de hacer esto. Una de las más sencillas es la utilizada por dos paquetes basados en disco producidos por Acorn, denominados Memoplan y Fileplan, un procesador de textos y una base de datos, respectivamente.

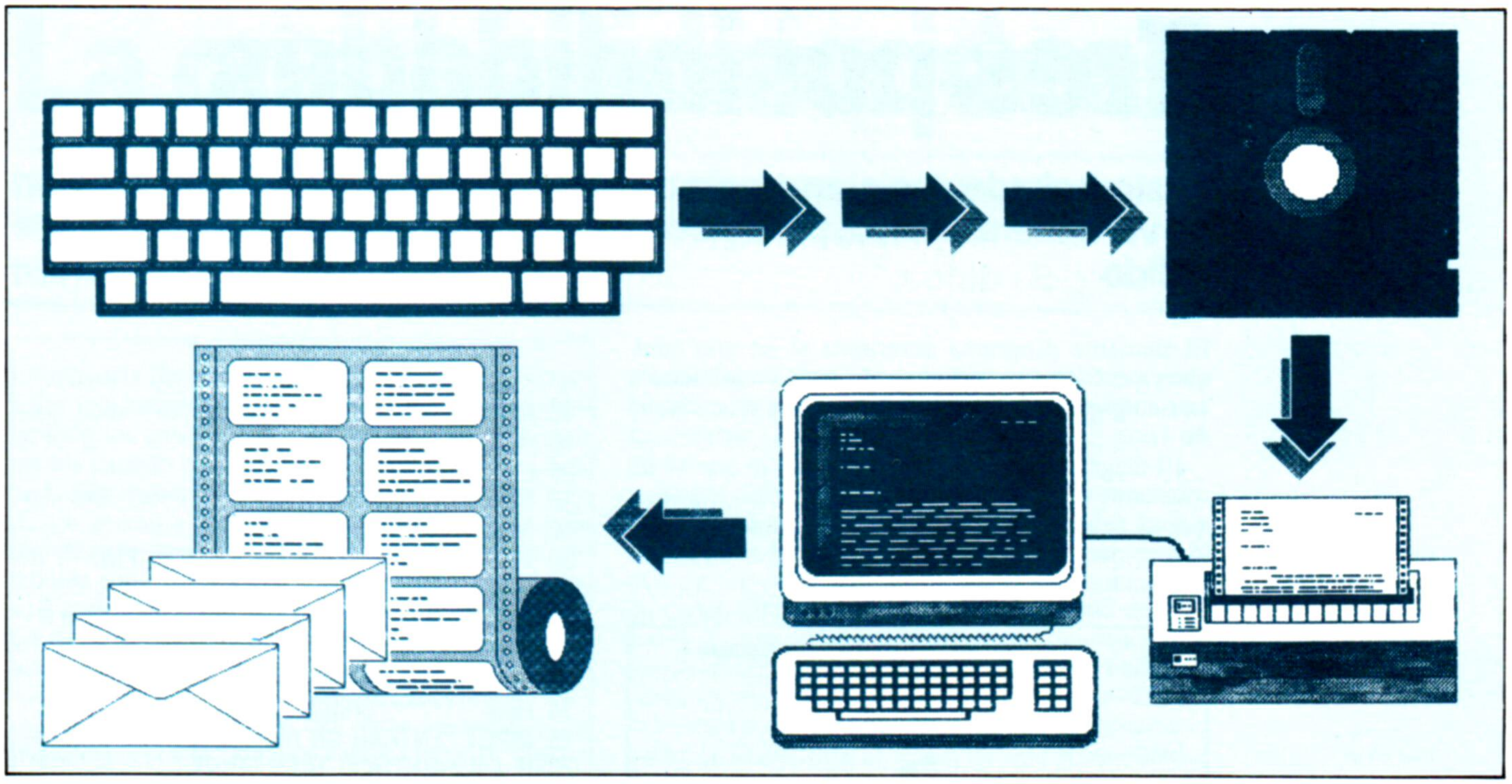
Los paquetes no pueden producir cartas personalizadas independientemente y, por tanto, deben usarse juntos. Los detalles de nombre y dirección se deben preparar con el Fileplan, y la carta estándar (o formulario) se crea mediante el Memoplan. Cada campo del archivo de nombres y direcciones está numerado y en los puntos pertinentes de la carta estándar el usuario simplemente teclea el número del campo cuyo contenido necesita transferir a la carta. El ordenador trabajará entonces secuencialmente con la lista de correspondencia y en la carta estándar aparecerán los contenidos de los números de campo para los registros seleccionados.

El procesador de textos ajusta automáticamente el texto circundante de modo que los destinatarios reciben una carta que parece haber sido escrita personalmente para ellos. El Memoplan también permite que la persona que escribe una carta estándar incluya un recordatorio como ayuda para identificar el contenido de un campo determinado. Por ejemplo, 2(APELLIDO) indica que el segundo campo contiene el apellido de cada destinatario.

Los paquetes de correspondencia especializados destinados a ordenadores de oficina como el IBM PC y el Sirius, tales como el Mailmerge, producido por Micropro (el paquete de correspondencia más



Liz Dixon



Ian McKinnell

conocido en Gran Bretaña), o el Mailing List Manager, creado por Peachtree, incluyen numerosas facilidades muy útiles y sofisticadas. No sólo se puede crear un gran número de archivos de direcciones diferentes, sino que también se dispone de amplias facilidades de búsqueda y selección que permiten realizar envíos especiales de correspondencia sólo a partes de una lista. Por ejemplo, la secretaria de un club de golf que utilice este sistema no tendría ningún problema para enviarles correspondencia a todos los nuevos socios del club con un récord inferior a 15 y que hayan pagado sus cuotas.

La búsqueda y la selección se consiguen mediante las técnicas estándar para bases de datos de indexación y claves de registros, llevando a cabo después comparaciones lógicas de los campos seleccionados del registro.

Estos avanzados paquetes también poseen facilidades de formateo muy precisas. Ésta es una característica particularmente útil, porque los nombres y direcciones producidos por el sistema se pueden confeccionar a medida para que se ajusten al tamaño y forma de las etiquetas. Con el sistema de Peachtree, por ejemplo, seleccionar la opción LABEL FORMAT en el menú principal produce en pantalla la imagen de una caja, junto con una lista de todos los campos del registro de la lista de correspondencia. Ello le proporciona al usuario una representación visual de la etiqueta que se está creando. Asimismo, hay facilidades para informar a la impresora de que, por ejemplo, la hoja de las etiquetas es de tres etiquetas de ancho, de modo que a cada pasada del cabezal de impresión se deben imprimir tres etiquetas, y así sucesivamente.

En general, los programas de correspondencia son de mayor utilidad cuanto mayores listas de correspondencia se explotan. Si usted sólo tiene que hacer unas cuantas etiquetas, probablemente lo más práctico sea mecanografiarlas una por una.

Un ejemplo de un paquete diseñado para el BBC Modelo B estándar es el que suministra GCC, una empresa de software con sede en Cambridge. Durante algún tiempo GCC ha comercializado su propia base de datos en ROM, llamada Starbase, que contaba con todas las facilidades necesarias para una base de datos (sólo listas de correspondencia). Pero ahora la empresa ha lanzado una versión de Starbase en 16 K de ROM, que proporciona completas facilidades para correspondencia, conjuntamente con el paquete de tratamiento de textos basado en ROM denominado Wordwise.

Starbase se suministra con un manual y un disco de utilidades. Junto con Wordwise, sus facilidades para correspondencia incluyen personalización de cartas normalizadas y creación de etiquetas. El disco de utilidades hace posible el envío de órdenes a la impresora durante la creación de etiquetas. De modo que en máquinas con impresoras adecuadas se puede utilizar una selección de distintos tipos de letras.

Una característica especialmente útil de Starbase como paquete para correspondencia es su capacidad para llevar a cabo operaciones aritméticas en los campos de un archivo de direcciones. De esta forma se pueden crear informes personalizados y facturas para los socios de clubes, realizando el ordenador todos los cálculos individuales necesarios.

Los usuarios del Commodore 64 pueden disponer de diversos paquetes para correspondencia. Un ejemplo es Visawrite, producido por Visa Software. A diferencia de Starbase, Visawrite funciona tanto con cassette como con disco, y se puede emplear con cualquier paquete de base de datos capaz de crear un archivo secuencial. Por otra parte, aplicado solo, como paquete de proceso, puede retener hasta 500 nombres y direcciones. Éstos se preparan utilizando cada página de documento como una ficha de registro y se pueden mezclar con una carta estándar.

Aplicado solo, Visawrite no posee facilidades de búsqueda selectiva, aunque los usuarios pueden repasar a mano todos los nombres y direcciones y marcarlos individualmente para su inclusión en una partida de correspondencia.

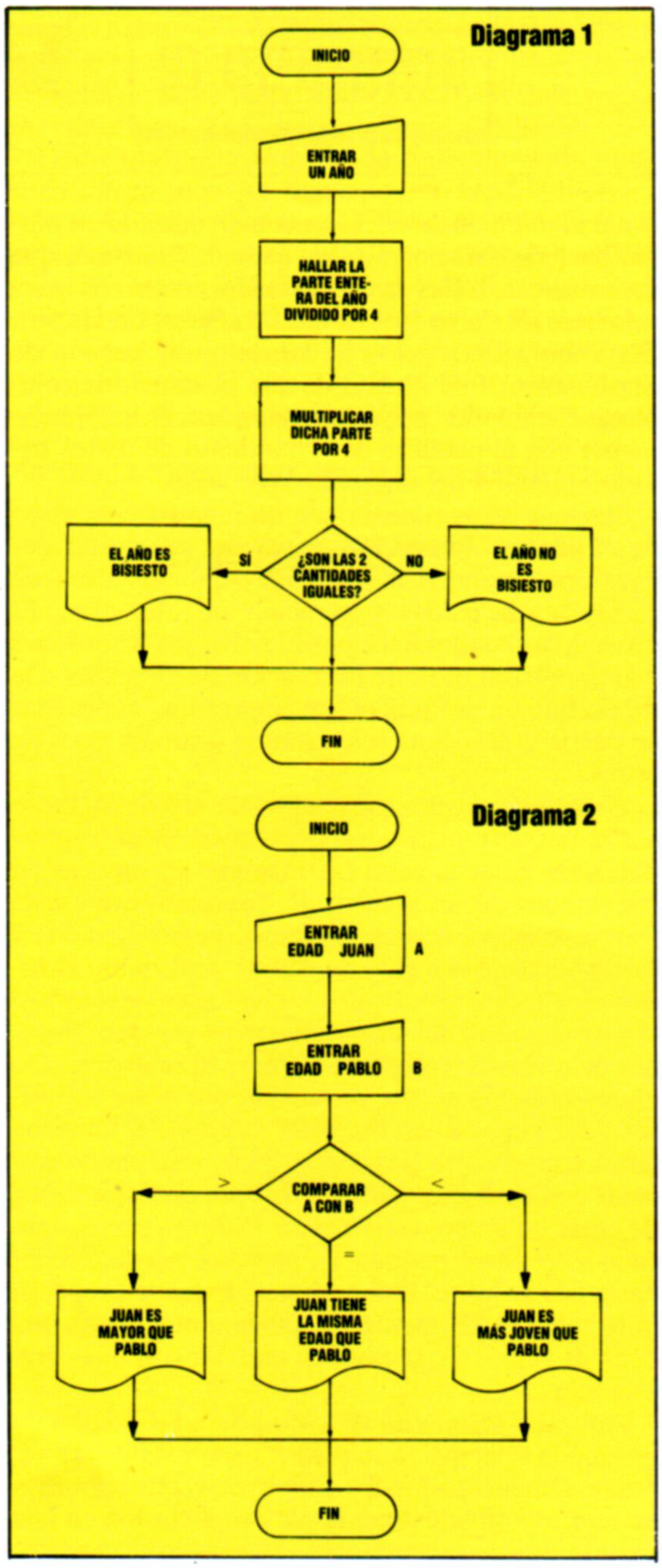
Correo en cadena
El procesador de textos proporciona un esquema de carta con espacios en blanco donde deberán consignarse los datos específicos como fecha, nombre y dirección, y la base de datos suministra los datos correspondientes contenidos en sus registros. Después de las cartas, se imprimen las direcciones en etiquetas autoadhesivas

Tres posibilidades

El símbolo de decisión o comparación determina si una afirmación es verdadera o falsa, pero puede proporcionar además una tercera salida

El siguiente programa determina si un año cualquiera entrado por teclado es bisiesto (suponiendo, en una primera fase, que un bisiesto es un múltiplo de 4).

El diagrama 1 muestra paso a paso lo que se ha transcrito en primer lugar (programa 1), mientras que el programa 2 se ha elaborado con la misma lógica, pero tratando de simplificar al máximo las operaciones.



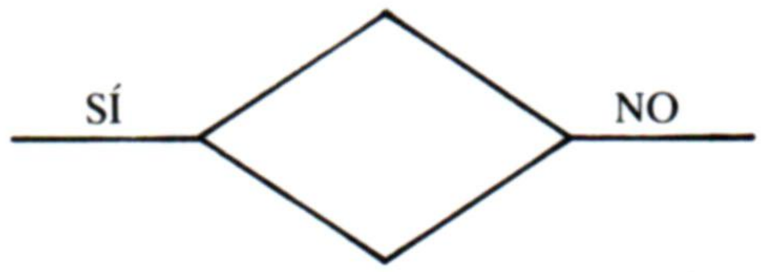
Programa 1

```
10 REM********BISIESTO
15 INPUT "ENTRAR UN AÑO";A
20 E = INT(A/4)
30 M = E*4
40 IF M = A THEN GOTO 60
50 PRINT "EL AÑO NO ES BISIESTO": END
60 PRINT "ES BISIESTO"
70 END
```

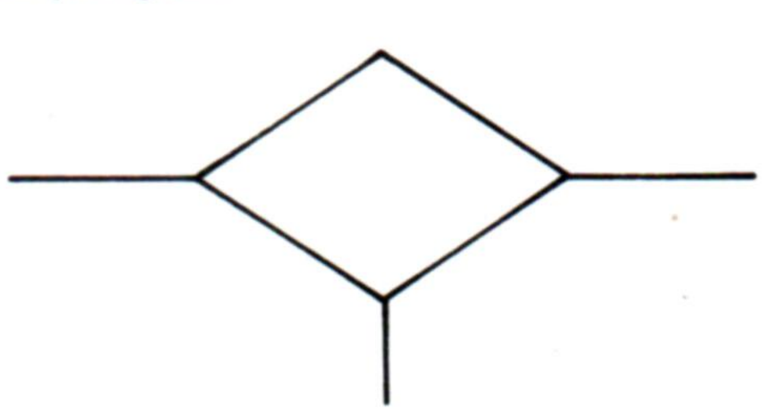
Programa 2

```
10 REM********BISIESTO
15 INPUT "ENTRAR UN AÑO";A
20 IF A/4 = INT(A/4) THEN PRINT
   "ES BISIESTO":END
30 PRINT "NO ES BISIESTO"
40 END
```

Hasta ahora, siempre que se ha precisado recurrir a una decisión, a base de confrontar elementos, esta comparación se ha hecho a nivel de verdadero o falso.



Para ello, el rombo ha proporcionado dos posibles salidas. Ahora bien, este mismo símbolo puede ofrecer una tercera vía de salida para que, en caso de que se necesite, tener la oportunidad de poder discernir las dos posibilidades de desigualdad, es decir, las de salida por mayor o menor además de la salida por igual.



En el diagrama 2 puede observarse cómo se utiliza esta técnica al comparar las edades de dos personas. Como se puede comprobar, la secuencia seguirá una de las tres posibles salidas, adquiriendo el resultado una precisión que no se hubiera obtenido en caso de preguntar solamente si Juan es mayor que Pablo, ya que en caso de que la respuesta hubiera sido negativa, habría quedado la duda de si se ha tomado esta vía por ser menor o por tener ambos la misma edad.

La combinación ideal

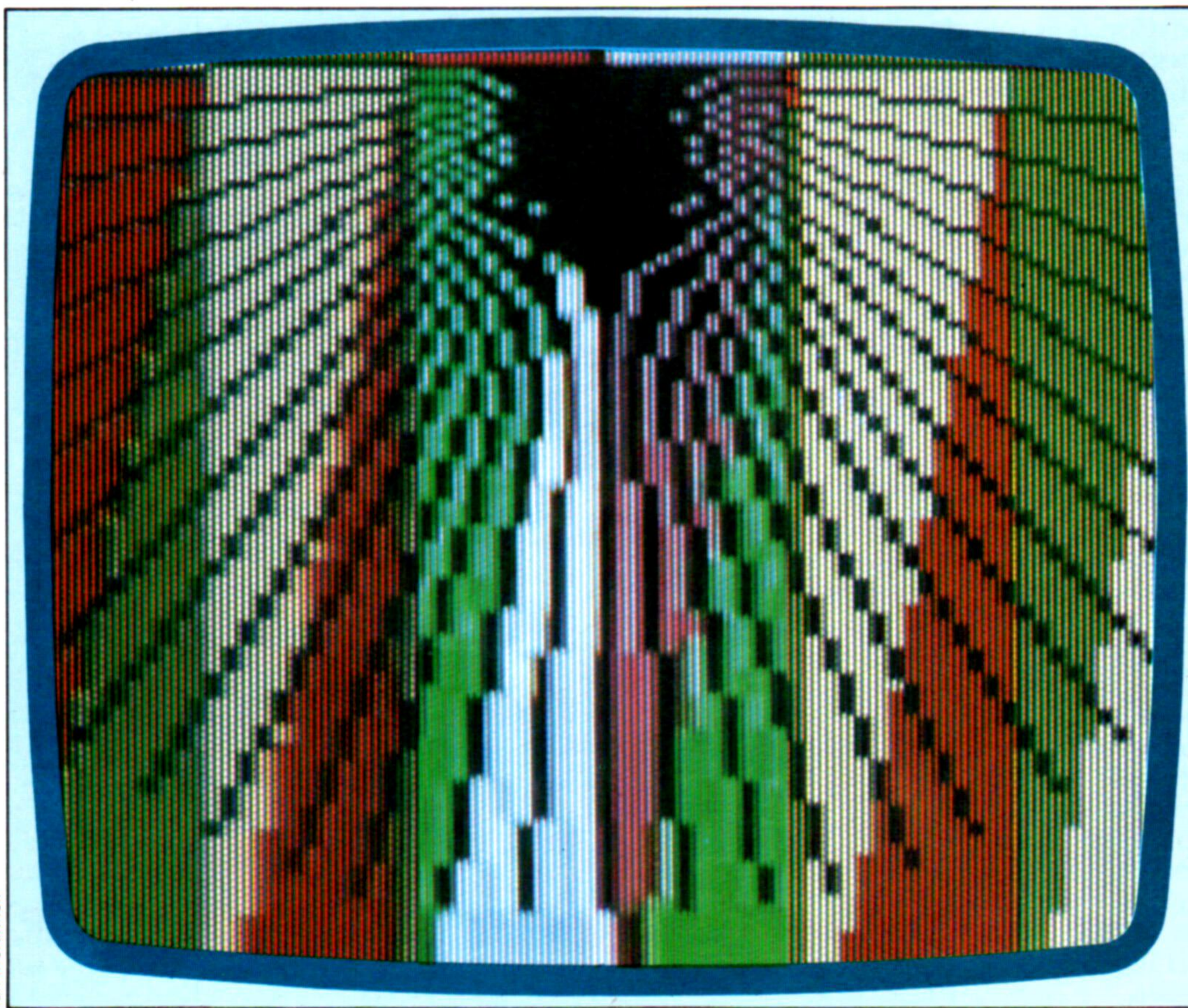
Un televisor combinado con una pantalla proporciona una visualización de gran calidad y permite, además, recibir transmisiones televisivas

La mayoría de los usuarios de ordenadores personales se acostumbra enseguida a las imágenes inestables y los colores confusos que producen sus máquinas cuando están conectadas a un televisor normal. Sin embargo, para aquellos afortunados que tienen acceso a una pantalla de ordenador, la calidad de la imagen se convierte en una revelación: los colores son claros y bien diferenciados, toda la visualización es más estable y hay una notoria ausencia de hormigueo de puntos que deben soportar los usuarios de televisores (se denomina *hormigueo de puntos* al brillo trémulo que se observa especialmente en los márgenes del texto visualizado en pantalla). Pero esta mayor calidad tiene su precio: las pantallas son más caras y no se pueden utilizar para recibir programas de televisión.

Actualmente, sin embargo, existe una tercera opción: el televisor-pantalla combinado proporciona a los usuarios lo mejor de ambas alternativas. Se compone de un aparato de televisión normal con un conector adicional que proporciona calidad de pantalla cuando se lo conecta a un microordenador. Quizá algunos usuarios ya posean uno de estos híbridos sin saberlo, ya que muchos de los televisores modernos vienen equipados con enchufes para la conexión de grabadoras de video, y éstos son igualmente apropiados para utilizarse con un micro.

El problema de emplear televisores convencionales como "visualizadores" de ordenador radica en la forma en que reciben las señales. Los programas de televisión se transmiten en forma de ondas de radio: éstas son recogidas por la antena de televisión y convertidas en imágenes. Un ordenador personal simplemente imita este proceso haciendo pasar su salida por un modulador (la pequeña caja del interior del ordenador en la cual se conecta el cable de la antena). Después de que el modulador ha alterado la señal convirtiéndola en la forma de onda de radio aceptable por el televisor, el receptor vuelve entonces a cambiarla para visualizar la imagen. Esto significa que la señal se puede degradar en dos lugares: en el modulador y dentro del propio receptor. Una pantalla omite la modulación; funciona directamente a partir de la señal primaria, proporcionando una visualización de gran calidad.

Un factor a considerar en el momento de adquirir una pantalla es el formato que utilice su ordena-



Ian McKinnell

Tres grados

Las señales de visualización que producen los ordenadores se clasifican en tres tipos principales. Todos los ordenadores personales emiten señales para televisión, pero esto a menudo significa una imagen pobre. La mayoría de los ordenadores producen, asimismo, señales para pantallas. La visualización que éstas proporcionan es mejor, pero son caras. Los televisores-pantalla combinan la calidad de una pantalla con la capacidad de captar imágenes de televisión. En cuanto a calidad, la principal diferencia entre los televisores y las pantallas es el tipo de señal con la que trabajan. Estas tres imágenes se produjeron en el mismo televisor-pantalla pero utilizando los tres tipos diferentes de señales: televisión (algunas veces llamadas RF), video compuesto y RGB. La señal de televisión (que aparece en el medio) proporciona la calidad más pobre; el video compuesto (a la izquierda) es un poco mejor, y la señal de RGB (a la derecha) es la que da mejor resultado

dor. En la actualidad hay dos tipos de señal para monitor: RGB (*Red, Green, Blue*: rojo, verde, azul) y video compuesto. El RGB da una imagen mejor, pero los dos tipos son considerablemente superiores a la salida de un televisor.

Existen, asimismo, dos clases de televisor-pantalla: los receptores de televisión normales que se han adaptado para tomar una señal para pantalla, y los aparatos construidos especialmente. Estos últimos son más adecuados, porque los aparatos convertidos a menudo se modifican sin el conocimiento del fabricante y, por lo tanto, no están amparados por ningún tipo de garantía. Los televisores-pantalla construidos especialmente están diseñados para ser utilizados de manera primordial, con grabadoras de video. Éstas por lo general disponen de entradas de video compuesto (busque un conector señalado como "video" o "audiovisual"). Los diagramas que acompañan este capítulo le mostrarán cómo conectar su ordenador (en el supuesto de que posea un modelo de video compuesto) a uno de estos conectores. Una vez hecho esto, hay que sintonizar el aparato para la visualización del ordenador de la misma forma en que seleccionaría un canal de televisión.

La principal ventaja del televisor-pantalla combinado respecto a la pantalla estándar es la facilidad de sonido. Muchos ordenadores personales (en especial los modelos Atari, Commodore y Dragon) se basan en el aparato de televisión para producir efectos sonoros. Una pantalla estándar no posee facilidades para sonido, mientras que los televisores-pantalla poseen altavoces y amplificadores incorporados.

Si su ordenador está dotado de una salida de RGB, sus opciones son más limitadas. Básicamente existen tres televisores-pantalla para entrada de RGB: el sistema Sony Profeel, los televisores con conectores Peri-TV (en particular la gama Normende) y el modelo ITT. El Sony Profeel acepta tanto señales de RGB como de video compuesto, pero utiliza un conector no estandarizado. El televisor-pantalla ITT posee un conector RGB que es compatible patilla a patilla con la salida del Oric y el Atmos, pero que también se podría utilizar con otros ordenadores RGB. El Normende es especialmente popular entre los usuarios de ordenadores personales, porque posee un conector Peri-TV. Éste es un conector de expansión para televisión, estandarizado internacionalmente, que acepta tanto señales de RGB como de video compuesto.

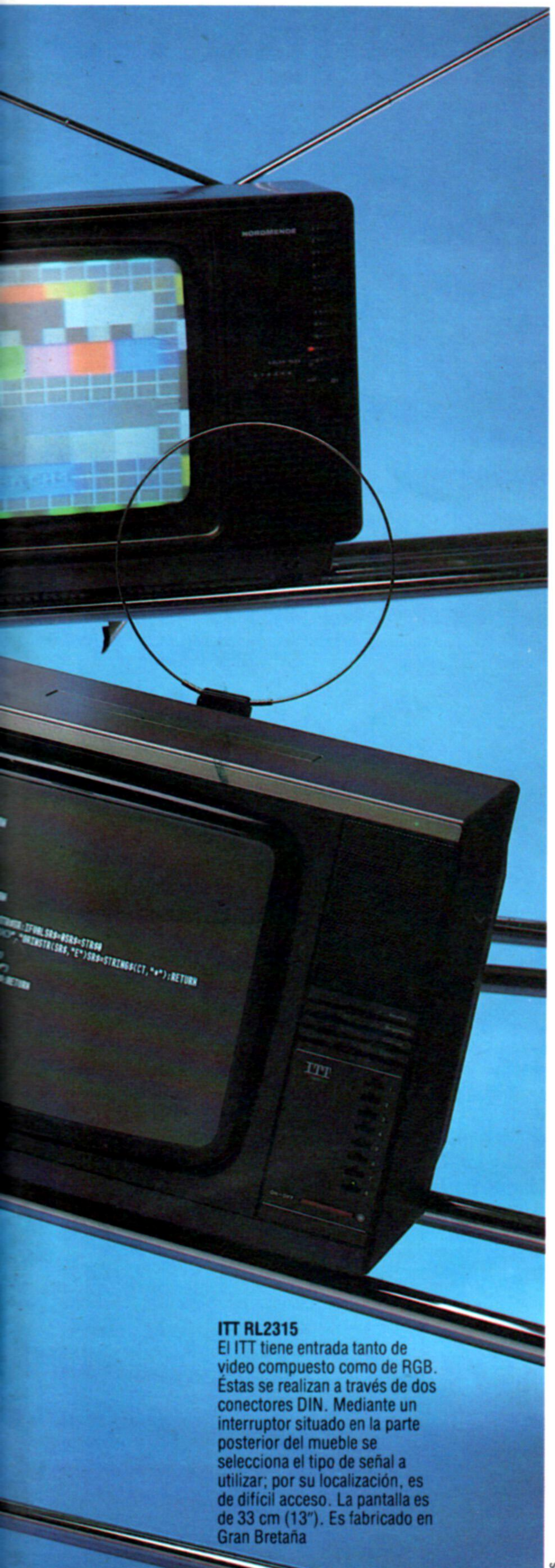
Otros aparatos de televisión se pueden equipar con entradas Peri-TV (también llamada *Scart*); compruebe, si su televisor es uno de ellos. El único problema de este sistema radica en que, en algunos televisores, para pasar de una modalidad a otra, se debe enchufar o desenchufar el conector Peri-TV. Esto es mucho menos conveniente que hacerlo conmutando los canales, tal como se haría en un receptor con conector de video. El hecho de que el sistema Peri-TV sea compatible tanto con las entradas RGB como con las de video compuesto significa que se puede conservar el mismo televisor-pantalla, aun cuando se cambie el ordenador.

Pero, independientemente del sistema escogido, el superior rendimiento de un televisor-pantalla combinado debería hacer que éste fuera el único tipo de aparato de televisión que el usuario de un ordenador adquiriera.

Ferguson TX con RGB
Este televisor-pantalla puede aceptar tanto entradas de RGB como de video compuesto a través de dos conectores DIN. Mediante los interruptores que hay en la parte delantera del TV se seleccionan los tres métodos de visualización. Están situados aquí porque la mayoría de los usuarios pasará a menudo de utilizar la unidad para mirar la televisión a trabajar con el ordenador. La pantalla es de 33 cm (13"). Se fabrica en Gran Bretaña

Normende 1534
Éste es uno de los muchos televisores Normende, todos los cuales poseen enchufes Peri y, por consiguiente, pueden aceptar señales de video compuesto o de RGB. Hay siete tamaños de pantalla diferentes disponibles y para la mayoría de los mismos existe la opción de control manual o control a distancia. El aparato que vemos aquí posee una pantalla de 33 cm (13"). Es fabricado en Singapur

Fidelity CM14
Existen dos modelos. Uno es pantalla solamente, con las entradas de RGB y de video compuesto a través de un enchufe Peri. El tamaño de la pantalla es de 33 cm (13"). Es fabricado en Gran Bretaña



ITT RL2315
El ITT tiene entrada tanto de video compuesto como de RGB. Éstas se realizan a través de dos conectores DIN. Mediante un interruptor situado en la parte posterior del mueble se selecciona el tipo de señal a utilizar; por su localización, es de difícil acceso. La pantalla es de 33 cm (13"). Es fabricado en Gran Bretaña

Chris Stevens

Conexiones correctas

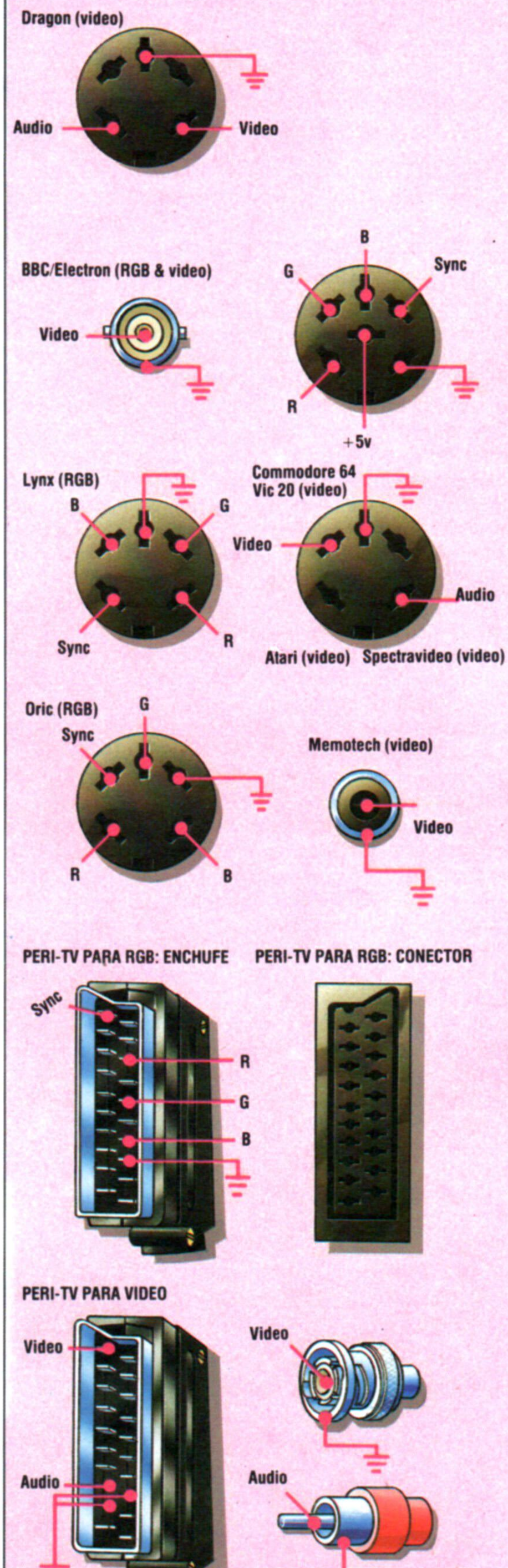
Existen dos tipos distintos de señales para pantalla: video compuesto y RGB. Las señales de RGB se componen de señales separadas rojas, verdes y azules más una señal de "sincronización". Las entradas y las salidas de un monitor RGB utilizan conectores de patillas múltiples (por lo general DIN). Una señal de video compuesto tiene todas las señales de color y la de sincronización combinadas en una sola señal. Los conectores para entrada y salida de video compuesto por lo general son conectores *phono* o BNC (tipo bayoneta). No obstante, algunos ordenadores incluyen la salida de sonido desde el mismo conector que la señal de video; en estos casos, se deben utilizar patillas múltiples.

Los diagramas ilustran las conexiones que se pueden encontrar en los ordenadores personales. Si su televisor-pantalla posee una de las conexiones ilustradas, todo lo que tiene que hacer es preparar un cable con los dos enchufes, según las conexiones dadas (p. ej., de R a R, de sync a sync, etc.).

Los televisores-pantalla con conectores Peri-TV exigen realizar la conmutación de modalidad televisión a modalidad pantalla mediante la conexión o desconexión del enchufe. Éstos suelen requerir una salida de cinco voltios desde el ordenador (como la del BBC) y algún tipo de circuito de conmutación como el que se ilustra.

Hay dos importantes ordenadores, el Sinclair Spectrum y el ZX81, que no están incluidos en la lista porque carecen de interfaces para pantalla; a pesar de ello se los puede modificar para que proporcionen una señal de video compuesto. Hay al menos una empresa que fabrica un adaptador para que el Spectrum dé una señal RGB de mejor calidad. Este adaptador se enchufa en el micro y, por lo tanto, no anula la garantía.

El Commodore Vic-20, el Atari, el Spectravideo y las primeras versiones del Commodore 64 utilizan la misma entrada para video compuesto. Los modelos más recientes del Commodore, sin embargo, emplean un enchufe DIN de ocho patillas. El cableado de los mismos se debe realizar con la patilla dos como tierra, la patilla tres como sonido y la patilla cuatro como video



Kevin Jones

Imágenes en relieve

En esta ocasión explicaremos cómo crear gráficos tridimensionales introduciendo en un programa distintas funciones matemáticas

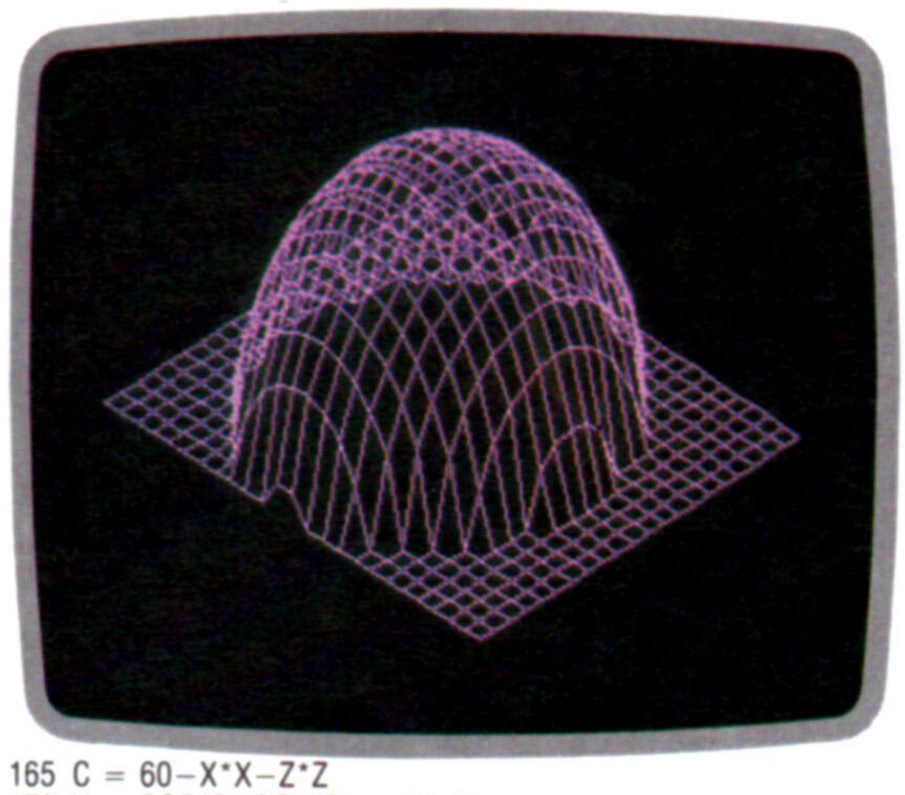
La sola mención de senos y tangentes, por no hablar de gráficos en tres planos, es suficiente para atemorizar a aquellos que se sintieron aliviados al haberse liberado de las matemáticas de la escuela. Pero con la ayuda de un ordenador estos temas se pueden convertir en una fuente de diversión, aun cuando no se comprendan cabalmente los principios sobre los que se sustentan.

La capacidad para gráficos de la mayoría de los ordenadores los hacen ideales para visualizar gráficos de ecuaciones matemáticas. Es posible que para la mayoría de nosotros dichas ecuaciones no tengan ningún significado cuando están escritas sobre el papel, pero al trazarse en forma de gráficos producen figuras muy atractivas. Incluso quienes aborrezcan las matemáticas podrían sentirse inspirados, al verlas, para realizar sus propias ecuaciones.

Todas las figuras que ilustran este capítulo se produjeron con un micro personal utilizando los programas listados. Todos están calculados como gráficos en tres dimensiones. Todo el mundo sabe el aspecto que tiene un gráfico normal en dos dimensiones. Un gráfico tridimensional se compone de varios gráficos bidimensionales visualizados al mismo tiempo, con ligeras diferencias entre cada uno de ellos. Dado que los ordenadores sólo pueden visualizar imágenes en dos dimensiones, el resultado no es auténticamente tridimensional, pero tal como se forman las imágenes se obtiene una ilusión de relieve.

Los programas listados aquí calculan los valores de una ecuación con dos variables, X y Z. El resultado, Y, se calcula para muchos valores de X y Z. Cada valor de Y se utiliza para situar un punto en la pantalla, correspondiendo los valores de Y a puntos sobre el eje vertical; o sea, cuanto más elevado es el valor de Y, más cerca del límite superior de la pantalla se trazará el punto. Los puntos vecinos se unen entre sí mediante líneas rectas, lo que proporciona un efecto de curva. Las curvas en una dirección representan gráficos de X e Y, manteniéndose Z constante, mientras que las curvas que intersectan a éstas son gráficos de Y y Z, con X manteniéndose constante (en este caso se trazan en el plano formado por los ejes Y y Z, que forman un ángulo recto respecto al plano X-Y de los gráficos bidimensionales normales). Dichas visualizaciones son útiles para comprender funciones complicadas.

Estas figuras también pueden ser un estímulo para las personas que normalmente no se sienten demasiado interesadas por las matemáticas. Es divertido (y bastante difícil) dar con la ecuación adecuada para producir una forma determinada. Para modificar el gráfico visualizado es necesario cambiar la función de la línea 170 del programa BASIC. Algunas funciones pueden ser relativamente complicadas y, por consiguiente, podrían requerir más de una línea de programa; de ser éste el caso, se podrían utilizar todas las líneas entre 151 y 179.

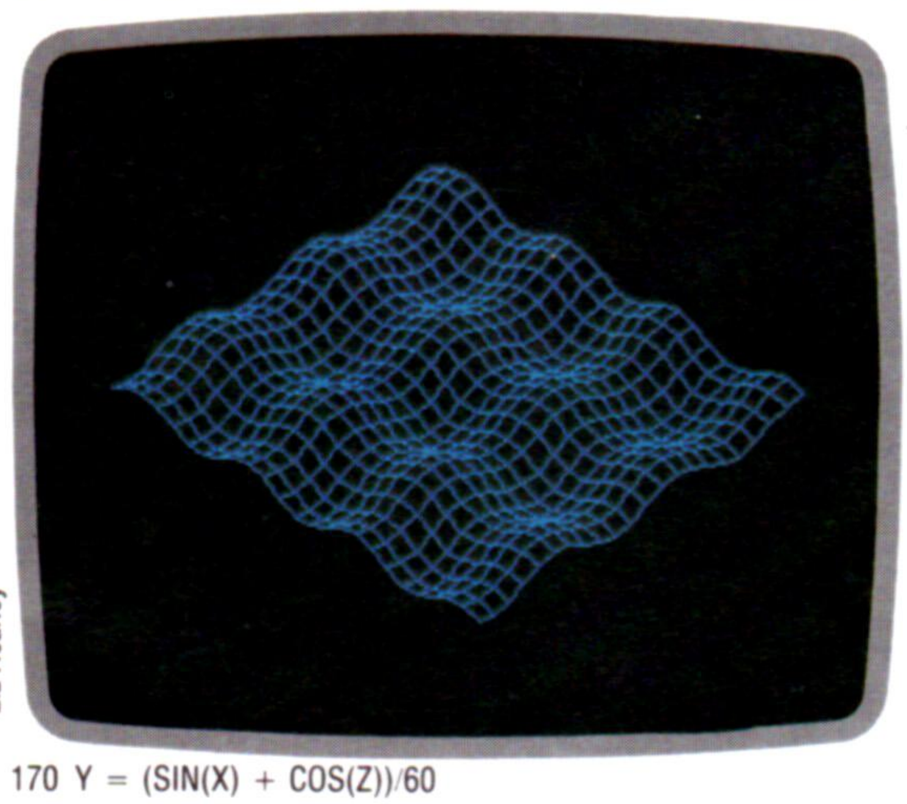


```
165 C = 60-X*X-Z*Z
170 Y = SQR(C*(SGN(C) + 1))/45
```

Además de elegir una función que produzca una forma atractiva, hay que tener cuidado en que los valores generados no sean tan grandes como para que el gráfico salga fuera de los límites de la pantalla. Para mantener la figura dentro de éstos, puede que sea necesario dividir la función por un número grande.

Posiblemente, el programa ofrecido funcionará en varios ordenadores personales. A modo de ayuda para efectuar la conversión, hemos diseñado el programa de una forma estándar. Esto significa que una ecuación que funcione en una máquina determinada también tendrá que funcionar en otras. La segunda parte del programa tiene como finalidad almacenar los valores utilizados para trazar los

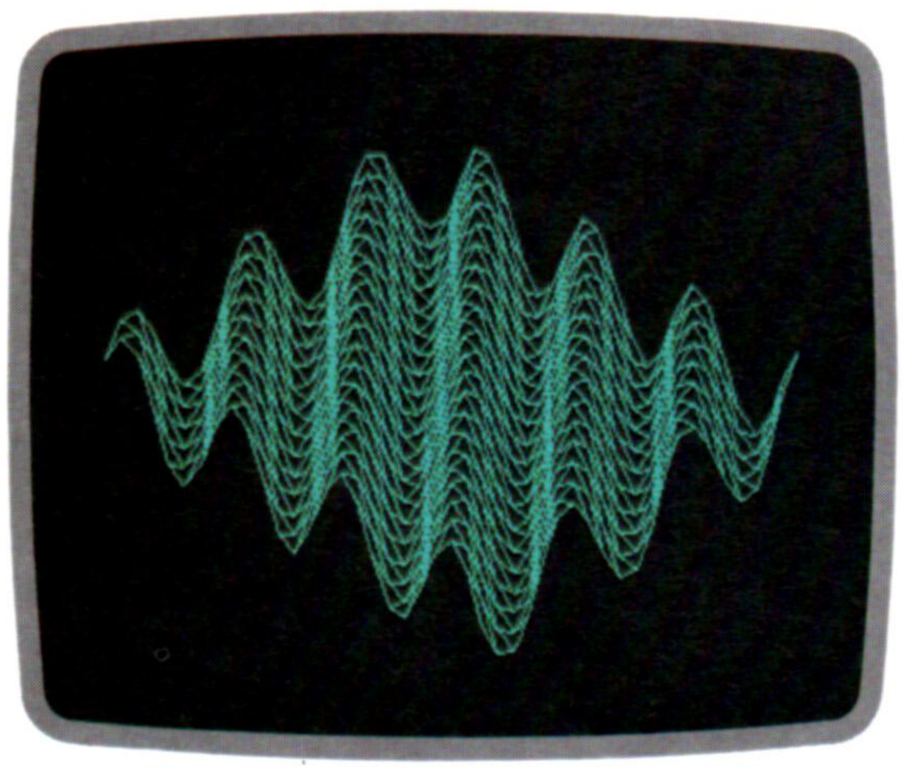
Creación de formas
Modifique el programa tal como se indica debajo de cada fotografía para producir estos gráficos tridimensionales



Liz Heaney

```
170 Y = (SIN(X) + COS(Z))/60
```

puntos en el gráfico. Estos resultados se guardan en una matriz y calcularlos requiere su tiempo. Los cálculos dependen de la función elegida y pueden



170 Y = SIN(X + Z)/12

consumir varios minutos; en este período parecerá que el ordenador está sin hacer nada. Calcular la función al principio ahorra tiempo a la larga. Si los cálculos se efectuaran mientras se están trazando las líneas, el programa tardaría casi el doble de tiempo en dibujar el gráfico.

Hemos listado varias funciones diferentes para que se experimente con ellas. Las ilustraciones reflejan los resultados que se pueden esperar. Usted también debería intentar desarrollar sus propios gráficos entrando al programa distintas funciones. Tenga cuidado al hacer esto; asegúrese de que el gráfico quepa en la pantalla y no intente ninguna operación matemática ilegal. Los dos errores más comunes son intentar dividir por cero (lo que da infinito) e intentar hallar la raíz cuadrada de un número negativo (lo que no existe).

Para evitar la división por cero, súmele una constante muy pequeña (p. ej., 0,00001) a cualquier variable que pueda convertirse en cero. La única forma de protegerse contra la raíz cuadrada de los números negativos es emplear la función ABS para hacer que todos los números sean positivos antes de hallar sus raíces cuadradas.

Se pueden producir algunas visualizaciones interesantes mediante funciones matemáticas comunes como SIN, COS, LOG, etc. Otras se pueden obtener utilizando funciones de las que sólo disponen los ordenadores (pruebe con INT, SGN y ABS).

Este programa se podría mejorar de muchas maneras. Trate de adaptarlo de modo que cualquier función gradúe automáticamente sus valores para ajustarse a los límites de la pantalla, o bien intente trazar puntos en una tercera dirección, dando curvas para X y Z mientras se mantiene a Y constante (esto es relativamente complicado). Pero aun si usted simplemente utiliza el programa tal como está escrito aquí, descubrirá lo entretenido que es probar con las ecuaciones más absurdas que se le ocurran. Quizá los resultados le sorprendan.

165 C = X*X + Z*Z + 0.00001
170 Y = SGN(INT(23/C))/3 + SGN(INT(55/C))/15

Este programa está escrito en BASIC BBC.

```
 10 REM *TRAZADO DE GRAFICOS
 20 :
 30 REM *PREPARAR PANTALLA
 40 ACROSS = 1280:TAL = 1024:UP = -1
 50 XGAP = 25:ZGAP = 15
 60 WIDE = INT(ACROSS/XGAP/2)
 70 DEPTH = INT(TALL/ZGAP/3)
 80 MODE4:CLS:PRINTTAB(12)"CALCULANDO"
 90:
100 REM *CALCULAR GRAFICO
110 START = 20
120 DIM G(WIDE,DEPTH)
130 FOR A = -DEPTH/2 TO DEPTH/2
140 FOR B = -WIDE/2 TO WIDE/2
150 X = A*20/WIDE:Z = B*20/DEPTH
160 REM *INSERTAR FUNCION AQUI ABAJO
170 Y = (SIN(X) + COS(Z))/60
180 G(B + WIDE/2,A + DEPTH/2) = Y*UP*TALL
190 NEXT B:NEXT A:CLS
200 :
210 REM *DIBUJAR GRAFICO; PLANO X-Y
220 FOR Z = 1 TO DEPTH
230 XBASE = XGAP*Z
240 ZBASE = TALL/2 + Z*ZGAP + START*UP
250 XOLD = XBASE + XGAP
260 ZOLD = ZBASE-ZGAP-G(1,Z)
270 FOR X = 1 TO WIDE
280 XNEW = XBASE + X*XGAP
290 ZNEW = ZBASE-X *ZGAP-G(X,Z)
300 PLOT 4,XOLD,ZOLD:PLOT 5,XNEW,ZNEW
310 XOLD = XNEW:ZOLD = ZNEW
320 NEXT X:NEXT Z
330 :
340 REM *DIBUJAR GRAFICO; PLANO Z-Y
350 FOR X = 1 TO WIDE
360 XBASE = XGAP*X + DEPTH*XGAP
370 ZBASE = TALL/2-X*ZGAP + DEPTH*ZGAP + START*UP
380 ZOLD = ZBASE-ZGAP-G(X,DEPTH-1)
390 XOLD = XBASE-XGAP
400 FOR Z = 0 TO DEPTH-1
410 XNEW = XBASE-Z*XGAP
420 ZNEW = ZBASE-Z*ZGAP-G(X,DEPTH-Z)
430 PLOT 4,XOLD,ZOLD:PLOT 5,XNEW,ZNEW
440 XOLD = XNEW:ZOLD = ZNEW
450 NEXT Z: NEXT X
460 :
470 REM *MANTENER VISUALIZACION
480 GOTO 470
```

Complementos al BASIC

Spectrum

Insertar LET en todas las sentencias de asignación.
Insertar las siguientes líneas:

```
 40 LET ACROSS = 256:LET TALL = 176:LET UP = -1
 50 LET XGAP = 5:LET ZGAP = 3
 80 CLS
290 PLOT XOLD,ZOLD:DRAW XNEW-XOLD,ZNEW-ZOLD
410 PLOT XOLD, ZOLD:DRAW XNEW-XOLD,ZNEW-ZOLD
```

Oric-1/Atmos

Insertar las siguientes líneas:

```
 40 ACROSS = 239:TALL = 199:UP = 1
 50 XGAP = 5:ZGAP = 3
 80 HIRES
300 CURSET XOLD,ZOLD,1:DRAW XNEW-XOLD,ZNEW-ZOLD,1
430 CURSET XOLD,ZOLD,1:DRAW XNEW-XOLD,ZNEW-ZOLD,1
```

Dar en la diana

Iniciamos una serie de capítulos que pretenden dar a conocer las técnicas que utilizan los buenos programadores

La buena programación se desarrolla mediante la experimentación y la experiencia. El programador novato, que suele resolver los problemas gracias a un enorme entusiasmo y un inmenso esfuerzo, se va transformando gradualmente en un técnico consciente de los métodos expeditos y empíricos que consiguen los resultados deseados. Finalmente, el programador desarrollará la claridad y el enfoque directo del experto. Pero no existe ninguna razón por la que no se pueda acelerar el progreso personal de un programador de ordenadores personales mediante el aprendizaje en función de los errores de otros que han seguido su mismo camino. Las lecciones están allí, para aprenderlas, y la programación de todos se puede beneficiar con ellas. Nuestro curso comienza con un análisis de algunas de las pistas más útiles que pueden serle de ayuda al principiante.

La programación es un proceso para resolver problemas, y gran parte de ella se debe llevar a cabo en la mente y con lápiz y papel mucho antes de que se pueda escribir una línea de código. Las etapas de este proceso son bien conocidas: un claro y amplio enunciado del problema en términos prácticos, repetido reiteradas veces con creciente precisión, hasta formularlo con tanto detalle y exactitud como sea posible. Esta descripción casi siempre contiene o implica la solución esencial, que se debe entonces exponer con mayor detalle y de forma más práctica de modo que se convierta en un método de trabajo. En programación, sólo la última etapa incluye codificación y ésta debe ser una concreción directa de las etapas precedentes. Cuando la etapa de codificación se superpone con la verdadera solución del problema, el resultado son soluciones pobres y código malo.

A las soluciones se las suele denominar *algoritmos*, procesos de cálculo analizados en etapas lógicas. La eficiencia de un programa depende básicamente de la eficiencia de su algoritmo, y éste se juzga en función de lo completo que sea y de su corrección. Estas cualidades de puro sentido común se refieren a la capacidad teórica y práctica del programa para hacer frente a la gama previsible de condiciones de entrada y a la coherencia de su lógica interna. Huelga decir que es mucho más fácil reconocer su ausencia que demostrar su presencia, pero todos los problemas deben someterse a esta valoración y lo mejor es realizarla en la etapa más temprana posible de su desarrollo.

Las soluciones deben ser *fiables*, además de completas y correctas. No sólo deben tratar el ámbito de problemas prescritos, sino que también deben funcionar de forma predecible y segura en condiciones fuera de su ámbito. Esto por lo general significa posser la capacidad de reconocer condiciones de error potenciales y ser capaz de detener la operación con todos los datos intactos, así como visualizar algunos mensajes de estado significativos. Juzgar si un programa es suficientemente fiable es difícil: es más fácil reconocer un programa que no es fiable que reconocer uno que sí lo es. La experiencia es la fuente de una mejor valoración.

Hacer que los programas sean fiables y sólidos es una tendencia meritoria que casi siempre entra en conflicto directo con un objetivo que es igualmente deseable: lograr que los programas sean *económicos*. Todo cuesta dinero, aun cuando sólo se trate del tiempo que se tarda en escribir programas como mera diversión. Siempre llega un momento en el que uno tiene que decidir entre seguir trabajando en un programa que parece "a prueba de todo", o abandonarlo para comenzar un proyecto nuevo. Incluso aunque se disponga de tiempo ilimitado, la memoria y la velocidad de operación del ordenador no lo son. Es bastante posible rodear el algoritmo central con tanto código preventivo y de detección de errores, que protegerlo contra casos extraños pueda ocupar más tiempo que resolver el problema original.

Verificación y depuración

Resolver problemas lógicos y analíticos en teoría es sumamente importante, pero los programas están hechos para realizar una tarea. Después de solucionar los primeros errores lógicos y de sintaxis, es hora de empezar la *verificación*. Ésta es una idea tan familiar que apenas si parecería merecer alguna explicación, por no hablar de hacer hincapié en ella. Pero, en realidad, es un proceso que no se suele comprender bien. En todos los programas, exceptuando los triviales, suelen existir demasiadas posibles combinaciones de las condiciones de entrada como para poder realizar pruebas exhaustivas, de modo que se deben poner a punto juegos de ensayo para comprobar al máximo las partes del programa más vulnerables (y que han de ser las más fuertes). Generar juegos de ensayo exhaustivo no es una labor sencilla y lleva tiempo y dinero. El enfoque profesional a la verificación es que no existen programas perfectos, sino pruebas malas.

Las pruebas adecuadas revelan las insuficiencias de un programa, y deben hacerlo de una forma lógica, de modo que la *depuración (debugging)* consuma el menor tiempo posible. Al igual que la verificación, la depuración, o eliminación de errores, es un proceso esencial que fracasa muy a menudo precisamente porque personifica los fallos humanos que la hacen necesaria. Un error en un programa se debe enfocar como otro problema a resolver, exactamente como hemos descrito antes (enunciación, análisis, algoritmo, verificación), pero lo más frecuente es abordarlo como una plaga a machacar, envenenar o pisotear, con consecuencias predeciblemente desastrosas para las áreas circundantes.

A medida que se van completando estas etapas de desarrollo, la familiaridad y la satisfacción se van combinando para convencer al programador de que el programa funciona ahora, funcionará siempre y no requerirá jamás ninguna modificación y que, de cualquier forma, el código es un modelo de claridad. Pero los programadores, y no los programas, necesitan documentación. No hay ningún programa que sea autoexplicativo y siempre existen razones para cambiar programas que ya funcionen. Al igual que cualquier otro mecanismo, necesitan *mantenimiento*, es decir, manuales. Los programas han de estar documentados internamente (utilizando líneas REM) para beneficio del programador, y documentados exteriormente con literatura alusiva por el bien del usuario, aun cuando éste sea el propio programador.

Todas estas lecciones las tuvieron que aprender una vez los programadores de ordenadores centrales y han sido ignoradas y dolorosamente redescubiertas por los programadores de microordenadores. Tomadas en conjunto constituyen una "estructura" de programación, un enfoque unificado a la solución de problemas mucho más amplio que un libro de consejos relativos a evitar los GOTO o a adoptar WHILE...WEND. Los programas eficientes los escriben programadores eficientes, por supuesto, pero en todo caso sobre la base de la experiencia estructurada y el pensamiento lógico. Esta serie tiene por objeto fomentar ambos.

Ian McKinnell

Diagramas de barras
Los colores y la profundidad de las barras que conforman el diagrama se ajustan fácilmente por programa cambiando los valores de las variables de control

```
399 REM ************************************
400 REM*GRAFICOS DE BARRAS TRIDIMENSIONALES*
401 REM ************************************
500 GOSUB 1500:          REM INICIALIZAR
720 YY = 22:XX = 2:      REM ORIGEN COORD
760 GOSUB 3200:          REM DIBUJAR EJES
800 FOR E = LT - 1 TO 0 STEP -1
820 OC = XX + E*DB:OL = YY-E*DB
840 GOSUB 2200:          REM INPUT DATOS
900 FOR D = 1 TO NN
920 HT = DT(D)
940 X0 = OC + (D-1)*(BB + LT*DB):Y0 = OL-HT-DB
960 GOSUB 4000:          REM IMPRIMIR BARRA
980 NEXT D
1000 NEXT E
1100 XP = 10:YP = 23:GOSUB 3500
1120 PRINT"HISTOGRAMA TRIDIMENSIONAL"
1200  A$ = INKEY$:IF A$ = "" THEN GOTO 1200
1400 END
1499 REM ************************************
1500 REM*INICIALIZAR S/R                    *
1501 REM ************************************
1520 CLS = CHR$(147):    REM LIMPIAR PANTALLA
1540 PRINT CL$
1560 PO$ = CHR$(19):     REM SITUAR CURSOR
1580 RT$ = CHR$(13):     REM <RETURN>
1600 BB = 2:DB = 1:      REM DIMENSIONES BARRAS
1620 SW = 40:SD = 25:    REM DIMENSIONES PANT.
1640 HB = SD-DB:         REM ALT. MAX. BARRAS
1660 DIM B$(HB + DB)
1680 FOR K = 1 TO SD:PO$ = PO$ + RT$:NEXT K
1800 DIM DT(SW)
1900 GOSUB 2400:         REM CONSTRUIR BARRA
2100 LT = 4:             REM FACTOR PROFUNDIDAD
2190 RETURN
2199 REM ************************************
2200 REM*        ENTRAR DATOS S/R MATRIZ     *
2201 REM ************************************
2220 READ NN
2240 FOR Z = 1 TO NN:READ DT(Z):NEXT Z
2310 DATA 6,12,10,4,7,8,10
2320 DATA 5,7,8,8,6,7
2330 DATA 6,7,4,8,5,3,9
2340 DATA 5,11,6,4,11,6
2390 RETURN
2399 REM ************************************
2400 REM*      S/R CONSTRUIR BARRA ENTERA    *
2401 REM ************************************
2500 TC$ = CHR$(158):    REM LADOS = AMARILLO
2520 FC$ = CHR$(31):     REM FRENTE = AZUL
2540 RV$ = CHR$(18):     REM ACTIVAR REVERSE
2560 NL$ = CHR$(146):    REM DESACTIVAR REVERSE
2580 CR$ = CHR$(29):     REM CURSOR DERECHA
2600 CH$ = CHR$(32):     REM CAR. ESPACIO
2620 C1$ = CHR$(169):    REM CAR. "◤"
2640 C2$ = RV$ + C1$:    REM CAR. "◢"
2680 FOR K = 1 TO SW
2700 SP$ = SP$ + CH$
2720 RC$ = RC$ + CR$
2740 FF$ = FF$ + CH$
2760 NEXT K
2800 TL$ = SP$ + "/"
2820 BL$ = SP$ + NL$ + C1$
2840 FL$ = LEFT$(FF$,BB)
2900 L$ = TC$ + C2$ + RV$ + RIGHT$(TL$,BB)
2920 FOR K = 1 TO DB
2940 B$(K) = LEFT$(RC$,DB-K) + L$ + LEFT$(SP$,K-1)
2960 NEXT K
3000 L$ = FC$ + RV$ + FL$ + TC$ + RIGHT$(TL$,DB)
3020 FOR K = DB + 1 TO HB
3040 B$(K) = L$
3060 NEXT K
3100 L$ = FC$ + RV$ + FL$ + TC$
3120 FOR K = 1 TO DB
3140 B$(HB + K) = L$ + RIGHT$(BL$,DB + 2-K)
3160 NEXT K
3190 RETURN
3199 REM ************************************
3200 REM*          DIBUJAR EJES S/R          *
3201 REM ************************************
3300 PRINT TC$:           REM ESTABLECER COLOR
3320 FOR Y = 2 TO YY-1
3340 XP = XX-1:YP = Y:GOSUB 3500:PRINT"!"
3360 XP = XX + YY-Y:GOSUB 3500:PRINT"/"
3380 NEXT Y
3400 YP = YY
3420 FOR X = XX-1 TO SW-1
3440 XP = X:GOSUB 3500:PRINT"-"
3460 NEXT X
3490 RETURN
3499 REM ************************************
3500 REM*       PONER CURSOR @XP,YP S/R      *
3501 REM ************************************
3600 PRINT LEFT$(PO$,YP)TAB(XP-1);
3620 RETURN
3999 REM ************************************
4000 REM*       IMPRIMIR BARRA PARCIAL S/R   *
4001 REM ************************************
4100 FOR V = 1 TO HT
4120 XP = X0:YP = Y0 + V-1
4140 GOSUB 3500           REM COLOCAR CURSOR
4160 PRINT B$(V)
4180 NEXT V
4200 FOR V = 1 TO DB
4220 XP = X0:YP = Y0 + HT + V-1:GOSUB 3500
4240 PRINT B$(HB + V)
4260 NEXT V
4490 RETURN
READY.
```

Estructura
Modular y razonablemente autoexplicativa

Documentación
Obviamente esta rutina es crucial, pero está completamente sin explicar

Nombres de las variables
Bien comentadas, pero los nombres no son muy significativos

"Completitud"-corrección
¿Funciona esto para todos los valores? ¿Produce la salida correcta en todos los casos, por ejemplo, cuando un valor es menor que cero?

Detección de errores
Ninguna suma de control, ninguna graduación, ningún tratamiento de errores

Documentación
Buena: no hay "números mágicos" ni caracteres de control misteriosos: todo es traducible

Cal y arena

Este programa para visualizar gráficos de barras tridimensionales es una molesta mezcla de buen y mal estilo: la documentación interna es buena donde existe y la estructura es modular, pero el programa no es autoexplicativo, no hay detección de errores y ninguna documentación para el usuario

Complementos al BASIC

Este programa está escrito en BASIC Microsoft y funcionará sin ninguna modificación en micros con una visualización en pantalla de 40×25. Las dimensiones de la pantalla se establecen en la línea 1620. Los valores de los caracteres de control inicializados en las subrutinas 1500 y 2400 son para el Commodore 64; consulte la tabla ASCII en su manual para otras máquinas

Aventura medieval

Analizamos aquí un interesante juego de aventuras: "Twin kingdom valley" (El valle de los dos reyes), creado por Bug-Byte

Los juegos de aventuras para ordenadores derivaron originalmente del juego *Dungeons and dragons* (Calabozos y dragones). En los años sesenta los programadores de ordenadores centrales comenzaron a desarrollar las primeras versiones para ordenador, utilizando las enormes cantidades de memoria disponible para almacenar detalles de un complicado mundo fantástico lleno de hechiceros y monstruos, enanos y gnomos. Todas las aventuras actuales para microordenadores descienden de aquellos primeros ejemplos, pero están ambientadas en una gama de lugares mucho más amplia, desde naves espaciales abandonadas a calles de Chicago en los años treinta, la época de los gángsters.

Pero todas las buenas aventuras tienen una característica común: deben hacer que el jugador sienta que el mundo de fantasía es real. Las mejores aventuras son casi como novelas, implicando por completo al jugador en las situaciones descritas. Originalmente, todas las aventuras eran solamente de textos, pero la nueva generación emplea gráficos de alta resolución para conferir a los juegos una sensación adicional de realismo. La primera aventura con gráficos que obtuvo un gran éxito de ventas fue *The Hobbit*, basado en la novela del mismo nombre de J. R. R. Tolkien. La aventura que vamos a analizar en este capítulo, *Twin kingdom valley* (El valle de los dos reyes), utiliza gráficos en un escenario tradicional de aventuras, con bosques y castillos medievales.

Si bien los gráficos le suelen conferir a la aventura una cierta brillantez, debe decirse que por sí solos no pueden disimular una falta de imaginación por parte del programador, y esto es lo que sucede con *Twin kingdom valley*. La historia en que se basa el juego es muy sencilla. El jugador asume el papel de un vagabundo que se adentra en el valle, que está gobernado por dos reyes en perpetua lucha (el rey del desierto y el rey del bosque). En el valle hay varios ríos (todos los cuales tienen un aspecto parecido) que confluyen en el lago mágico Watersmeet. Al recorrer el reino, el jugador (una especie de héroe mercenario) debe recoger la mayor cantidad posible de tesoros. Cuando se han acumulado los suficientes y el marcador del jugador ha llegado a 1 024, sucede algo sorprendente (no vamos a estropear el interés del juego revelándole de qué se trata).

El movimiento y las acciones se controlan mediante instrucciones. El programa acepta 23 verbos, que se combinan con sustantivos que aluden a objetos del juego. Se aceptará una instrucción como "golpear al guardián con el martillo", siempre que se esté en posesión de un martillo y que haya un guardián cerca. Las direcciones se indican mediante los puntos de la brújula, más las palabras *up* (arriba) y *down* (abajo). En el reino habitan otros personajes y se puede emplear el verbo *ask* (pedir) para adquirir sus posesiones. En la mayoría de los casos, sin embargo, si intenta dirigirles la palabra obtendrá sin ningún motivo una respuesta violenta.

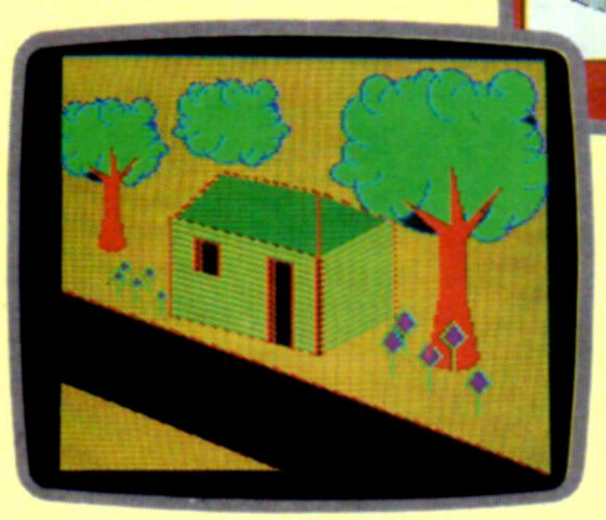
La función de los gráficos es ilustrar los 175 lugares en que se desarrolla el juego. El BBC Micro, en particular, utiliza una gran cantidad de memoria para producir visualizaciones de alta resolución, de modo que la mayor parte de las imágenes se componen de distintas combinaciones de las mismas formas básicas; por ejemplo, un bosque consta de 10 o 12 formas de árboles repetidas formando distintos patrones. En el caso del Commodore, su mayor memoria y sus gráficos tipo sprite permiten la animación en algunas pantallas, con ardillas trepando por los árboles y el agua goteando por las estalactitas. Los gráficos se pueden omitir, pero aun así utilizan un espacio de memoria que se podría haber empleado para hacer que la aventura resultara más emocionante.

Twin kingdom valley sólo es moderadamente difícil de resolver y en cuanto a concepto, apenas si es original. Existen muchas otras aventuras de texto solamente que son mucho más complicadas, si bien, por otra parte, le proporcionan al jugador una sensación mucho más intensa de vivencia en el mundo que describen.

Reinos de experiencia

Las visualizaciones del BBC Micro suelen ser distintas combinaciones de figuras básicas, mientras que al Commodore 64 su mayor capacidad de memoria y sus gráficos tipo sprite le permiten más variedad y algo de movimiento

Commodore 64



BBC Micro

Ian McKinnell

Twin kingdom valley: Para el BBC Micro y el Commodore 64
Editado por: Bug-Byte Software, Mulberry House, Canning Place, Liverpool LI 8JB
Autor: Trevor Hall
Palancas de mando: No se necesitan
Formato: Cassette

Cómo usar el pixel

Comenzamos con una interesante aplicación del lenguaje máquina: la pantalla en alta resolución del Commodore 64

Ya conocemos los diversos procedimientos y pasos para obtener gráficos de alta resolución en un Commodore (véase p. 734). Sabemos que ante todo debemos activar el chip de la interface para video (el VIC) poniéndolo en la modalidad de alta resolución, y cambiar el puntero de la dirección de base del juego de caracteres. Debemos asimismo limpiar el bloque de 8 Kbytes que empieza en la posición 8192 que albergará el mapa de la pantalla, e inicializar después el mapa de la pantalla normal (posiciones 1024 a 2033, o sea, $0400 a $07E7), que va a ser usado para contener la información del color de la pantalla. Tarea esta última algo complicada en visualizaciones polícromas, pero que a nosotros nos resultará trivial pues nos limitaremos a un solo color de fondo en la pantalla.

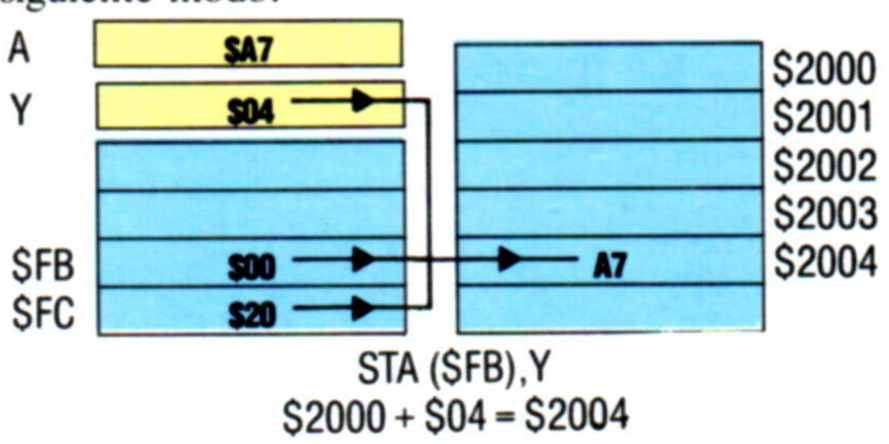
El programa debe permitir la activación y la desactivación de la modalidad de alta resolución, realizando posteriormente los cálculos necesarios para trazar un punto sobre la pantalla en alta resolución. Por tanto, la primera parte de dicho programa se va a ocupar de dos importantes tareas previas al uso de la alta resolución: colocar la información del color en cada una de las posiciones de la pantalla normal, y limpiar el mapa de bits de 8 Kbytes.

Para poder llamar a la misma rutina al activar o desactivar la modalidad de alta resolución se empleará un flag o indicador, que llamaremos HRSFLG (en inglés, **H***igh* **R***e***S***olution* **FL***a***G**). Ahora bien, no siempre desearemos limpiar el mapa de bits al entrar en modalidad de alta resolución, sobre todo si queremos que en la pantalla permanezca una figura anteriormente dibujada. Para esto, un segundo flag, el CLRFLG (en inglés, **CL***ea***R FL***a***G**), servirá de indicador si deseamos limpiar la pantalla o no. A la derecha encontrará usted el diagrama de flujo que emplea estos dos flags dentro de la rutina en lenguaje máquina.

Vayamos ahora a la tarea relativamente fácil de acceder, por medio de un bucle en lenguaje máquina, a bloques de memoria de 256 bytes o menos. El siguiente fragmento de programa coloca el número $03 en cada una de las posiciones que van desde la dirección BASE a la dirección BASE + 255 (en total, 256 posiciones), empleando un direccionamiento indexado absoluto (véase p. 676).

```
      LDY $00
      LDA $03
NEXT  STA BASE,Y
      DEY
      BNE NEXT
```

Observe que con esta técnica se accede primero a BASE, y al bloque restante se accede descendiendo desde BASE + 255 hasta BASE + 1. Pero nuestro programa nos va a pedir el acceso a más de un bloque de 256 bytes, por lo que debemos servirnos también del direccionamiento llamado indirecto postindexado. El sistema operativo del Commodore 64 necesita casi toda la página cero para él, pero deja unos cuantos bytes a disposición del programa de lenguaje máquina, por ejemplo, los bytes 251 y 252 (o sea, $FB y $FC). En este tipo de direccionamiento, el ordenador asume que el byte *lo* de la dirección en cuestión se encuentra en la posición de la página cero especificada, y el byte *hi* en la posición siguiente, siempre en esa misma página cero. Luego una instrucción como STA ($FB),Y, suponiendo que $FB y $FC contienen $00 y $20, e Y contiene por su parte $04, calculará la dirección pedida del siguiente modo:



STA ($FB),Y
$2000 + $04 = $2004

Un método similar es el que puede servirnos para acceder a un bloque entero de 256 bytes de la memoria. La potencia de este procedimiento se basa en la posibilidad de acceder al byte *hi* de la dirección BASE. Un incremento de una unidad en este byte equivale a un incremento de la dirección BASE en 256 (lo que quiere decir el comienzo del bloque siguiente de 256 bytes de la memoria). Esta técnica puede resultarnos útil en las tareas de colocar la información del color en el área de la pantalla normal o de limpiar el área del mapa de bits.

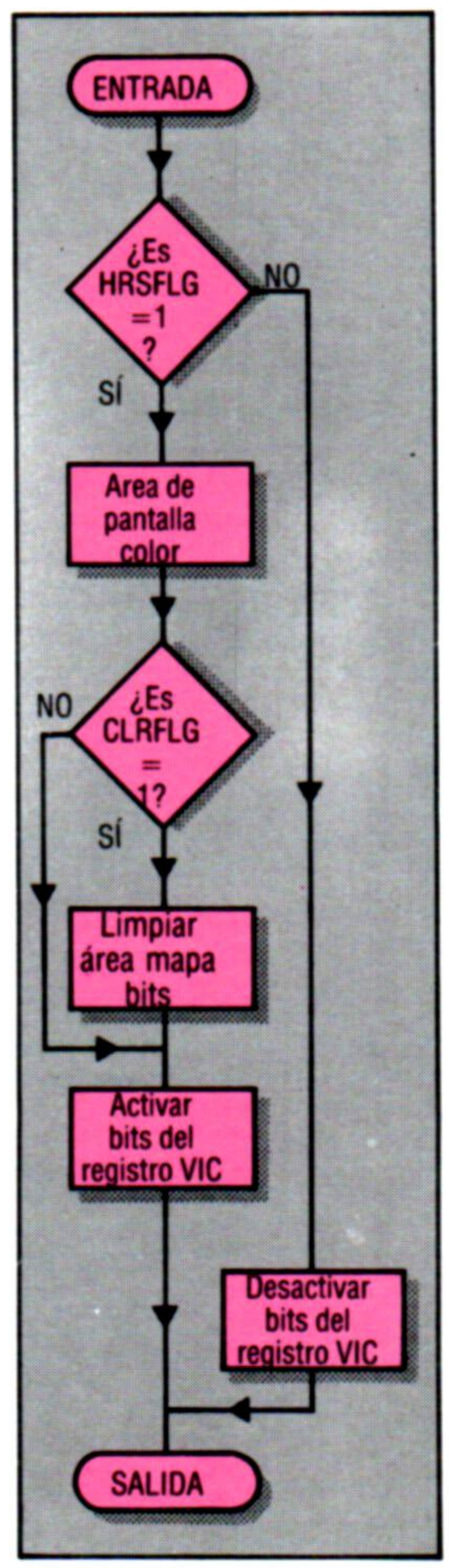
El área de la pantalla se extiende desde $0400 hasta $07E7. Es decir, se compone de tres bloques de 256 bytes más un resto de $E7 bytes. El trozo de assembly que denominamos "Área colores pantalla" en la p. 819 emplea el direccionamiento indirecto postindexado para poner en cada byte una información del color. Se sirve de las variables SCBLO y SCBHI que representan los bytes *lo* y *hi* de la dirección inicial de la memoria correspondiente a la pantalla normal SC, así como de SCBLK y SCBREM para el número de bloques de 256 bytes y el resto (en inglés, **BL***oc***K** y **REM***ainder*). Por último, PTR es la posición en la página cero donde almacenaremos el byte *lo* de la dirección base.

Cálculo de la posición de un pixel

La segunda parte de esta rutina en lenguaje máquina trata del cálculo del bit correspondiente a unas coordenadas (x,y) específicas dentro del área que contiene el mapa de bits. En una pantalla de alta

La rutina A. RES

Realiza tres tareas diferentes. Establece la información del color de la pantalla, limpia (*clear*) el área del mapa de bits y activa y desactiva los bits del registro de A. RES. (en inglés, HIRES), según sea el estado de los flags HRSFLG y CLRFLG



resolución, la abscisa *x* adquiere valores que van del 0 al 319, y la ordenada *y*, del 0 al 199. Estos últimos valores de *y* pueden ser representados por un byte, pero los de *x* necesitan dos, ya que pueden superar el valor 255. En BASIC el bit correspondiente a una *x* e *y* determinadas se calculaba tal como se expone a continuación:

```
1000 HB = XAND248:VBYTE = INT(Y/8)
1010 RMY = YAND7:RMX = XAND7
1020 ROW = VBYTE*320+HB
1030 BYTE = BASE+ROW+RMY
1040 POKEBYTE,PEEK(BYTE)OR(2 ↑ (7−RMX)
```

Los mismos cálculos debe realizar ahora la rutina en lenguaje máquina. El proceso aritmético se dificulta, ya que a veces emplea dos bytes HB, ROW, BASE y BYTE. Hay dos fragmentos de la codificación que deben ser explicados: la división de *y* por ocho y la multipliación de VBYTE por 320. Es fácil dividir por potencias de dos:

```
45          = 00101101
45/2 = 22   = 00010110
22/2 = 11   = 00001011
11/2 =  5   = 00000101
5/2  =  2   = 00000010
2/2  =  1   = 00000001
1/2  =  0   = 00000000
```

Observando los resultados binarios en cada división sucesiva por 2, vemos cómo los bits que forman el resultado binario van desplazándose hacia la derecha hasta desaparecer. De donde se deduce que la división por 8 se puede obtener mediante tres "desplazamientos lógicos a la derecha" (LSR:*Logical Shifts Right*). Y como sólo necesitamos la parte entera del resultado, podemos muy bien despreciar todo bit que "caiga" por el extremo derecho del número. La operación LSR coloca, no obstante, el bit desechado en el flag de arrastre, por lo que podemos utilizarlo si hace falta. Es fácil adivinar que la multiplicación por 2 se realiza desplazando un lugar a la izquierda el número binario (operación ASL). Esto nos permite crear una rutina para multiplicar VBYTE por 320. Debemos tomar nota de que $320 = 5\times64 = 5\times2\times2\times2\times2\times2\times2$. De aquí deducimos que si sumamos cinco veces VBYTE consigo misma y después realizamos seis operaciones ASL, habremos conseguido el producto deseado: VBYTE por 320. Sólo hay que tener en cuenta que el resultado puede superar el número 255 y, por tanto, necesitamos dos bytes.

Para concluir diremos que las dos rutinas estudiadas pueden ser llamadas por un programa BASIC por medio de la instrucción SYS y que es importante asegurarse de que, una vez concluida la rutina, continuará el programa BASIC. Durante la ejecución de un programa BASIC el intérprete hace uso de los registros X, Y y A, que son también usados por las rutinas en lenguaje máquina; por consiguiente, debemos salvar sus contenidos al iniciar dichas rutinas y restaurarlos al terminarlas. Para esto nada mejor que usar la pila. Los valores de Y, X y A se colocan en la pila al entrar en la rutina en lenguaje máquina y se recuperan de ella poco antes de abandonarla. Los valores de las coordenadas, de los colores y de los flags se pasan al programa en lenguaje máquina mediante POKE, que los coloca en las posiciones especificadas, tal como vemos en el programa BASIC que hemos transcrito.

```
1 GOTO 200
2 POKE 53265,PEEK(53265)AND223
3 POKE 53272,PEEK(53272)AND240OR4
4 STOP
200 REM ****DEMO ALT. RES. C64****
210 :
220 POKE 56,32:CLR: REM BAJAR TOPE MEMORIA
250 GOSUB 3000: REM INICIALIZAR S/R
350 :
360 REM ****ACTIVAR MOD. ALT. RES.****
370 :
380 PRINT CC$:PRINT:PRINT
390 INPUT"COLOR PRIMER PLANO";FG
400 INPUT"COLOR FONDO";BG
410 TT = FG*16+BG: REM CALC. COLOR
420 POKE COLOUR,TT: REM COL. S/R LENG. MÁQU.
430 POKE HRSFLG,1: REM ACTIVA ALT. RES.
440 POKE CLMFLG,1: REM ACTIVA PANT. ALT. RES.
450 SYS BEGIN: REM ENTRA S/R DE LENG. MÁQU.
460 :
500 REM ****DIBUJO DEL MODELO****
510 :
515 Z = 0:Y1 = 50:Y2 = 150:X1 = 160:SP = 6
520 FORY = Y1 TO Y2 STEP SP
530 FORX = Y2−Z TO Y2+Z STEP SP
540 GOSUB 1000: REM DIBUJAR PUNTO
550 NEXT X
555 Z = Z+SP
557 NEXT Y
560 :
565 GETJ$:IFJ$<>""THEN 565
570 GETA$:IF A$ = ""THEN 570:REM ESPERA OPRESION TECLA
580 :
600 REM ****LIMPIA PANT. ALT. RES.****
605 POKE HRSFLG,1:POKE CLMFLG,1
610 SYS BEGIN
620 :
630 REM ****DEMO DE LINEA****
640 X1 = 0:X2 = 300:Y1 = 0:Y2 = 190:SP = 1
670 GOSUB1500: REM TRAZADO LINEA
690 :
695 GETJ$:IFJ$<>""THEN695
700 GETA$:IF A$ = ""THEN700:REM ESPERA OPRESION TECLA
710 :
720 REM ****RESTABLECER PANTALLA****
730 :
740 POKE HRSFLG,0: REM DESACTIVAR ALT. RES.
750 SYS BEGIN
760 PRINT CC$:PRINT:PRINT
770 PRINTTAB(9)"*****FIN DE PROGRAMA*****"
780 END
999 :
1000 REM *** S/R TRAZADO EN A. RES. ***
1010 :
1020 XHI = INT(X/HX):XLO = X−XHI*HX
1030 POKE XBYTE,XLO:POKE XPAGE,XHI:POKE YBYTE,Y
1035 SYS PLOT: REM ENTRADA S/R TRAZADO
1040 RETURN
1500 REM *** S/R TRAZADO LINEA ***
1550 C9 = (Y2−Y1)/(X2−X1):C8 = C9*X1−Y1
1600 FORX = X1 TO X2 STEP SP
1650 Y = X*C9−C8
1700 GOSUB 1000: REM DIBUJAR PUNTO
1750 NEXT X
1800 RETURN
3000 REM *** S/R INIC. ***
3020 CC$ = CHR$(147): REM LIMPIA PANTALLA
3025 HX = 256
3030 HRSFLG = 49408: REM $C100
3040 CLMFLG = 49409: REM $C101
3050 COLOUR = 49410: REM $C102
3060 XBYTE = 49411: REM $C103
3070 XPAGE = 49412: REM $C104
3080 YBYTE = 49413: REM $C105
3085 BEGIN = 49422: REM $C10E
3090 PLOT = 49539: REM $C183
3095 PRINT CC$:PRINT:PRINT
3100 PRINTTAB(9)"*****CARGA LENG. MAQ.*****"
3150 PRINTTAB(9)"1) LENG. MAQU. EN CINTA"
3200 PRINTTAB(9)"2) LENG. MAQU. EN DATA"
3250 PRINTTAB(9)"3) LENG. MAQU. EN MEM."
3300 PRINT "DIGITE NUMERO ESCOGIDO"
3350 FOR LP = 0 TO 1 STEP 0
3400 GET OP$
3450 IF OP$>"0" AND OP$<"4" THEN LP = 1
3500 NEXT LP
3600 ON VAL(OP$) GOSUB 4000,5000,6000
3900 RETURN
4000 REM** S/R CARGA LENG. MAQU. DESDE CINTA**
4100 PRINT "S/R INSERCION CINTA CON LENG. MAQU."
4200 IF A = 0 THEN A = 1:LOAD "PLOTSUB.HEX",1,1
4900 RETURN
```

```
5000 REM** S/R CARGA LENG. MAQU. DESDE DATA**
5005 PRINTTAB(11)"******CARGANDOSE******"
5010 FOR I = HRSFLG TO HRSFLG+312:READ A
5020 POKE I,A:S = S+A:NEXT I
5030 READ CC:IF CC<>S THEN PRINT"CHECKSUM ERROR":END
5040 DATA 2,0,255,255,2,2,255,255,2,18
5050 DATA 255,255,2,2,72,138,72,152,72
5060 DATA 173,0,193,240,83,169,0,133,251
5070 DATA 169,4,133,252,162,3,160,0,173
5080 DATA 2,193,145,251,136,208,251,230
5090 DATA 252,202,48,8,208,244,145,251
5100 DATA 160,231,208,238,173,1,193,240
5110 DATA 24,169,0,133,251,169,32,133
5120 DATA 252,162,32,160,0,169,0,145,251
5130 DATA 136,208,251,230,252,202,208
5140 DATA 246,173,24,208,41,240,9,8,141
5150 DATA 24,208,173,17,208,9,32,141,17
5160 DATA 208,76,125,193,173,24,208,41
5170 DATA 240,9,4,141,24,208,173,17,208
5180 DATA 41,223,141,17,208,104,168,104
5190 DATA 170,104,96,72,138,72,152,72
5200 DATA 173,4,193,141,7,193,173,3,193
5210 DATA 41,248,141,6,193,173,3,193,41
5220 DATA 7,141,8,193,173,5,193,41,7,141
5230 DATA 10,193,162,3,78,5,193,202,208
5240 DATA 250,173,5,193,141,9,193,169,0
5250 DATA 141,11,193,141,12,193,162,5
5260 DATA 173,11,193,24,109,9,193,141,11
5270 DATA 193,202,208,243,162,6,14,12
5280 DATA 193,14,11,193,144,3,238,12,193
5290 DATA 202,208,242,173,11,193,24,109
5300 DATA 6,193,141,11,193,173,12,193
5310 DATA 109,7,193,141,12,193,173,11
5320 DATA 193,24,105,0,141,11,193,173,12
5330 DATA 193,105,32,141,12,193,173,11
5340 DATA 193,24,109,10,193,141,11,193
5350 DATA 173,12,193,105,0,141,12,193
5360 DATA 173,11,193,133,251,173,12,193
5370 DATA 133,252,169,1,141,13,193,56
5380 DATA 169,7,237,8,193,170,14,13,193
5390 DATA 202,208,250,160,0,177,251,13
5400 DATA 13,193,145,251,76,125,193
5410 DATA 38698:REM*CHECKSUM*
5900 RETURN
6000 REM** S/R LENG. MAQU. EN MEM.**
6100 RETURN
```

Uso de PLOTSUB

Este programa BASIC de demostración ilustra las diversas fases necesarias para el uso de las rutinas para alta resolución en lenguaje máquina:

1) Si usted dispone de un ensamblador, puede digitar el programa en assembly, ensamblarlo, guardarlo como archivo fuente y a continuación guardar el código objeto contenido en $C100 hasta $C238 con el nombre de "PLOTSUB.HEX". No intente ejecutar la subrutina en esta fase, pues puede que la pantalla de alta resolución machaque el propio ensamblador, colgando la subrutina.

2) Para usar en un programa estas rutinas para alta resolución se debe bajar el tope de la memoria MEMTOP (véase línea 220 del programa) y cargar el código desde la cinta (subrutina de línea 4000).

3) Se puede guardar la subrutina de línea 5000, si se quiere, como un programa BASIC (denominado, p. ej., "MCODELOAD"). A la hora de usarlo baje el MEMTOP, cargue después y ejecute este MCODELOAD (lo que hará es cargar el lenguaje máquina en la memoria). Digite NEW y cargue a continuación el programa que desea ejecutar: las rutinas para alta resolución están ahora dentro de la memoria y pueden ser accedidas mediante las instrucciones SYS adecuadas.

4) El último grupo de datos de la subrutina 5000 es una suma de control: la suma que totaliza los datos anteriores. Si el programa se para con un "CHECKSUM ERROR", quiere decir que usted introdujo mal los datos y debe corregir el error.

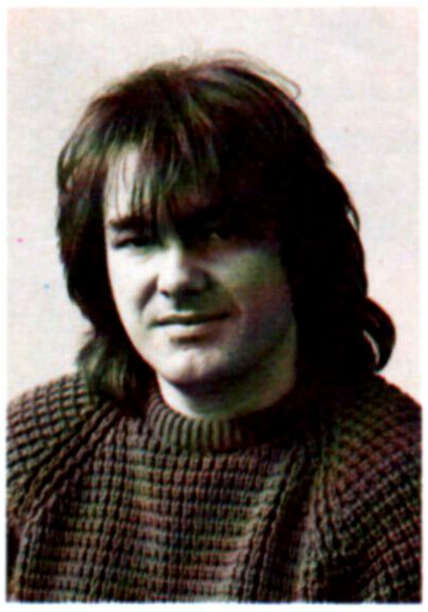
```
;**********************************************
;**********************************************
;**                                          **
;**          RUTINA DE TRAZADO               **
;**                                          **
;**          EN ALTA RESOLUCION              **
;**                                          **
;**          PARA UN COMMODORE 64            **
;**                                          **
;**********************************************
;**********************************************

PTR     EQU  $FB
MPBLO   EQU  $00
MPBHI   EQU  $20
SCBLO   EQU  $00
SCBHI   EQU  $04
SCBLK   EQU  $03
SCREM   EQU  $E7
MPBLK   EQU  $20
        ORG  $C100
;
HRSFLG  DB   $00
CLRFLG  DB   $00
COLOUR  DB   $00
XLO     DB   $00
XHI     DB   $00
YLO     DB   $00
HBLO    DB   $00
HBHI    DB   $00
REMX    DB   $00
VBYTE   DB   $00
REMY    DB   $00
ROWLO   DB   $00
ROWHI   DB   $00
BPOS    DB   $00
;****PASO A LA MOD. A. RES.*****
;****Y LIMPIEZA AREA MAPA BITS***
        PHA
        TXA
        PHA
        TYA
        PHA
        LDA  HRSFLG
        BEQ  RESET
;****AREA COLORES PANTALLA*******
        LDA  #SCBLO
        STA  PTR
        LDA  #SCBHI
        STA  PTR+1
        LDX  #SCBLK
        LDY  #$00
        LDA  COLOUR
AGAIN STA (PTR),Y
        DEY
        BNE  AGAIN
        INC  PTR+1
        DEX
        BMI  CLTEST
        BNE  AGAIN
        STA  (PTR),Y
        LDY  #SCREM
        BNE  AGAIN
CLTEST  LDA  CLRFLG
        BEQ  HRESON
****LIMPIEZA AREA MAPA BITS*******
        LDA  #MPBLO
        STA  PTR
        LDA  #MPBHI
        STA  PTR+1
        LDX  #MPBLK
        LDY  #$00
        LDA  #$00
NEXT    STA  (PTR),Y
        DEY
        BNE  NEXT
        INC  PTR+1
        DEX
        BNE  NEXT
;****PASO A LA MOD. MAPA BITS*********
HRESON  LDA  $D018
        AND  #$F0
        ORA  #$08
        STA  $D018
        LDA  $D011
        ORA  #$20
        STA  $D011
        JMP  EXIT
;****VUELTA A PANTALLA NORMAL***
RESET   LDA  $D018
        AND  #$F0
        ORA  #$04
        STA  $D018
        LDA  $D011
        AND  #$DF
        STA  $D011
;****SALIDA DE LENG. MAQ. ******************
EXIT    PLA
        TAY
        PLA
        TAX
        PLA
        RTS
;****CALCULO TRAZO EN A. RES***
        PHA
        TXA
        PHA
        TYA
        PHA
;****CALCULO BYTE HORIZ.*****
        LDA  XHI
        STA  HBHI
        LDA  XLO
        AND  #$F8
        STA  HBLO
        LDA  HLO
        AND  #$07
        STA  REMX
;****CALCULO BYTE VERT.*****
        LDA  YLO
        AND  #$07
        STA  REMY
        LDX  #$03
SHIFT   LSR  YLO
        DEX
        BNE  SHIFT
        LDA  YLO
        STA  VBYTE
;****CALCULO FILA (ROW)************
        LDA  #$00
        STA  ROWLO
        STA  ROWHI
        LDX  #$05
        LDA  ROWLO
        CLC
        ADC  VBYTE
        STA  ROWLO
        DEX
        BNE  FIVE
        LDX  #$06
        ASL  ROWHI
        ASL  ROWLO
        BCC  NCARRY
        INC  ROWHI
        DEX
        BNE  MULT
;****SUMA BYTE HORIZ.******
        LDA  ROWLO
        CLC
        ADC  HBLO
        STA  ROWLO
        LDA  ROWHI
        ADC  HBHI
        STA  ROWHI
;****SUMA BASE MAPA A. RES.*******
        LDA  ROWLO
        CLC
        ADC  #MPBLO
        STA  ROWLO
        LDA  ROWHI
        ADC  #MPBHI
        STA  ROWHI
;****SUMA RESTO DE Y LO*****
        LDA  ROWLO
        CLC
        ADC  REMY
        STA  ROWLO
        LDA  ROWHI
        ADC  #$00
        STA  ROWHI
;****COMBINACION DE 2 BYTES*******
;****DE DIRECC. EN PAG. CERO*****
        LDA  ROWLO
        STA  PTR
        LDA  ROWHI
        STA  PTR+1
;****CALCULO POSIC. PIXEL****
        LDA  #$01
        STA  BPOS
        SEC
        LDA  #$07
        SBC  REMX
        TAX
        ASL  BPOS
        DEX
        BNE  POWER
;****ENCENDIDO DEL PIXEL************
        LDY  #$00
        LDA  (PTR),Y
        ORA  BPOS
        STA  (PTR),Y
        JMP  EXIT
```

Para todo el mundo

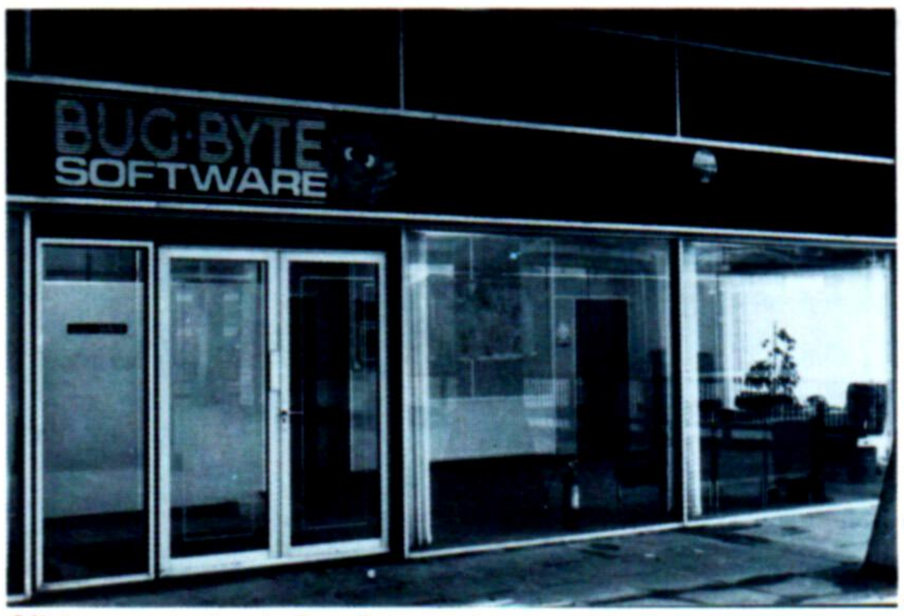
Bug-Byte es una empresa de software que ha crecido sostenida y paralelamente a la industria británica de ordenadores personales

Bug-Byte se fundó en la primavera de 1980, cuando Tony Baden y Tony Milner, dos estudiantes de química del University College, de Oxford, empezaron a escribir programas para el ZX80 que se acababan de comprar. Comprendiendo que era muy poco el software existente para la máquina, decidieron comercializar sus juegos, compraron 40 cassettes vírgenes e hicieron publicidad en una revista de ordenadores anunciando un paquete de cinco juegos. Los pedidos empezaron a llegar en un promedio de 15 por semana y los socios reinvirtieron sus beneficios en más publicidad y escribieron otros programas para el ZX80. Luego, durante ese mismo año se lanzó al mercado el Acorn Atom, y Bug-Byte amplió su ámbito para satisfacer la demanda de software para la nueva máquina



Tony Milner

Tony Baden



Roger Sanders

Oficinas centrales de Bug-Byte, en Liverpool

A principios de 1981 apareció el ZX81 y la demanda de juegos para el ZX80 disminuyó enseguida, de modo que Baden y Milner se dedicaron a escribir software para el ZX81. Obtuvieron su licenciatura en Oxford en junio de 1981 y entonces Bug-Byte se trasladó a Liverpool, la ciudad natal de Tony Baden. A partir de entonces la empresa comenzó a trabajar con dedicación exclusiva y al cabo de poco tiempo duplicaron las ventas de programas.

En la Navidad de 1981 la competencia en el mercado de software de juegos se habían hecho más dura. Para mantener las ventas Bug-Byte contrató una agencia de publicidad para que se encargara de la comercialización y empezó a cuidar el anuncio de sus cassettes así como su presentación, hecha a todo color, utilizando al mismo tiempo una empresa de reproducción de cintas profesional para asegurarse cassettes de alta calidad.

Este nuevo enfoque a la presentación de sus productos, junto con la introducción de una red de distribuidores a nivel nacional, tuvo un marcado efecto sobre las ventas y la empresa contrató más personal. A medida que hacían su aparición más ordenadores personales, Bug-Byte contrataba programadores por cuenta propia para satisfacer la creciente demanda. Muchos de estos programadores abandonaron luego la colaboración con Bug-Byte para formar sus propias firmas de software, como Quicksilva y Software Projects. La empresa les paga a sus programadores un porcentaje fijo por cassette y afirma que un autor cuyo juego consiga situarse entre los veinte mejor vendidos puede ganar entre 10 000 y 40 000 libras en el primer año.

A finales de 1982 la red de ventas de Bug-Byte estaba compuesta por más de 200 distribuidores independientes, además de las más importantes cadenas de tiendas, concluyendo la época en que la empresa hacía sus ventas por correspondencia. El aprovisionamiento de cintas estaba resultando un gran dolor de cabeza, ya que las empresas de reproducción eran incapaces de satisfacer la demanda de las firmas de software que estaban preparando sus existencias para el *boom* de ventas de la Navidad, por lo que Bug-Byte creó su propia empresa de reproducción, Spool. En junio de 1983 la empresa se trasladó a unas instalaciones más grandes, en Canning Place (Liverpool); éstas se diseñaron siguiendo las especificaciones de Bug-Byte.

Entre los mayores éxitos de Bug-Byte figuran el juego de aventuras con gráficos *Twin kingdom valley* (para el BBC y el Commodore 64) y *Manic miner*, que funciona con el ZX Spectrum y el Commodore 64. El autor de *Manic miner*, Matthew Smith, recientemente ha dejado la empresa para formar Software Projects, llevándose consigo los derechos de autor *(copyright)* del juego.

La expansión de Bug-Byte ha sido ininterrumpida y el software de la empresa en la actualidad se vende en la mayoría de los países de Europa occidental, así como en Australia, Nueva Zelanda y República de Sudáfrica. Un reciente acuerdo con la CBS británica, representante comercial de Bug-Byte en Europa, podría significar la penetración de la empresa en el lucrativo mercado norteamericano. John Phillips, director de marketing de Bug-Byte, ha manifestado a este respecto que "utilizando el contacto con CBS como modelo, esperamos conseguir que la operación tenga una proyección auténticamente mundial".


mi COMPUTER
Editorial Delta, S.A.

9 788485 822836 10041