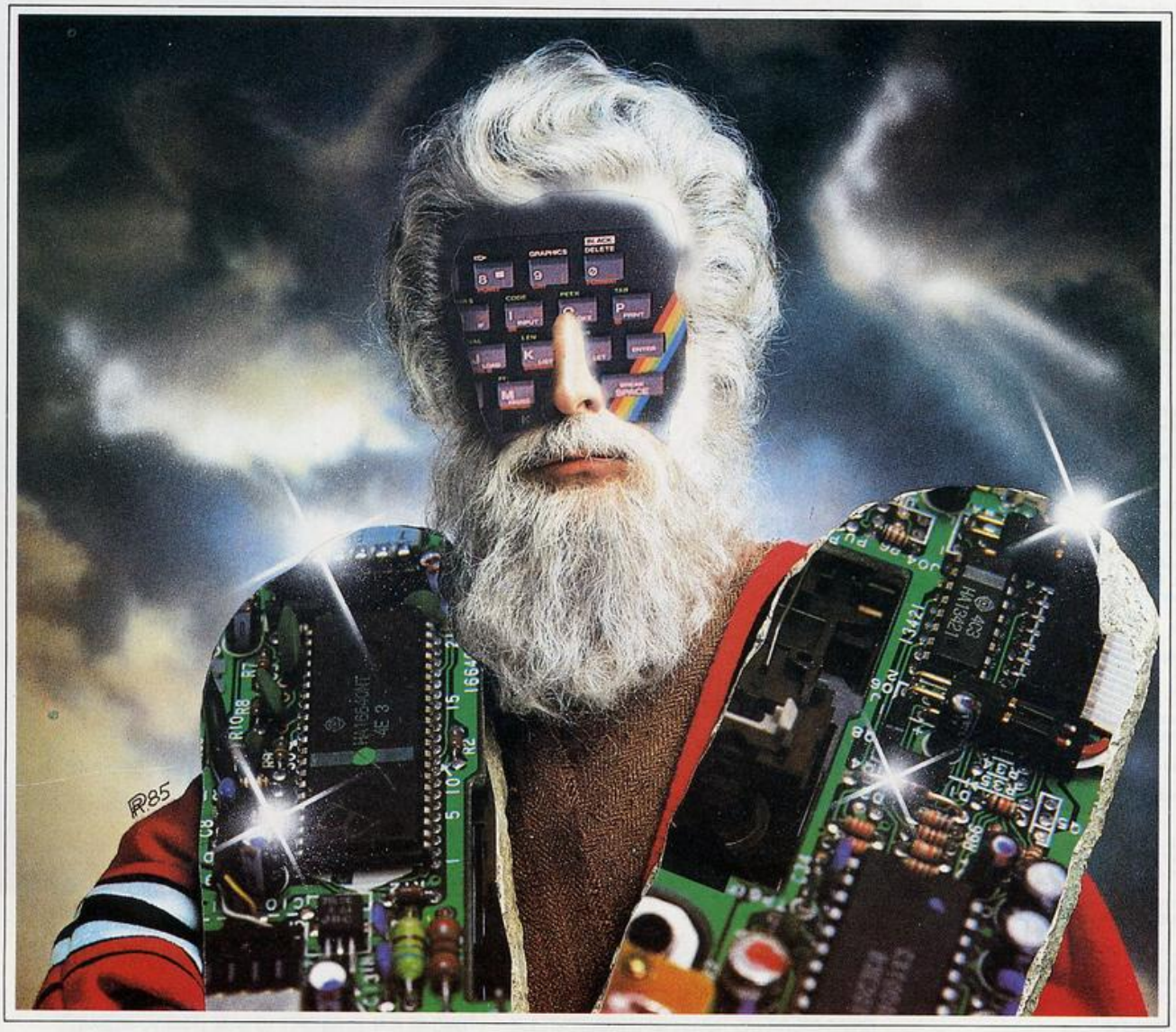
42
185 pts.
IVA INCLUIDO

RUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

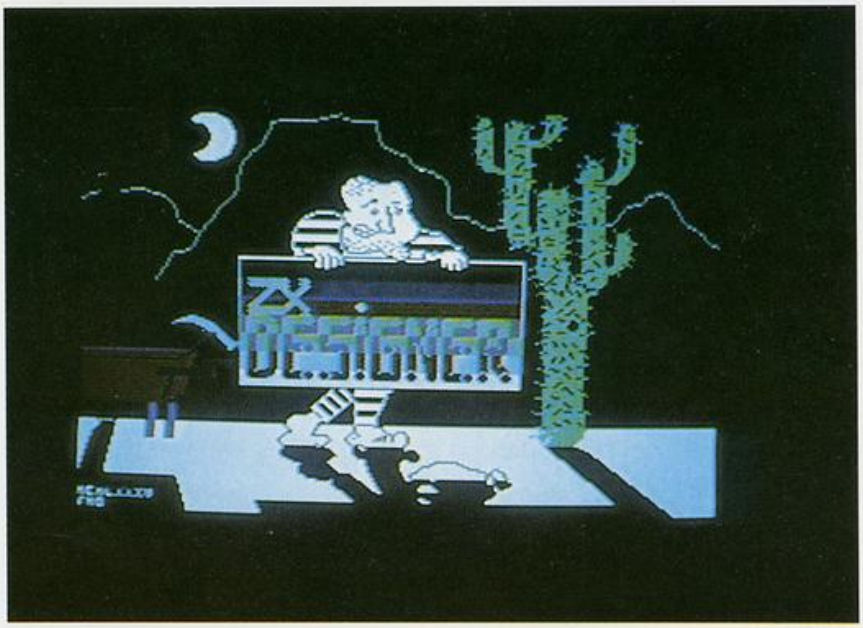
DISEÑADORES DE PANTALLAS

COMO ya hemos comentado anteriormente, incluso dedicándole un capítulo entero al tema, la estética es uno de los puntos fuertes en el éxito de cualquier programa. A nadie se le oculta que ésta se concreta principalmente en la pantalla, debido a lo cual, gran parte de los esfuerzos de programación se consumen en crear pantallas atractivas. Como quiera que ésto no es en muchas ocasiones tarea fácil, los programadores han creado herramientas que les hagan algo más sencillo su trabajo: los diseñadores de pantallas.
Entre los diseñadores de pantallas existen lógicamente programas más o menos depurados, que se ajustarán a nuestro objetivo según lo complicado que sea éste. En general, un diseñador de pantallas es un programa que nos permite dirigir cómodamente por una pantalla limpia un punto que va dejando un rastro, a modo de lápiz.
Inevitablemente, y dado que la simple utilidad de ésto es bien poca, se le van añadiendo otras características encaminadas a mejorar la calidad estética del resultado. Así, el lápiz es capaz de pintar en cualquiera de los colores del Spectrum, borrar, invertir, o pasar de un punto a otro sin afectar a lo ya escrito.

ZX DESIGNER (ABC SOFT)

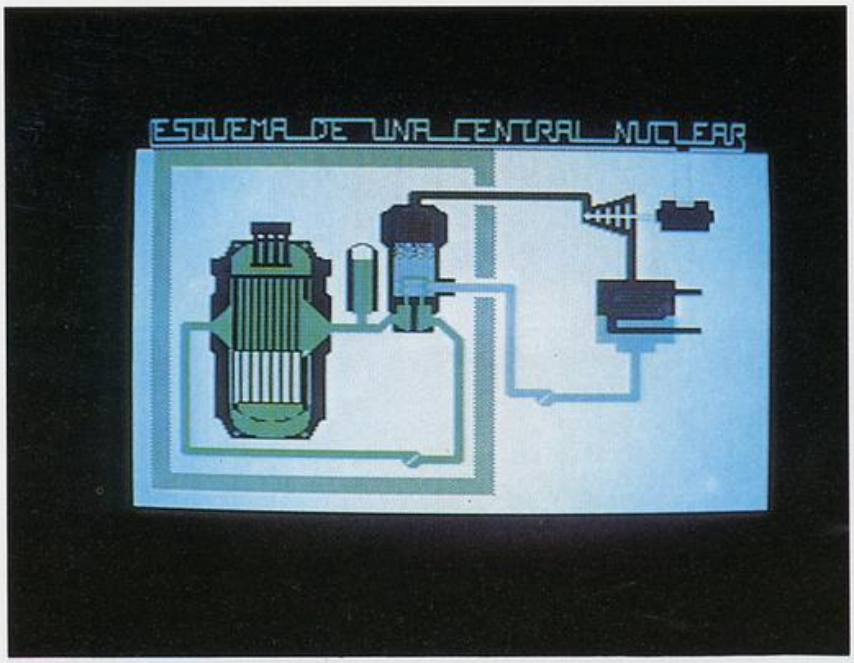
Las características de estos programas se van complicando, aportando cada vez más herramientas para facilitar el trabajo de diseño. Por ejemplo, el programa ZX DESIGNER, además de las cualidades comunes mencionadas, incorpora las siguientes posibilidades:
— Fijar un punto que se utilizará como base en las opciones de trazado de figuras que a continuación veremos.
— Trazar un arco, para lo cual bastará con pulsar la tecla **A** e introducir la longitud del arco en radianes.
— Trazar un paralelogramo.
— Trazar una línea hasta la «punta» de nuestro lápiz.
— Trazar un círculo, estableciendo como centro el cursor.
— Copiar la pantalla en la impresora.
— Cambiar los atributos con los que se dibuja en la pantalla.
— Rellenar de color cualquier figura.
— Guardar la configuración de la pantalla actual en la memoria.
— Sombrear una figura, mediante el relleno con un trazo discontínuo.
— Variar el paso con que el «lápiz» se desliza por la pantalla, para así poder alterar su velocidad de desplazamiento.
— Cuadricular la pantalla, que será una función muy útil para determinar los atributos de cada posición de carácter.
— Borrar la pantalla.
— Grabar o cargar la pantalla en cinta.
— Hacer más grueso el trazo del «lápiz».
— Definir caracteres gráficos (UDG).
— Cambiar de banco de caracteres definidos (dispone de ocho bancos).
— Grabar o cargar UDGs.
— Escribir o cargar UDGs.

Aunque la pantalla de presentación del ZX Designer no es especialmente espectacular, con este programa podremos conseguir buenos resultados.



— Escribir textos en la pantalla.
— Borrar la última operación efectuada, función de extremada utilidad para los que somos demasiado ligeros de dedos.
— Ampliar una zona de pantalla.
— No alterar los atributos al paso del cursor.
Como podemos comprobar es muy amplia la gama de efectos que se puede obtener con este programa. Pero existen aún ciertas exigencias que quedan sin cubrir, como por ejemplo la confección de figuras tridimensionales.

El ARTIST cuenta entre sus características más destacables con un generador de gráficos definibles (UDG).



También la realización de gráficos técnicos puede ser contemplada por un diseñador de pantallas.

VU-3D (INVESTRONICA)

VU-3D es un sofisticado programa para dibujo tridimensional. Utilizando comandos sencillos podrá crear un objeto sólido, o conjunto de ellos, en el espacio tridimensional. Con este programa se puede obtener una imagen tridimensional del mundo haciendo rotar los objetos mirándolos desde diferentes ángulos.
Este mundo tridimensional se trata en conjunto con sus estructuras de datos, del mismo modo que una cámara programable, en el mismo sentido en que el ojo observa el mundo tridimensional. Podemos rodear un objeto y mirarlo desde lejos o de cerca desde diferentes direcciones. En este sentido, podemos diseñar un objeto sólido y obtener una sensación de su realidad tridimensional.

Para la confección de pantallas de este género contamos con la colaboración de los DISEÑADORES DE PANTALLAS.

La función de creación permite, mediante un sencillo juego de comandos, construir un objeto en tres dimensiones. Una vez construido éste, existe la posibilidad de acercarlo o alejarlo en un efecto de *zoom* o girarlo en cualquier dirección y sentido; estas operaciones posteriores a la creación son efectuadas en una segunda fase denominada *display* (pantalla).

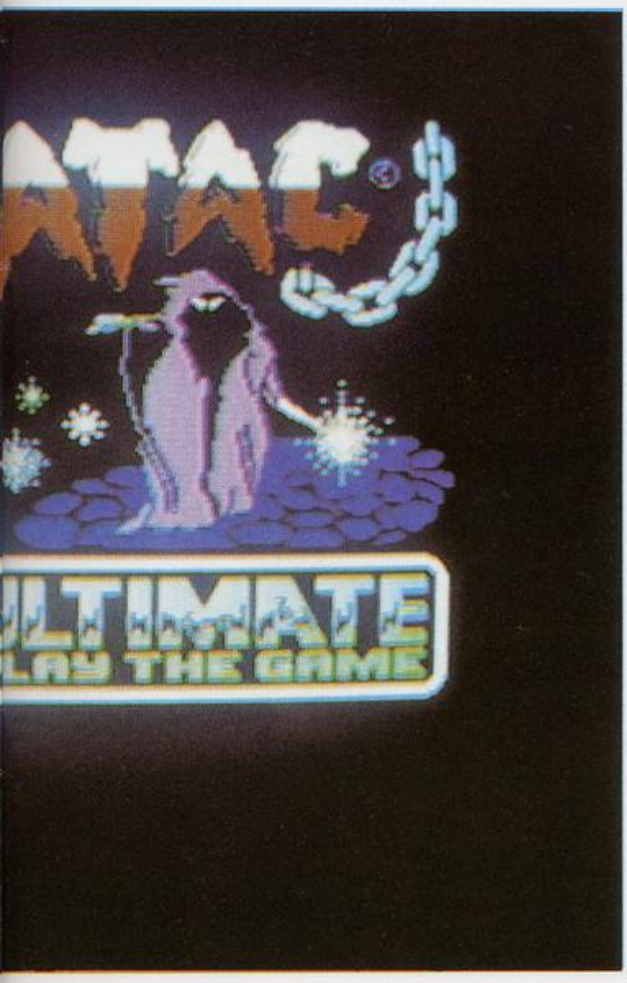
En una tercera etapa, habiéndose creado un objeto en la fase de creación y fijada su ampliación y punto de observación en la de *display*, podemos obtener, en un color de nuestra elección, el objeto en sus otras dos formas de presentación: *hidden line* (superficie frontal) o *shade* (sombreado). Esta tercera etapa es la denominada *picture* (imagen).

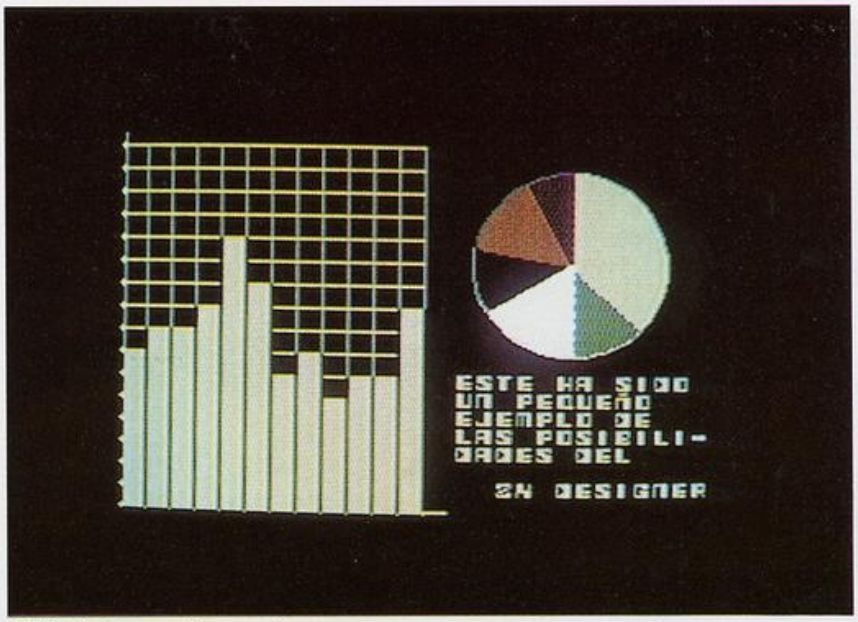
Las opciones de la fase *picture* son: *shade* (sombra), *hidden line* (superficie frontal), *print* (impresora), *colour* (color), *keep* (guardar), *quit* (salir). La función *colour* permite elegir el color apropiado en cada caso a la imagen representada, *print* realiza una copia en papel y *quit* nos devuelve a la fase *display*.

keep es una opción muy a tener en cuenta, puesto que nos permite grabar la figura como una pantalla, la cual podrá ser cargada utilizando **SCREEN$** sin la presencia del programa VU-3D.

El comando *hidden line* produce una imagen frontal de los objetos existentes en pantalla. Los extremos que limitan las caras superiores del objeto, que son aquellas que el observador puede ver, son las únicas líneas dibujadas, mientras que los de la cara posterior no se dibujan. La ejecución de este comando puede tardar un tiempo considerable si se trata de una figura complicada con muchas caras.

Otra forma de ver el objeto u objetos de forma tri-

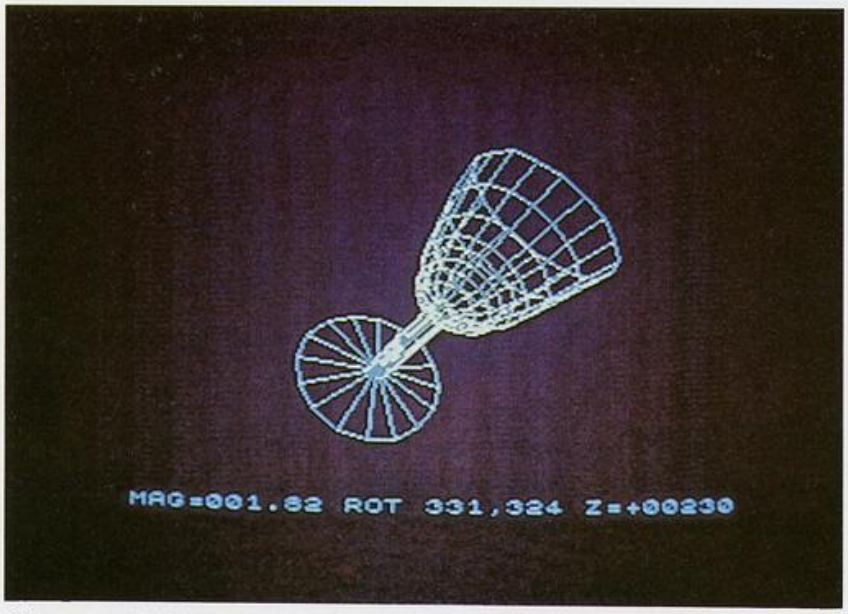




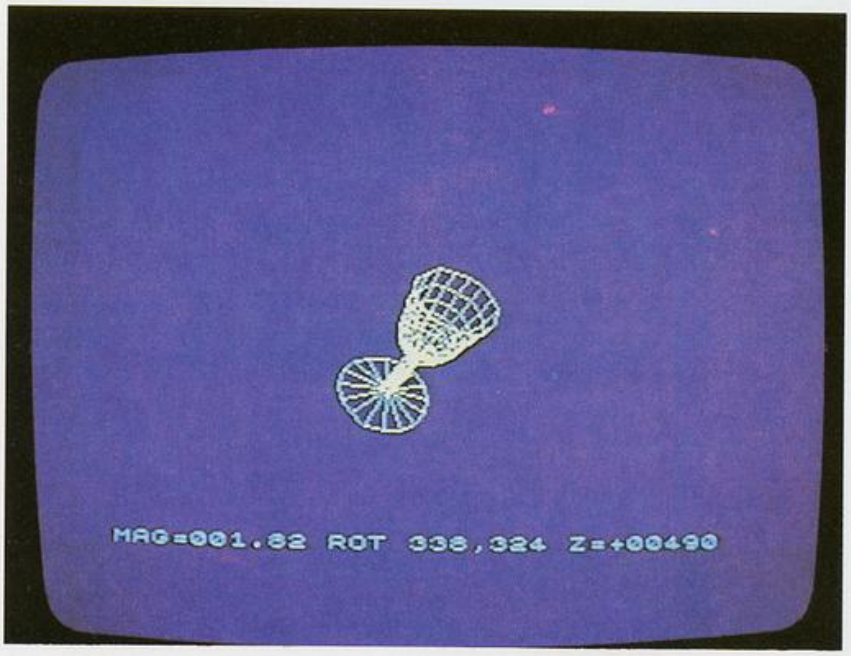
El diseño de pantallas no sólo se emplea en el ámbito de los juegos, sino también en el comercial.

La cara B de la cinta de ZX Designer se encabeza con esta pantalla de presentación, seguida del programa en sí. La cara A está destinada a demostraciones.

dimensional es sombrear aquellas caras que el observador puede ver. La claridad del sombreado de cada cara viene determinada por una combinación de luz de fondo y reflexión difusas desde una única fuente luminosa.
Así, utilizando el comando *shade* se nos interrogará sobre dónde deseamos situar la fuente de luz: arriba, centro o abajo, y además, izquierda, centro o derecha.
Además de las etapas mencionadas el programa nos permite modificar la figura diseñada o grabar, y por supuesto a cargar, las características de la figura definida, no en forma de pantalla sino de fichero de datos. Finalmente, la opción *abandon* (abandonar) nos permite borrar el fichero de datos existente, dejando la memoria libre para una carga desde casete, o crear una nueva figura en la primera etapa.



El programa VU-3D nos permite obtener una imagen tridimensional, haciendo rotar los objetos y mirando desde diferentes ángulos.



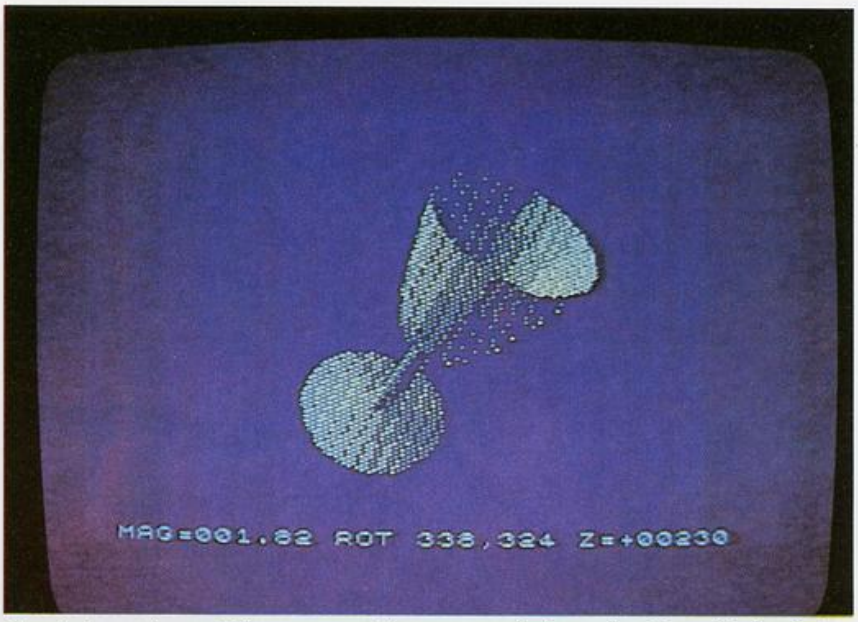
Al igual que la ampliación, VU-3D permite también la reducción de la imagen dibujada.

ARTIST (DINAMIC)

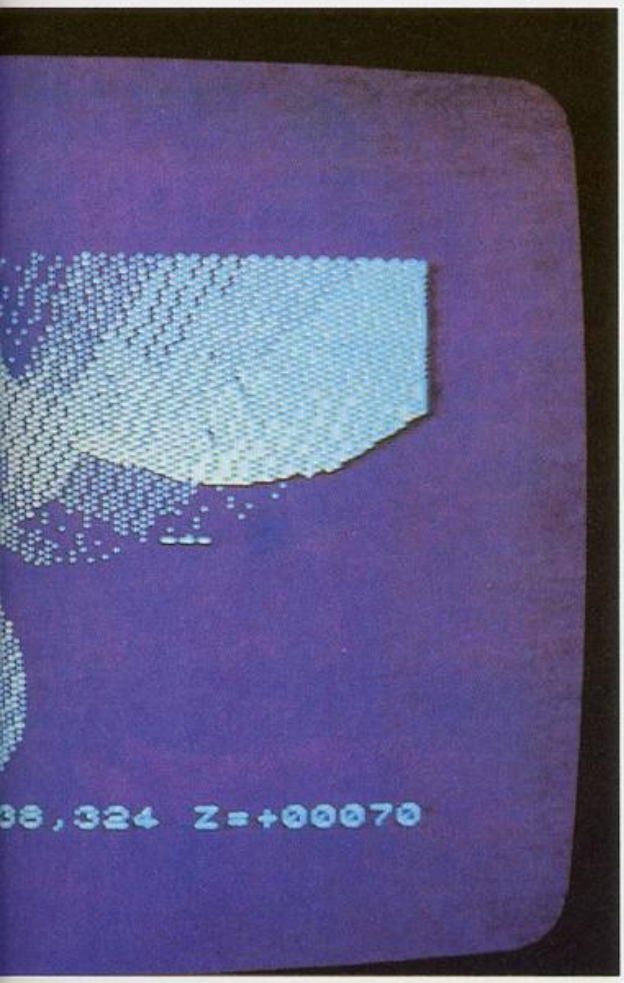
ARTIST es, sin duda, uno de los más completos equipos gráficos para el diseño de pantallas presentes en el mercado nacional. Su acción se concreta en tres puntos fundamentales: creación de figuras en tres dimensiones, definición de caracteres UDGs y diseño sobre la pantalla de alta resolución.

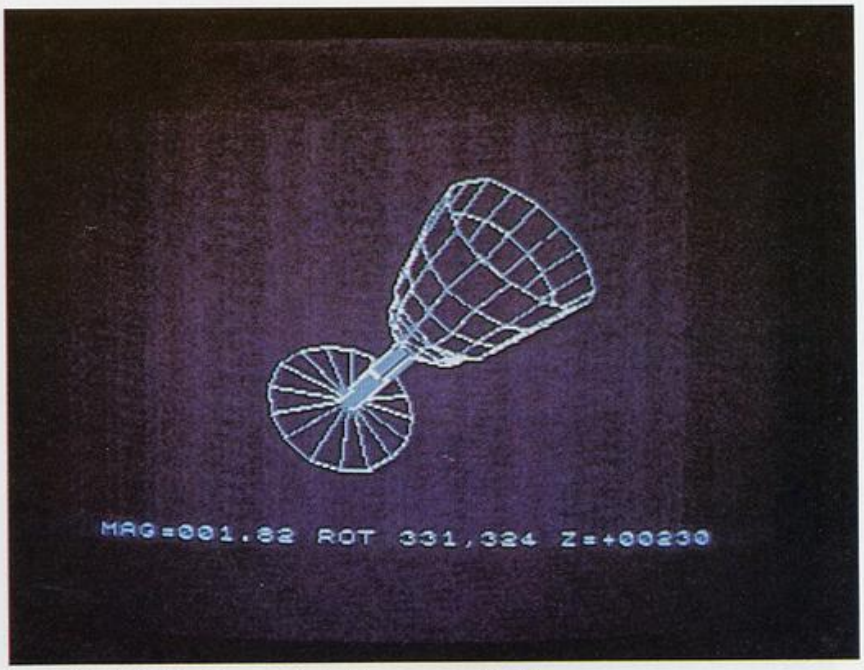
Las diversas opciones del programa son combinables: en el ejemplo se ha empleado el sombreado y la ampliación.

En el denominado modo *plotter* se nos presenta una pantalla libre de 176 por 256 *pixels* en la cual podemos desplazar un cursor dejando un rastro a su paso, permitiendo deslizar dibujos sobre la pantalla. Este rastro vendrá dado en el color que nosotros indiquemos, pudiendo ser alterado éste en cualquier momento mediante la pulsación de las teclas numéricas del 0 al 7.
De modo similar podemos alterar el color de fondo y también de dos características tan importantes como el **FLASH** y el **BRIGHT**. Asimismo, también es posible definir el tamaño del cursor (pequeño o grande), o la velocidad de desplazamiento del mismo por la pantalla (tres velocidades). Naturalmente, también podemos desplazar el cursor borrando en vez de escribiendo, o simplemente dejando inalterado el contenido de la pantalla.
El programa también prevé la posibilidad de trazar líneas rectas, arcos circulares, e incluso círculos, representando sin duda una gran ayuda a la hora de confeccionar este tipo de figuras de cierta dificultad sin pretender ser realizadas a mano alzada.
Asimismo, la posibilidad de hacer *fill* (rellenar una superficie cerrada de un color determinado), acompañada del establecimiento de un *grid,* conforman las características más destacables de este modo. El *grid* es una cuadrícula de colores que no afectan al dibujo pero nos ayudan, aprovechando la posibilidad de realizar *scrolls* de la pantalla, a realizar una correcta utilización del color en cada posición de carácter.
En este modo el programa ARTIST es muy similar a MELBOURN DRAW que más adelante pasaremos a estudiar.
Sobre esta misma pantalla también es posible es-



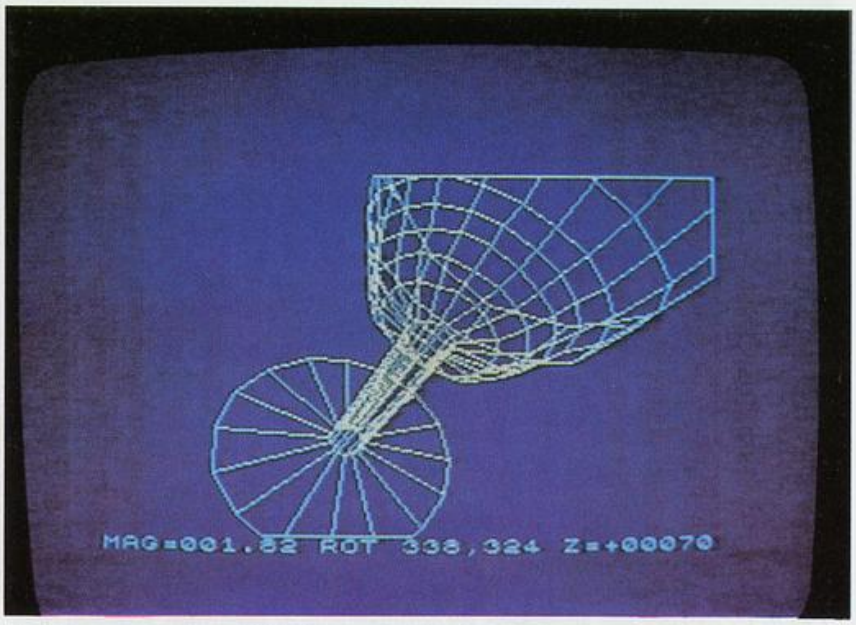
Una de las opciones más interesantes del programa es la de sombreado, pudiendo elegir incluso la situación del foco de luz.





La opción de hidden line no efectua la representación de las lineas ocultas del objeto.

cribir texto, incluso en caracteres gigantes, introduciendo el mensaje y el tamaño deseado para las letras.
En cuanto a la creación de gráficos definidos, disponemos de un clásico generador de UDGs muy similar a otros de este tipo, como el que acompaña la cinta de demostración de nuestro aparato. Así por ejemplo nos permite:
— Definir el gráfico en el carácter que queramos del banco seleccionado.
— Invertir el gráfico.
— Girar el gráfico a la derecha.
— Reflejar el gráfico.
— Borrar el grafico.
El Artist también incorpora un tratamiento de imágenes tridimensionales, que podemos fundir una vez confeccionada con la pantalla principal. Para su diseño empleamos puntos definidos por tres coordenadas (X, Y y Z), que podemos dejar libres, o unir mediante una línea con el punto anteriormente definido. De forma similar al VU-3D, es posible acercar y alejar la figura, así como girarla en cualquier dirección y sentido.

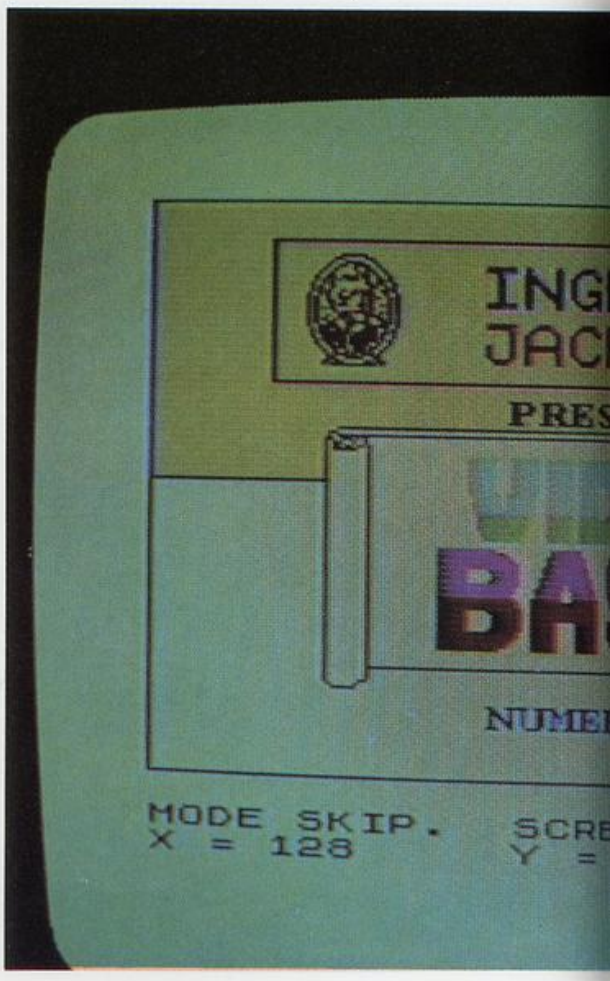


VU-3D contempla la posibilidad de ampliación de las imágenes.

En el primer nivel de ZOOM la pantalla se muestra multiplicada por cuatro.

MELBOURNE DRAW (MELBOURNE HOUSE)

Para finalizar vamos a hablar de un tipo de diseñador de tipo medio, que incorpora funciones si-



El MELBOURNE DRAW incorpora funciones muy similares al modo plotter del ARTIST.

milares al modo *plotter* del Artist, y tiene como característica fundamental su sencillez de manejo: el MELBOURNE DRAW.

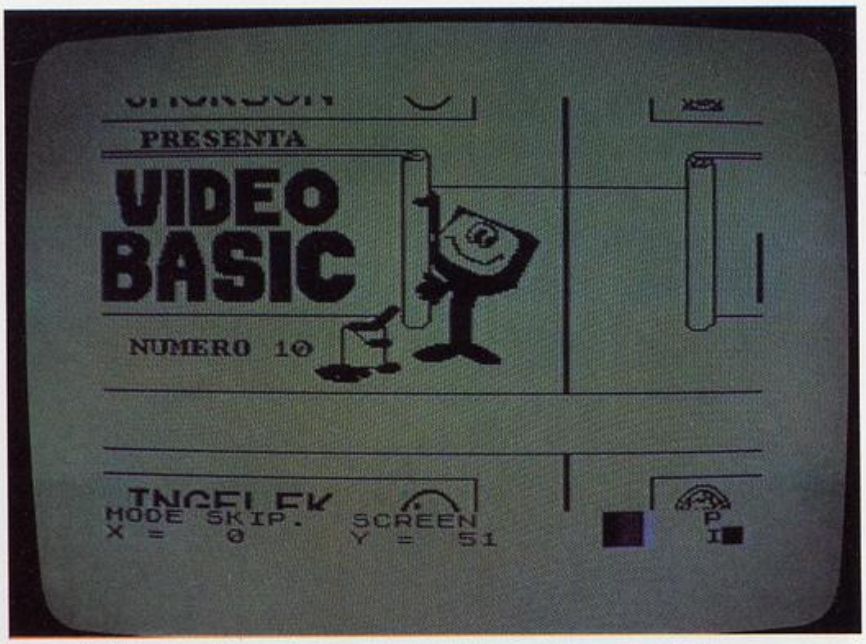
El programa se compone de un pequeño soporte BASIC, y un bloque de código máquina, que efectúa propiamente el trabajo de diseño gráfico. El BASIC tan solo contiene un menú de opciones que da acceso al modo de edición y se ocupa de gestionar la grabación, carga y verificación de las pantallas o juego de gráficos definidos creados.

Una copia de la pantalla se realiza sobre otra zona de memoria, de forma que podemos trabajar sin problema en las 24 líneas de la pantalla.

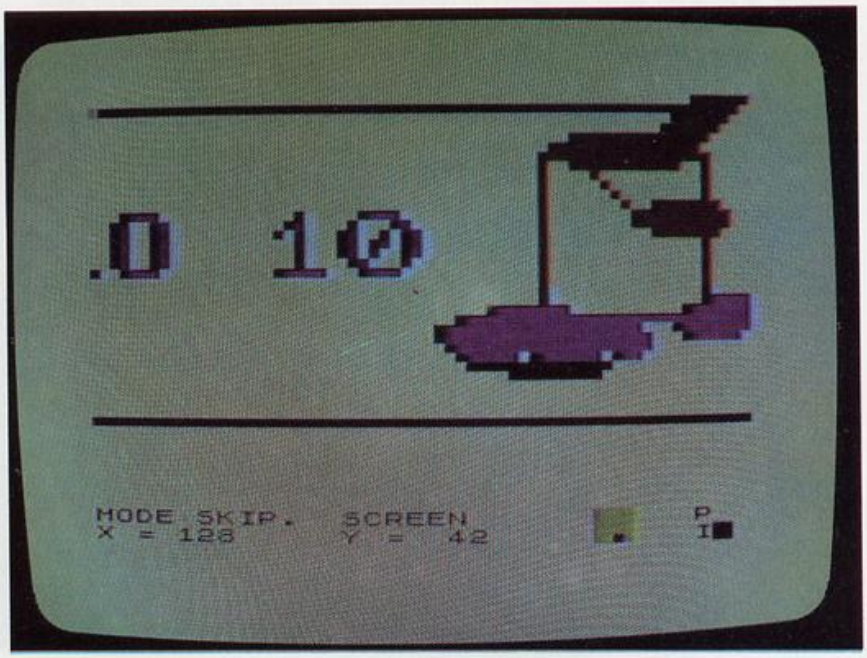
Al entrar en el modo edición, un cursor representado por un punto de *plot* intermitente indica nuestra posición.

Para el desplazamiento del punto, utilizaremos las teclas que circundan la **S** (Q, W, E, A, D, Z, X y C), pudiendo encontrarse el cursor en cuatro modos: SKIP, SET, RESET o INVERT. El primero de ellos se selecciona mediante la pulsación de **SPACE** y nos permite desplazarnos sin afectar a la pantalla; el segundo se activa mediante **ENTER** y hace que el punto vaya dejando rastro a su paso. El tercer modo (RESET) se utiliza para borrar y se activa con la tecla **O**, mientras que la tecla **I** sitúa el modo INVERT que cambia de estado la pantalla en los puntos por los que pasa el cursor, es decir, apaga los puntos encendidos, y enciende los apagados.

Aunque el desplazamiento del cursor es nuestra herramienta básica para el dibujo, disponemos también de otras funciones de gran interés. En primer lugar, la pantalla va siempre acompañada de una zona de indicadores de dos líneas, que indican el lugar de la pantalla en que se encuentra el cursor (coordenadas X, Y), el modo de dibujo



La posibilidad de scroll punto a punto y en cualquier sentido, es otra de las características más interesantes del MELBOURNE DRAW.

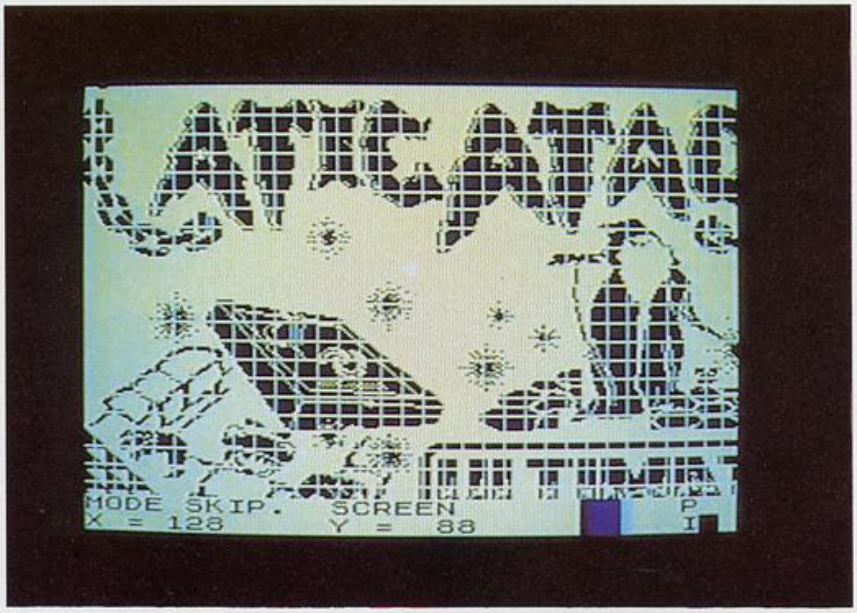


En el segundo y último nivel de ZOOM la pantalla aparece multiplicada por dieciséis.

(SKIP, SET, ETC...), los atributos empleados (fondo, tinta, brillo y parpadeo), y otras características que a continuación veremos. Esta zona, que habitualmente se sitúa en las dos últimas líneas de la pantalla (las reservadas al sistema), puede ser desplazada temporalmente a las dos primeras, pulsando **CAPS SHIFT + 9**, lo cual nos permite dibujar en las veinticuatro líneas de la pantalla.

El cambio de atributos se efectúa de una manera muy rápida; simplemente tenemos que pulsar la tecla numérica correspondiente al color deseado, sin **CAPS SHIFT** para la tinta y con **CAPS SHIFT** para el fondo, mientras que las teclas **CAPS SHIFT + V** afecta al modo **FLASH** y **CAPS SHIFT + B** a **BRIGHT**. Por otra parte, para hacernos una idea de conjunto, también podemos alterar el color del marco de pantalla, pulsando la tecla **B** y contestando a la pregunta **BORDER?** que aparecerá en la zona de indicadores con el código de color deseado.

Una de las propiedades del MELBOURNE DRAW es marcarnos, sin deteriorar la pantalla, la situación de las posiciones de carácter, para facilitar la asignación de color.



Además del aumento temporal (ZOOM), el MELBOURNE DRAW puede efectuar aumentos definitivos mediante la opción SCALE.

Para efectuar acciones sobre la pantalla completa, deberemos pulsar **CAPS SHIFT + R**, lo cual nos dará acceso a un menú en la zona de indicadores para situar todo el fondo de pantalla del color fijado para fondo (opción P del menú), toda la tinta del color de la tinta (opción I), asignar fondo y tinta a la vez (B), borrar la pantalla (S), borrar la pantalla y ajustar el color de fondo y tinta (A), o bien regresar a la pantalla principal (N).

La asignación de color no sólo se efectúa en el momento de trazar el dibujo, sino que también puede ser establecida independientemente pulsando **H**. El cursor pasará a tener las dimensiones de un carácter, funcionando en los modos SET o SKIP, aunque en esta ocasión, en el modo SET no producirá modificación de la pantalla sino de los colores, tal como son establecidos mediante las teclas numéricas.

Para ayudarnos en la tarea de colorear, disponemos de un grid para orientarnos claramente sobre las posiciones de carácter en las que se encuentran los elementos del dibujo (tecla **G**) y de un sistema de *scroll* que nos puede ayudar a cuadrar las figuras en los caracteres que nos interesen; para ello pulsaremos la tecla **K** y desplazaremos la pantalla utilizando los mismos controles que para el cursor (teclas de alrededor de la **S**). Finalmente, el **FILL** (rellenado de superficies) que se consigue con la pulsación de **CAPS SHIFT + F** es otra característica de interés.

En lo referente al texto, también puede ser incluido libremente en la pantalla, pasándose a este modo con la pulsación de **T**. En cuanto a los caracteres definidos, pueden ser también utilizados, disponiendo el modo **GRAPHICS (CAPS SHIFT + 9)** dentro del modo texto. La pulsación de **U** en el modo de pantalla principal, nos facilita la definición de cualquier UDG, con lo cual la pantalla se convierte en una enorme parrilla de definición.

Por último, existe la posibilidad de ampliar y reducir temporalmente el contenido de la pantalla (**M** y **N**), para mayor detalle, o bien ampliar o reducir la pantalla definitivamente **(CAPS SHIFT + 8)**.

En el primer caso, se trata simplemente de un ZOOM de la zona de la pantalla en que nos encontremos, que puede ampliar la imagen cuatro u ocho veces. En el segundo caso se trata de una modificación permanente para ampliar el dibujo de toda la pantalla, con la consiguiente pérdida de parte de la información.

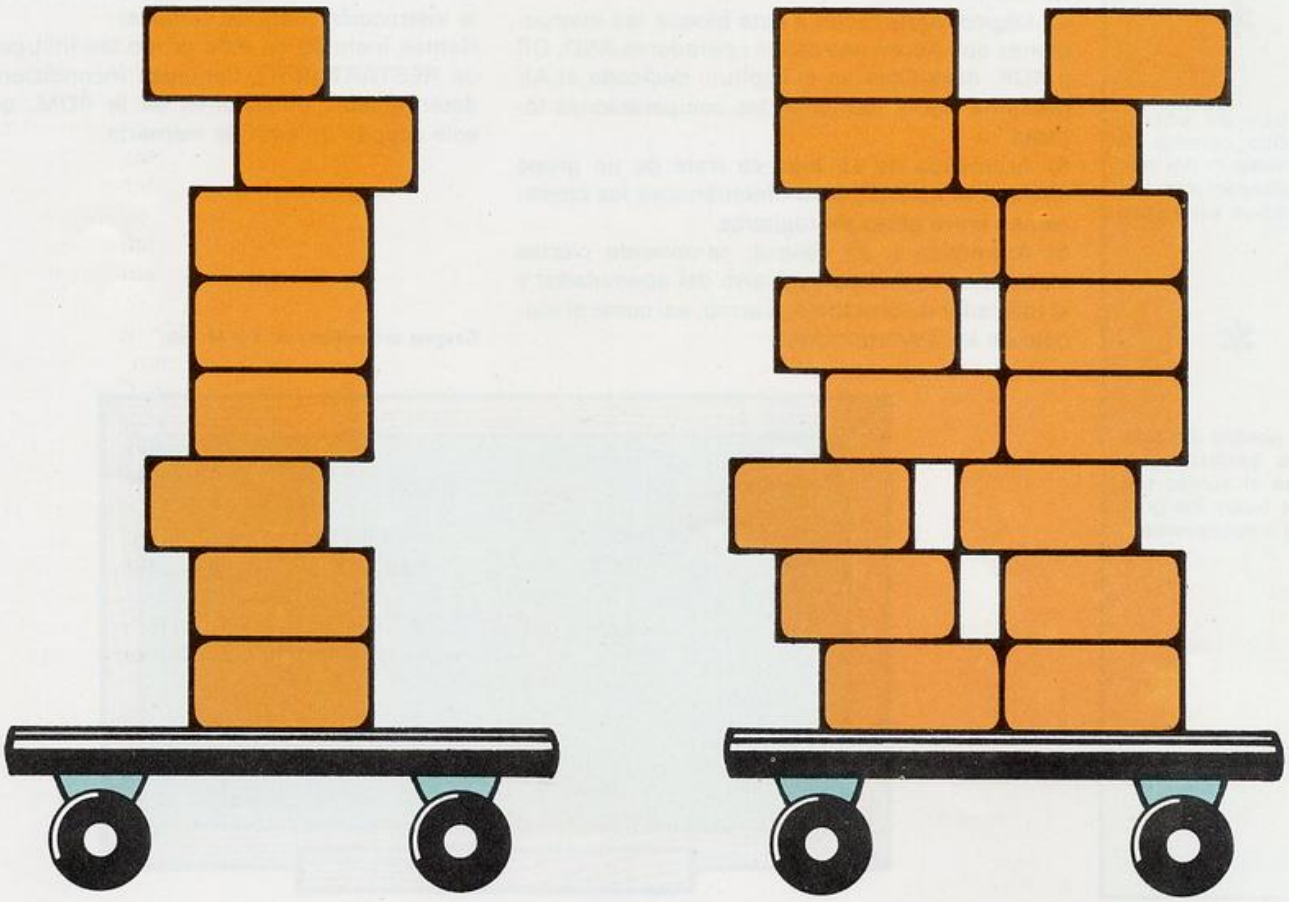
ANALIZANDO LOS GRUPOS

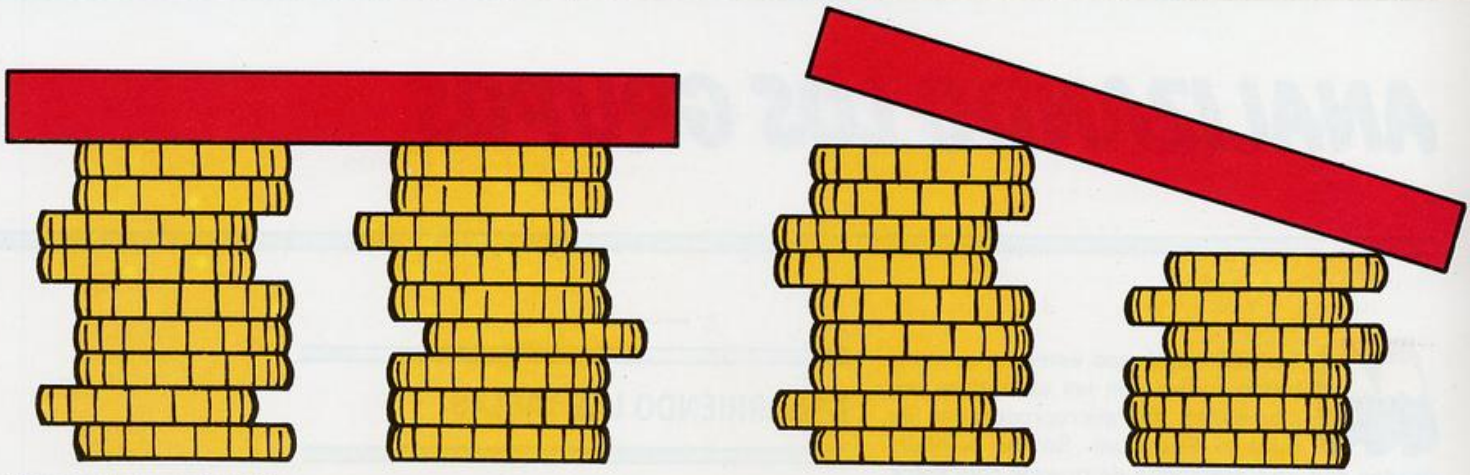
MUCHOS quizá nos estemos preguntando cuáles son las auténticas posibilidades del microprocesador de nuestro Spectrum. Somos bombardeados con una avalancha de nuevos conceptos y todavía no se ha justificado su utilidad.
Paciencia, fue necesario sentar unas sólidas bases antes de proseguir nuestro estudio del C/M. Efectuemos un recorrido a través de los diferentes grupos de instrucciones para hacernos una idea bastante aproximada de las operaciones que puede ejecutar el Z 80.

RECORRIENDO LAS TABLAS

Volvamos a las tablas del capítulo anterior. Allí encontraremos el juego de instrucciones completo del Z 80 (casi 700 instrucciones diferentes), las cuales hemos subdividido en los siguientes bloques:
1) Carga de 8 bits: en este grupo se hayan encuadradas todas las instrucciones de intercambio de información entre registros individuales, y entre registros y posiciones de memoria.

Grupos de carga de 8 y 16 bits.





¡!

Las instrucciones de manipulación de bit constituyen el más numeroso grupo entre las nuevas implementadas en el microprocesador Z 80.

*

El grupo de entrada/salida permite la comunicación del microprocesador con los dispositivos exteriores a él.

*

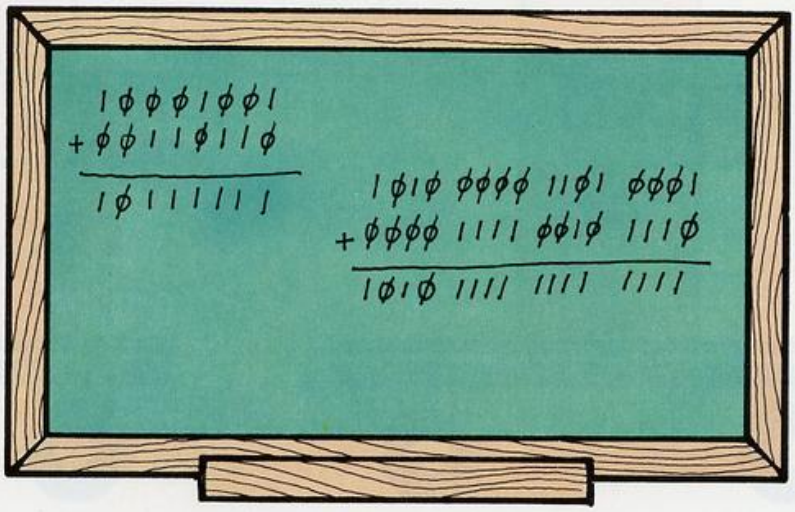
No es posible dar una norma general que permita el acceso común a todos los grupos de instrucciones.

2) Carga de 16 bits: como observaréis, muy pocas transferencias de información pueden efectuarse entre pares de registros. Sin embargo, a este grupo pertenecen las instrucciones PUSH y POP, fundamentales en el manejo del STACK o pila.
3) Aritmético de 8 bits: las operaciones de suma y resta entre cantidades de 8 bits son ejecutadas por las instrucciones de este grupo. Es posible también, incrementar o decrementar en uno el contenido de un determinado registro o dirección de memoria.
4) Lógico: pertenecen a este bloque las instrucciones que hacen uso de los operadores AND, OR y XOR, discutidos en el capítulo dedicado al Algebra de Boole, así como las comparaciones lógicas.
5) Aritmético de 16 bits: se trata de un grupo análogo al anterior pero efectuándose las operaciones entre pares de registros.
6) Aritmético y de control: comprende ciertas instrucciones de uso exclusivo del acumulador y el indicador de arrastre o acarreo, así como el manejo de las interrupciones.

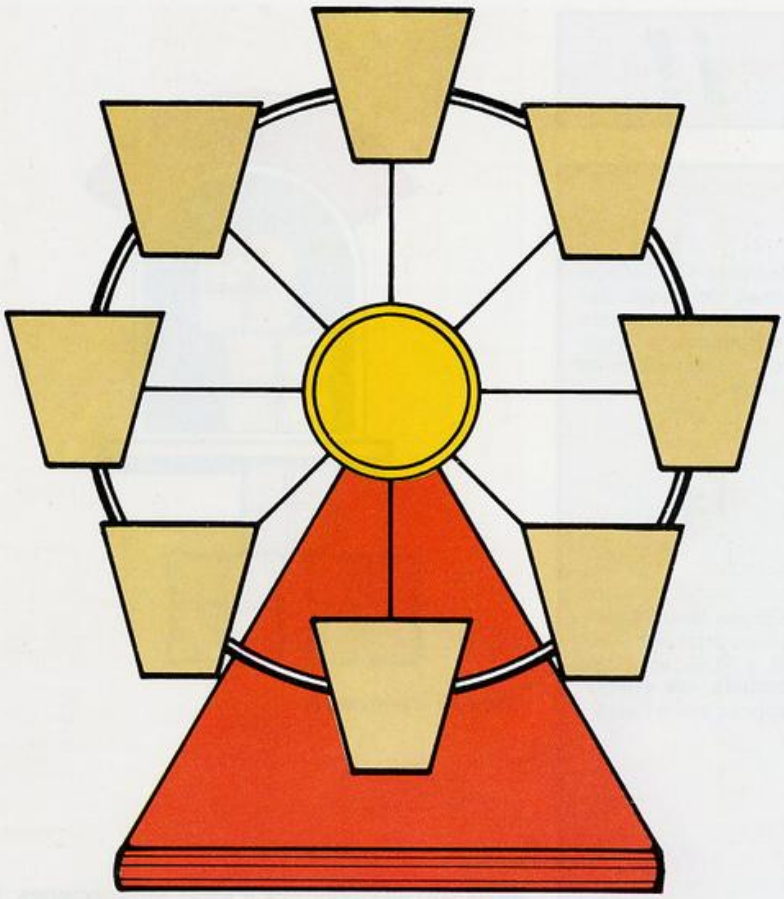
Grupo lógico.

7) Saltos y llamadas: permiten que el control del programa sea transferido a otra zona de éste, incondicionalmente o en determinados supuestos establecidos previamente, volviéndose al punto donde se efectuó la bifurcación, en el caso de que la instrucción fuera de llamada.
Hemos incluido en este grupo las instrucciones de RESTART (RST), llamadas incondicionales a determinadas direcciones de la ROM, que tan solo ocupan un byte de memoria.

Grupos aritméticos de 8 y 16 bits.



8) Retornos: las instrucciones de retorno permiten que el programa continúe su ejecución a partir del punto donde fue efectuada la última llamada. De la misma manera que en el grupo anterior, el regreso puede efectuarse según se cumplan o no algunas condiciones especiales.
9) Intercambio: en este grupo encontramos las únicas instrucciones del microprocesador Z 80 que permiten el manejo del grupo de registros alternativos (ARS).
10) Transferencia: facilitan el movimiento de bloques de datos desde unas posiciones de memoria a otras.
11) Búsqueda: permiten examinar un bloque de memoria con la intención de hallar un byte con el mismo contenido que el almacenado en el acumulador.
12) Tratamiento de bits: 240 nuevas instrucciones las cuales permiten manipular individualmente cada uno de los bits almacenados en cualquier registro o posición de memoria.
13) Rotación y desplazamiento: al igual que las anteriores, posibilitan la ejecución de operaciones a nivel de bit, dentro de cada octeto (byte).
14) Entrada/Salida: estas instrucciones habilitan al microprocesador para la comunicación con los dispositivos exteriores a él, recogiendo o enviando información desde o hacia ellos.
Nuestra curiosidad debe estar ahora satisfecha. No dudaremos que con semejante volumen de instrucciones, las posibilidades de programación que ofrece nuestro microprocesador son francamente fabulosas.
Pero no hemos hecho sino comenzar, y todavía hemos de seguir ampliando conocimientos antes de pasar al análisis de cada instrucción en particular. El camino puede resultar largo y complicado a principio, pero pronto estaremos en condiciones de obtener los primeros resultados.



Grupo de rotación y desplazamiento.

MANEJANDO LAS TABLAS

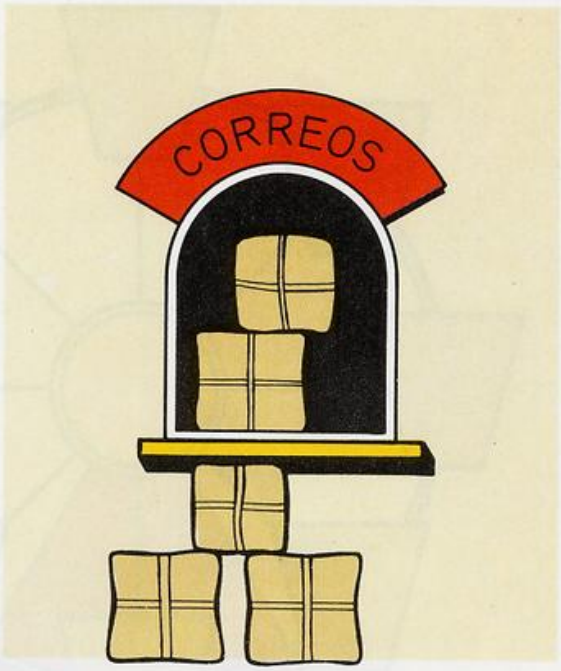
En las tablas podemos encontrar de manera rápida y clara los códigos de operación (en hexadecimal) correspondientes a todas las instrucciones implementables en el microprocesador de nuestro Spectrum.
Quizá nos encontremos algo sorprendidos por la anterior afirmación y no seamos capaces de ver en ellas otra cosa que un maremagnum de símbolos aparentemente sin sentido.
No es posible dar una norma general, la cual permita el acceso común a todos los grupos, pues como descubriremos cuando sean analizados cada uno en particular, su utilidad se hará patente cuando estudiemos su forma de operar.
Por ejemplo, tomemos el grupo de carga de 8 bits. Entre paréntesis está señalado que en lenguaje ensamblador todas estas instrucciones comienzan por LD (abreviatura de LoaD, cargar).
Supongamos que buscamos la codificación de la instrucción LD B,A la cual significa que introduzcamos en el registro B el contenido del registro A. En la parte superior de la tabla está señalada la palabra ORIGEN y a la izquierda DESTINO.
Por tanto, el destino es B y el remitente o fuente de la información es A. Entrando en la tabla por la fila señalada con B, buscamos la intersección con la columna marcada con A y en la casilla correspondiente encontramos que el código de operación asociado al ensamblador LD B,A es 47h.
En otros grupos se define directamente el mnemónico de la instrucción, como en el caso de los saltos y llamadas. Supongamos que la instrucción a codificar es CALL 7FFF, es decir una llamada a la subrutina ubicada a partir de la dirección 7FFF hexadecimal o 32767 decimal.
En la columna de la izquierda están los mnemó-

¡!

Mediante las instrucciones de carga, son transferidos datos entre registro, o registros, y posiciones de memoria.

*

El grupo lógico habilita los operandos AND, OR y XOR, así como posibilita las comparaciones entre datos.



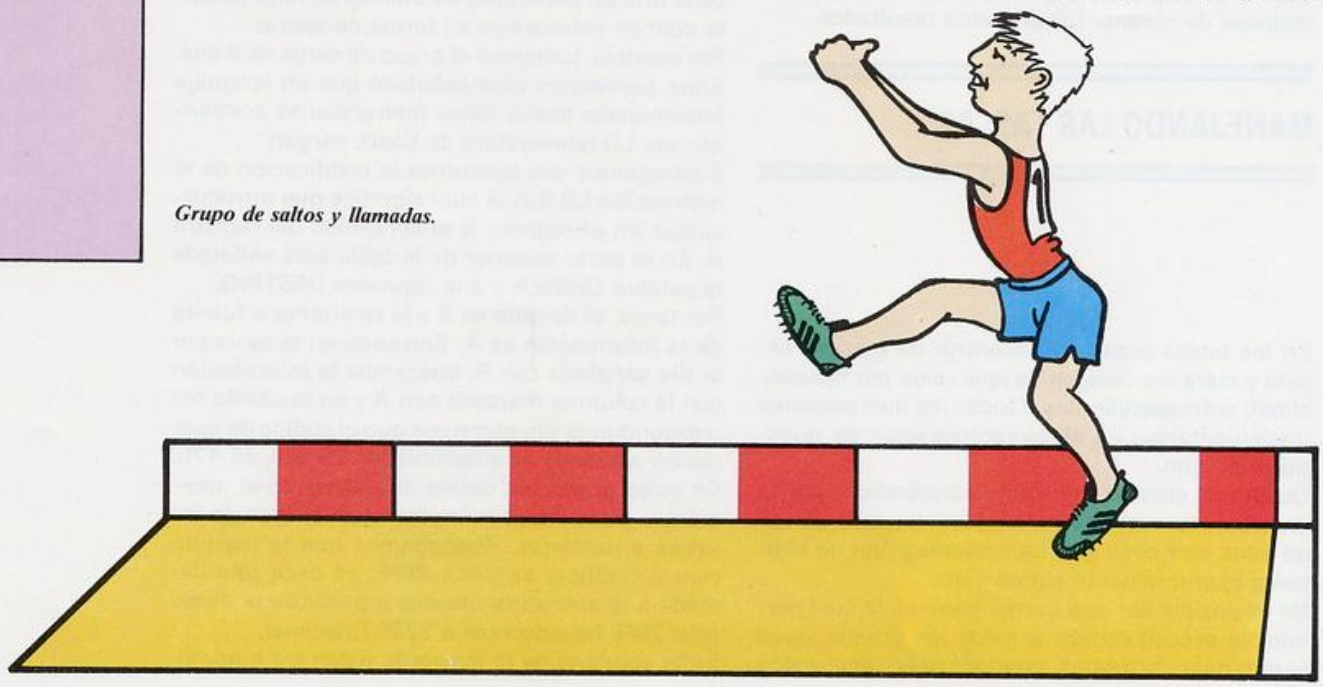
Grupo de transferencia.

nicos correspondientes a estas instrucciones. En la sexta fila encontramos CALL. La llamada ha sido incondicional a la dirección especificada, a continuación seguirá el formato CALL nn, siendo *nn* los dos bytes que definen dicha dirección.
Luego la codificación correcta será CD FF 7F. No nos sorprendamos si encontramos los dos bytes con su posición intercambiada. Ya aclaramos en su momento, que el microprocesador cuando se trata de una dirección espera encontrar en primer lugar siempre el byte de menor peso.
Otros símbolos utilizados son, por ejemplo, la «d» que representa el byte de desplazamiento en las instrucciones que manejan el direccionamiento indexado, o la «e», la cual representa la extensión del salto en el direccionamiento relativo (los modos de direccionamiento serán el tema principal de nuestro próximo capítulo).

CARGADOR DE C/M

Para terminar el presente capítulo, hemos preparado un pequeño programa, el cual actuará de cargador hexadecimal para todas las rutinas en C/M con las que pondremos en práctica los ejemplos necesarios al explicar los distintos grupos de instrucciones.
La elección del sistema hexadecimal nos ha parecido la más adecuada, dada la extensa bibliografía existente que lo utiliza, en especial la literatura dedicada a la descripción de las rutinas almacenadas en la ROM.
Los códigos hexadecimales correspondientes a las instrucciones que configuren nuestros programas estarán incluidos en las sentencias **DATA** que situaremos al principio del programa,

Grupo de saltos y llamadas.



y efectuando **RUN 9000**, quedarán almacenados en la memoria de nuestro ordenador para su posterior ejecución.

En caso de cometer algún error, se nos indicará la sentencia donde ocurrió para modificarla, mediante un mensaje intermitente que aparecerá sobre la pantalla del monitor.

Emplearemos el modo mayúsculas al introducir los caracteres hexadecimales que conformen nuestra rutina, siendo siempre los dos últimos de esta **, de modo que el programa reconozca que no ha de seguir almacenando más código en la memoria.

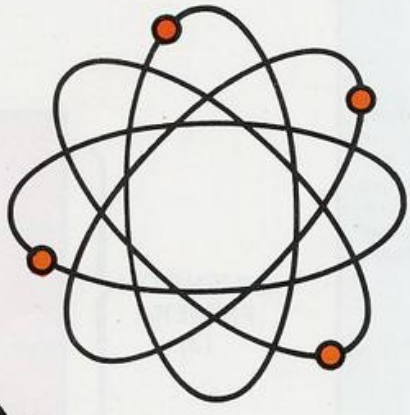
Al comienzo se nos preguntará sobre la posición de memoria donde deseamos introducir la rutina en C/M. En principio, toda la RAM podría utilizarse, pero para evitar posibles conflictos con las Variables del Sistema y otras zonas delicadas de éste, elegiremos posiciones de memoria cercanas al RAMTOP donde ejecutar nuestros programas, salvo que no se indique lo contrario.

Una vez almacenado el C/M es posible grabarlo utilizando el comando BASIC **SAVE "Nombre" CODE** dirección de inicio, longitud, siendo dirección de inicio la elegida por nosotros previamente y longitud, el número de bytes ocupados por la rutina (al finalizar la ejecución del programa cargador se nos informará también de este extremo en la pantalla).

A continuación encontraremos el listado del programa cargador. Tecleémoslo cuidadosamente y grabémoslo antes de ponerlo en funcionamiento.

```
9000 REM *CARGADOR HEXADECIMAL*
9010 REM *CARLOS DE LA OSSA VILLACANAS*
9020 DEF FN D(H$)=CODE H$-55+7*(H$<"A")
9030 DEF FN B(H$)=16*FN D(H$(1))+FN D(H$
(2))
9040 LET D$="SENTENCIA DATA No.- "
9050 LET DATA=0:" LET CONT=0: LET M=1
9060 INPUT "DIRECCION DE UBICACION  ",DI
R
9070 IF NOT M THEN  PRINT : PRINT  BRIGH
T 1;"COMPLETADA LA CARGA": PRINT : PRINT
 "LONGITUD, ";CONT;" BYTES": STOP
9080 LET SUMA=0: LET DATA=DATA+10
9090 READ A$,TOTAL
9100 FOR I=1 TO LEN A$-1 STEP 3
9110 IF A$(I)=CHR$ 42 THEN  LET M=0: GO
TO 9150
9120 LET H$=A$(I TO I+1)
9130 POKE DIR,FN B(H$): LET DIR=DIR+1: L
ET SUMA=SUMA+FN B(H$): LET CONT=CONT+1
9140 NEXT I
9150 IF SUMA<>TOTAL THEN  PRINT D$;DATA;
CHR$ 32; FLASH 1;"ERRONEA": GO TO 9070
9160 PRINT D$;DATA;CHR$ 32;"CORRECTA": G
O TO 9070
```

Grupo de manipulación de bits.



SALTO DE CABALLO

¡!

Según las características requeridas al pasatiempo, el programa tardará más o menos tiempo en su confección, aunque para aumentar su velocidad, hasta han sido eliminadas las llamadas a la subrutina de interrupción.

*

En caso de que no sea posible realizar un salto de caballo con las características que indiquemos, el programa nos lo hará saber mediante un mensaje.

*

Para la grabación del programa utilizaremos **SAVE ''S.CABALLO''**, o **SAVE "S.CABALLO" LINE 1**, si deseamos activar el sistema de autoejecución.

ES probable que muchos de nosotros compartamos una afición mutua: el sano deporte mental del «salto de caballo». Pues bien, con este programa podremos generar, a plena voluntad, todos los saltos de caballo que deseemos.

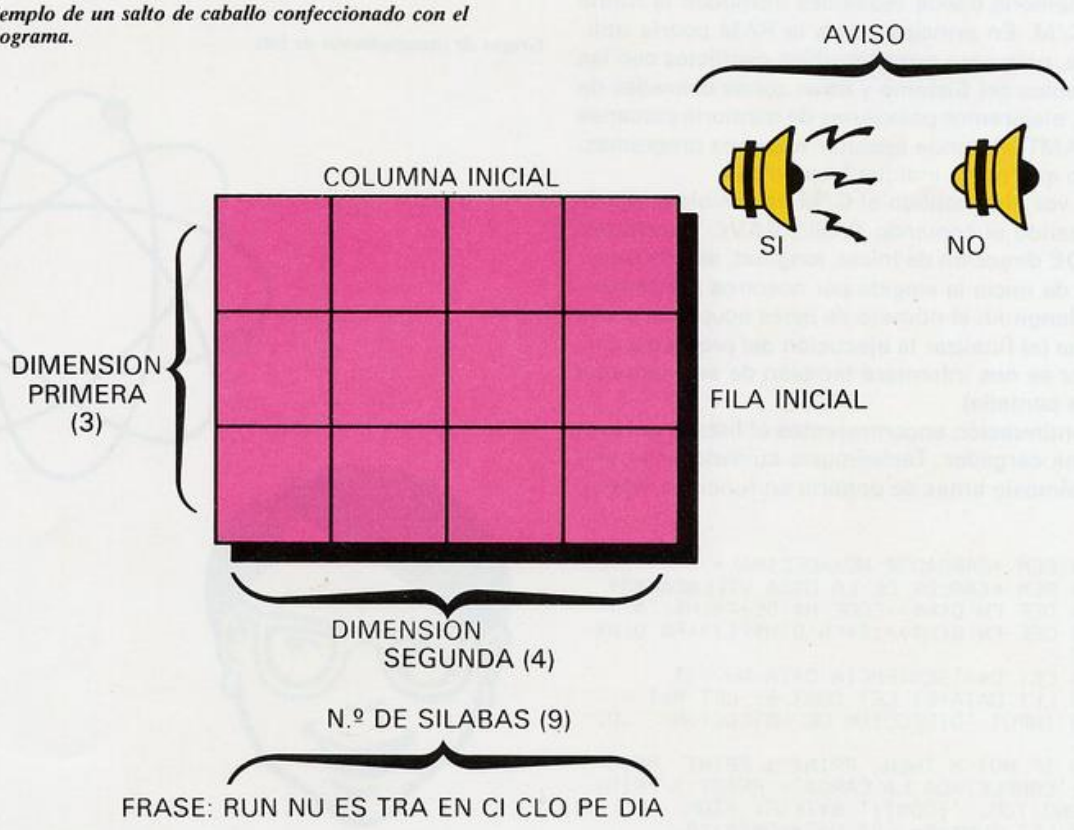
EL PROGRAMA

Este programa se compone de un soporte BASIC muy simple, que sirve para gestionar la toma de datos y para la visualización del resultado generado por la auténtica inteligencia del programa: la rutina en código máquina.

Los datos que ésta emplea para la confección del pasatiempo son los siguientes:

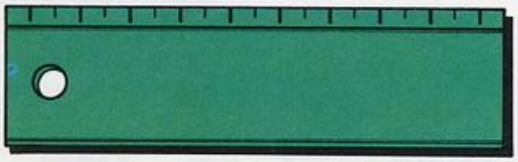
1. «FRASE»: a este INPUT debemos responder con la frase que deseamos utilizar en el salto de caballo, SEPARANDO SUS SILABAS MEDIANTE UN ESPACIO, y teniendo en cuenta que no son admisibles sílabas de más de tres letras.
2. «DIMENSION PRIMERA»: número de filas de que se compone el salto de caballo.
3. «DIMENSION SEGUNDA»: número de columnas de que se compone el pasatiempo.
4. «NUMERO DE SILABAS»: número de sílabas que se utilizan en el salto. Si este número es superior al total de sílabas que integran la frase, el cuadro del pasatiempo será completado con espacios vacíos.
5. «FILA INICIAL»: fila de la matriz elegida para

Ejemplo de un salto de caballo confeccionado con el programa.

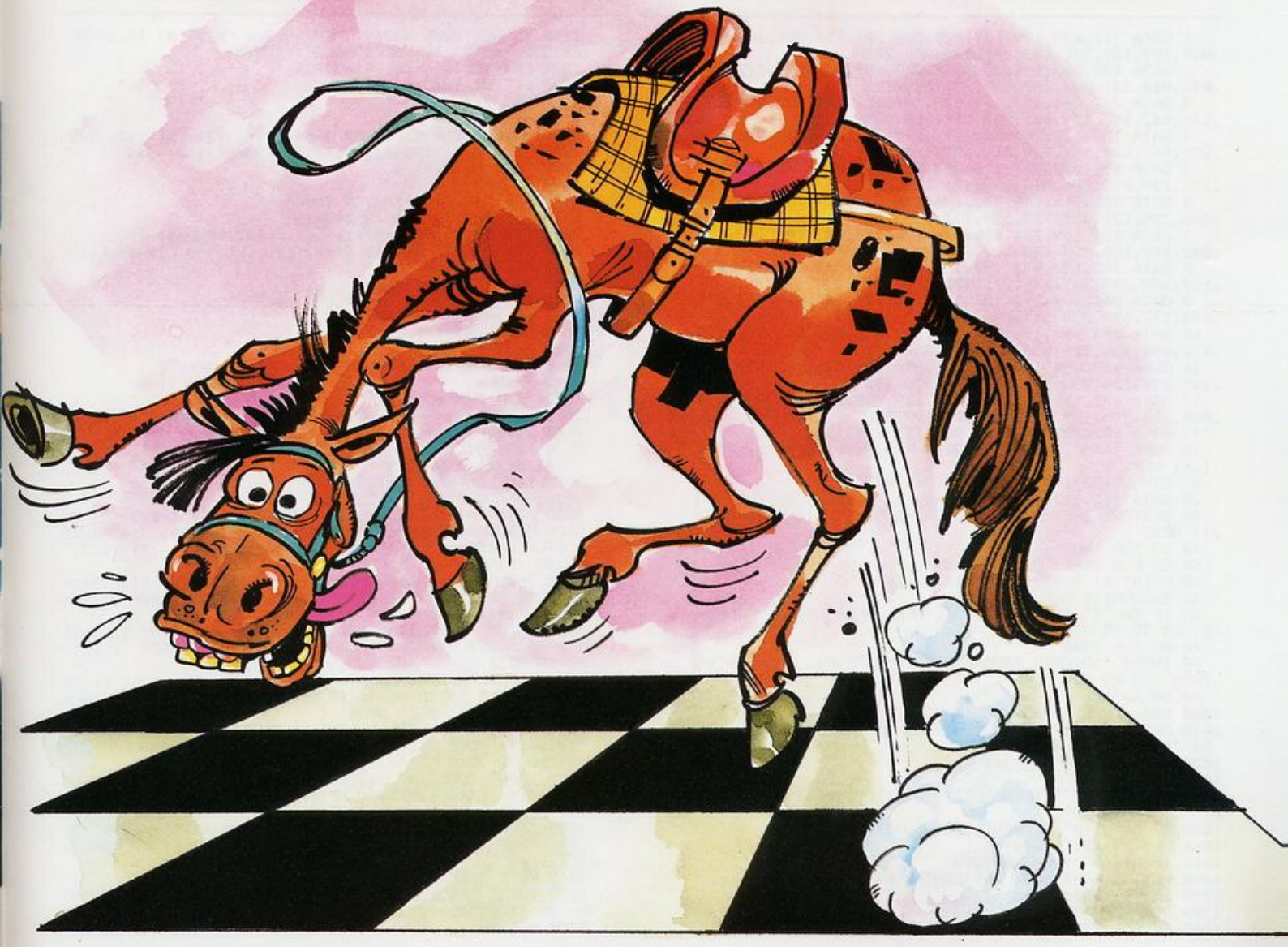


el comienzo del salto. La primera columna es la cero.
6. «COLUMNA INICIAL»: igual que «fila inicial», pero referido a las columnas.
7. «AVISO S/N»: emisión o no de un pitido intermitente como aviso de que ha finalizado la confección del salto de caballo.
La rutina en C/M actúa poniendo en práctica técnicas recursivas, gracias a lo cual su ocupación de espacio es mínima. A partir de la casilla de origen, el programa comprueba todas las series de

RUN ES ME JOR



Las sílabas que integren la frase a tratar no deben sobrepasar en ningún caso los tres caracteres.



movimientos posibles, escogiéndolos de forma aleatoria.

Si al final de la secuencia elegida, no localiza un salto posible, da un paso atrás en su camino y comprueba otra nueva vía.

Pese al sistema de programación empleado, el tiempo de confección de un pasatiempo puede ser grande, si las características que hemos requerido de él así lo exigen. De igual modo, es posible que el programa llegue a la conclusión de la imposibilidad material de realizar el salto de caballo deseado, cuestión ésta que nos anunciará mediante la emisión del mensaje: "NO existe tal salto de caballo".

Para reducir el tiempo de confección del salto podemos emplear un pequeño truco: definir un cuadrado con más casillas que sílabas tiene la frase a tratar. De este modo, se simplificará la labor de distribución por parte del programa, que completará las casillas sobrantes con tres asteriscos (***).

Para grabar el programa, basta con la utilización del siguiente comando: SAVE "S.CABALLO". Si prefiriésemos la opción con autoejecución, teclearíamos la orden de grabación de la siguiente forma: **SAVE "S.CABALLO" LINE 1.**

Para confeccionar un salto de caballo es necesario expresar la frase a tratar con sus silabas separadas por espacios.

RUN NU ES TRA EN CI CLO PE DIA

```
  1 DATA 221,033,199,129,221,009,221,126,000,201,221
,033,207,129,221,009,221,126,000,201
  2 DATA 120,203,039,203,039,203,039,203,039,177,079
,006,000,221,033,207,130,221,009,201
  3 DATA 237,091,197,129,033,187,129,001,005,000,237
,176,042,197,129,001,005,000,009,034
  4 DATA 197,129,201,042,197,129,001,005,000,055,063
,237,066,034,197,129,017,187,129,001
  5 DATA 005,000,237,176,201,062,255,050,191,129,033
,191,129,052,006,000,033,191,129,078
  6 DATA 205,215,129,033,190,129,134,033,188,129,119
,205,225,129,033,189,129,134,033,187
  7 DATA 129,119,254,000,250,192,130,033,193,129,150
,242,192,130,058,188,129,254,000,250
  8 DATA 192,130,033,192,129,150,242,192,130,237,075
,187,129,205,235,129,221,126,000,214
  9 DATA 000,194,192,130,058,196,129,221,119,000,033
,194,129,190,250,144,130,033,195,129
 10 DATA 054,001,195,192,130,205,255,129,033,196,129
,052,058,188,129,050,190,129,058,187
 11 DATA 129,050,189,129,205,044,130,033,196,129,053
,205,022,130,058,195,129,254,000,194
 12 DATA 192,130,237,075,187,129,205,235,129,221,054
,000,000,058,195,129,254,000,192,058
 13 DATA 191,129,254,007,200,195,049,130
 14 POKE 23658,8: FOR I=33239 TO 33486: READ X: POKE
 I,X: NEXT I
 17 DIM L$(32): DIM A(8,2): DIM B$(256,3)
 20 DATA 2,1,1,2,-1,2,-2,1,-2,-1,-1,-2,1,-2,2,-1
 30 RESTORE 20: FOR I=1 TO 8: READ A(I,1): READ A(I,
2): NEXT I
 40 GO SUB 1000
 50 GO SUB 2000: GO SUB 2525
 55 PRINT AT 3,12; FLASH 1;"ESPERA"
 60 RANDOMIZE USR 33324
 65 IF Q$="S" AND INKEY$="" THEN  BEEP .8,20: PAUSE
10: GO TO 65
 70 GO SUB 1500
 80 INPUT "QUIERES PROBAR OTRA VEZ? ";R$: IF R$<>"N"
 THEN  GO TO 50
 90 GO TO 10000
1000 REM  INPUT FRASE
1010 PAPER 6: BORDER 6: CLS : INPUT "FRASE: ";A$: LET
 A$="*** "+A$+" $ "
1020 LET CONT=1: LET J=1
1030 LET I=J
1040 IF A$(J)=" " THEN  LET J=J+1: LET I=J: GO TO 104
0
1050 IF A$(J)<>" " THEN  LET J=J+1: GO TO 1050
1060 LET C$=A$(I TO J-1)
1070 IF C$="$" THEN  RETURN
1080 LET B$(CONT)=C$: LET CONT=CONT+1
1090 GO TO 1030
1500 REM  SALTO
1505 GO SUB 2500
1510 IF NOT PEEK 33219 THEN  CLS : PRINT AT 10,2;"NO
EXISTE TAL SALTO DE CABALLO": RETURN
1520 LET Y1=INT ((31-N1*3)/2)
1530 LET X1=INT ((21-N2)/2)
1540 FOR I=0 TO N1-1
1550 FOR J=0 TO N2-1
1560 PRINT  INK 9; PAPER 1+6*(((I+J)/2)=INT ((I+J)/2)
);AT X1+J+1,Y1+I*3+1;B$(1+PEEK (33487+16*I+J))
1570 NEXT J: NEXT I
1580 RETURN
2000 REM  INICIALIZACION TABLAS
2010 FOR I=1 TO INT (RND*5)+1
2020 LET J=INT (RND*8)+1: LET K=INT (RND*8)+1
2030 FOR L=1 TO 2: LET X=A(J,L): LET A(J,L)=A(K,L): L
ET A(K,L)=X: NEXT L
2040 NEXT I
2050 FOR I=0 TO 7
2060 POKE 33223+I,A(I+1,1)
2070 POKE 33231+I,A(I+1,2)
2080 NEXT I
2090 FOR I=33487 TO 33743: POKE I,0: NEXT I
2100 RETURN
2500 REM  PANTALLA
2510 BORDER 2: PAPER 5: CLS : PRINT  PAPER 6;AT 0,0;L
$;AT 21,0;L$
2520 FOR I=1 TO 21: PRINT  PAPER 6;AT I,0;" ";AT I,31
;" ": NEXT I: RETURN
2525 REM  ENTRADA DATOS
2526 GO SUB 2500
2530 PRINT AT 6,5;"DIMENSION PRIMERA: ";AT 8,5;"DIMEN
SION SEGUNDA: ";AT 10,5;"NUMERO DE SILABAS: ";AT 12,5
;"FILA INICIAL:";AT 14,5;"COLUMNA INICIAL:";AT 16,5;"
AVISO (S/N):"
3010 INPUT "DIMENSION PRIMERA: ";N2;: PRINT AT 6,24;N
2: POKE 33216,N2: IF N2<=0 OR N2>10 THEN  BEEP .2,20:
 GO TO 3010
3020 INPUT "DIMENSION SEGUNDA: ";N1: PRINT AT 8,24;N1
: POKE 33217,N1: IF N1<=0 OR N1>16 THEN  BEEP .2,40:
GO TO 3020
3030 INPUT "NUMERO DE SILABAS: ";NCUA: PRINT AT 10,24
;NCUA: POKE 33218,NCUA: IF NCUA<=0 OR NCUA>N1*N2 THEN
  BEEP .2,40: GO TO 3030
3040 INPUT "FILA DE LA CASILLA INICIAL: ";Y: PRINT AT
 12,24;Y: POKE 33214,Y: IF Y<0 OR Y>N1 THEN  BEEP .2,
40: GO TO 3040
3050 INPUT "COL. DE LA CASILLA INICIAL: ";X: PRINT AT
 14,24;X: POKE 33213,X: IF X<0 OR X>N2 THEN  BEEP .2,
40: GO TO 3050
3055 INPUT "AVISO? (S/N): ";Q$: PRINT AT 16,24;Q$
3060 POKE 33219,0: POKE 33220,2: POKE 33221,0: POKE 3
3222,144
3065 POKE 33487+X*16+Y,1
3070 INPUT "ESTA BIEN ASI? (S/N): ";R$: IF R$="N" THE
N  GO TO 2526
3080 RETURN
```