23
150pts.

RUN

Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

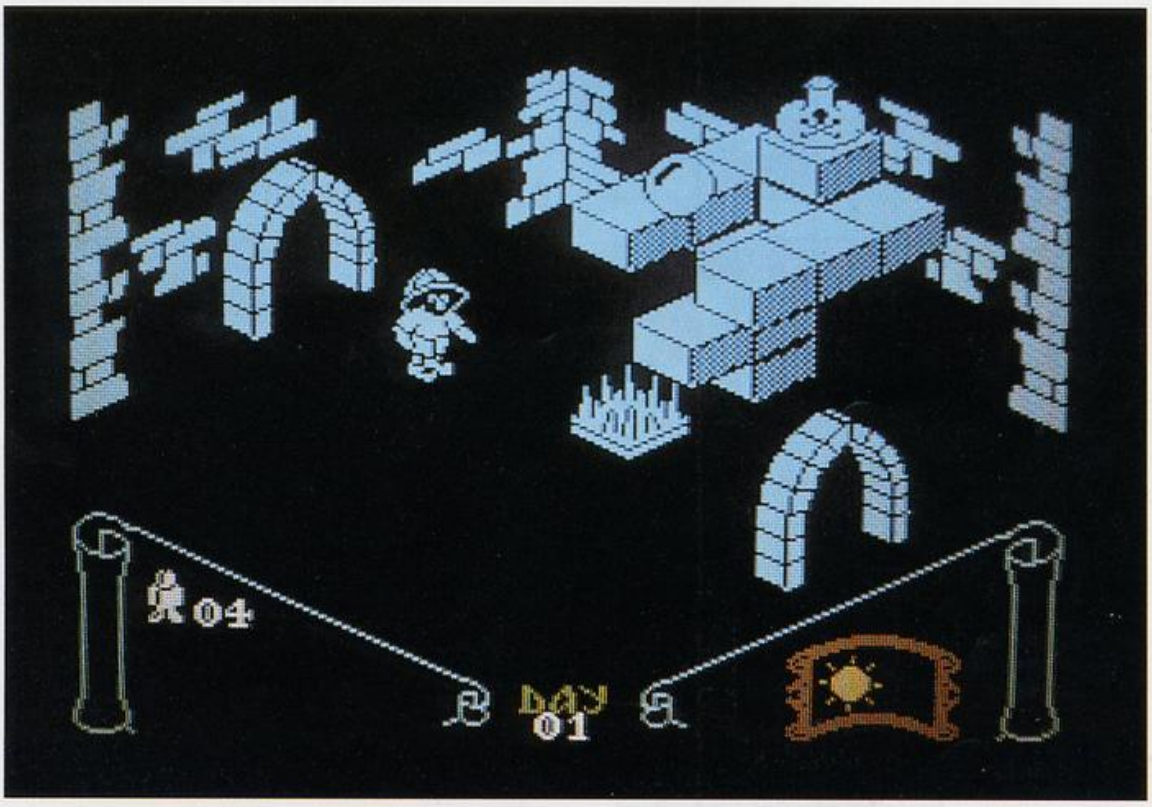
U.D.G.

ESTA es la abreviatura inglesa de *User Definable Graphics* (Gráficos Definibles por el Usuario). Bajo este nombre se encuadran los veintiún caracteres gráficos del Spectrum, comprendidos entre los códigos 144 y 164, ambos inclusive, cuya forma puede ser seleccionada por nosotros mismos.
La definición de los caracteres gráficos no es una tarea que implique una gran complicación; se puede decir incluso que es bastante simple, si bien es cierto que si deseamos obtener resultados de una cierta calidad, requiere emplear una buena cantidad de tiempo en la labor de diseño. Estudiemos el proceso de generación de un gráfico, paso a paso.

La definición de caracteres es una tarea bastante simple, aunque precisa mucho tiempo para alcanzar resultados satisfactorios.

APARECE EL PROBLEMA

Nos encontramos ante el proyecto de llevar a cabo un programa educativo, que al estar destinado a la juventud, debe cuidar en su máximo extremo la ortografía y legibilidad de sus textos. Así pues, y dado que el Spectrum carece de algunos signos ortográficos castellanos, como las vocales acentuadas, la u con diéresis, etc., nos vemos en la obligación de buscar una solución que nos permita, de la forma más elegante, cómoda y rápida posible, obtener dichos caracteres.
En primer lugar, es necesario que conozcamos



¡!

U.D.G. es la abreviatura inglesa de *User Definable Graphics*: Gráficos Definibles por el usuario.

*

Aunque la definición de los caracteres no es una tarea muy complicada, sí precisa de bastante tiempo para la obtención de resultados de cierta calidad.

*

La obtención de caracteres especiales se puede conseguir por dos sistemas: combinación de caracteres predefinidos o generación de gráficos definibles.

¡!

El proceso de generación de un gráfico comienza con el diseño del mismo. En esta fase nos pueden resultar de gran ayu : los programas generadores de caracteres incluidos en las cintas de demostración de Sinclair.

*

Los ceros a la izquierda no tienen efecto alguno en el sistema de numeración binario.

exactamente la magnitud de nuestro problema, es decir, los caracteres concretos que nos van a ser necesarios:

— las cinco vocales acentuadas
— la u con diéresis
— la letra eñe, tanto mayúscula como minúscula
— el signo de apertura de interrogación
— el signo de apertura de admiración

Una vez hecho esto, debemos proceder a discernir cual es el método por el cual intentaremos conseguir nuestro objetivo.

Hace bien poco, pudimos ver como mediante la función **OVER**, nos era posible obtener tanto las vocales acentuadas, como la eñe minúscula, ahorrándonos el trabajo de la definición de caracteres, si bien es cierto que con un resultado bastante menos estético. En nuestro caso concreto, queda descartado este sistema, pues no nos posibilita la obtención de algunos de los caracteres que nos son indispensables.

Teniendo que proceder a la definición de caracteres, ¿tenemos suficiente capacidad con los U.D.G., o será necesaria una reubicación del juego de caracteres? Aunque aún no alcancemos a comprender la diferencia entre estos dos sistemas, diremos que si bien es más sencilla la definición de U.D.G., estos son sólo veintiuno, y en algunos casos serán insuficientes.

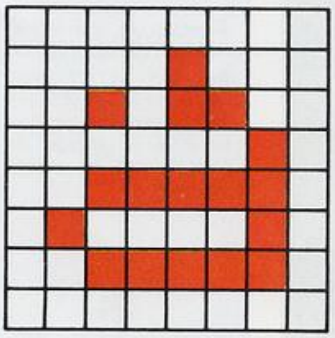
El Spectrum carece de ciertos caracteres o signos de uso frecuente en el castellano (vocales acentuadas, eñe, etc...).

Por otra parte, cuando al final de este capítulo estudiemos el sistema de reubicación del juego de caracteres, veremos que su manejo es ligeramente más complicado, y sobre todo, ocupa bastante más memoria.

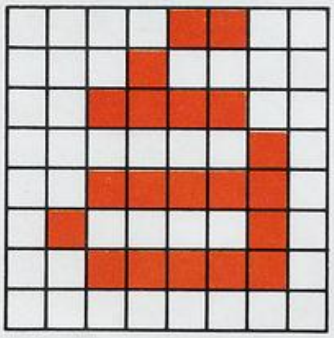
Así pues, dado que el número de caracteres que precisamos se limita a diez, tenemos la suficiente capacidad en los U.D.G. sin necesidad de recurrir a mayores complicaciones. Una vez decidido el sistema de obtención, llega la fase de definición de los caracteres.

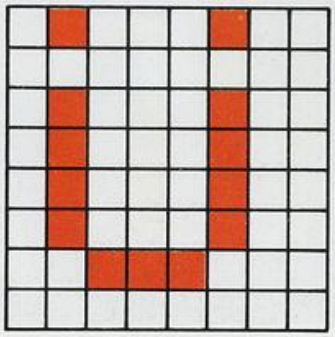
Aunque algunos caracteres especiales se pueden conseguir por combinación de predefinidos, otros están vedados a este sistema.

PREDEFINIDO



DEFINIDO





DISEÑO DE CARACTERES GRAFICOS

Para diseñar un carácter gráfico, basta con que nos hagamos con una hoja de papel cuadriculado, y marquemos sobre ella una zona de ocho por ocho cuadrículas.

Hecho esto, debemos dibujar la silueta del carácter en cuestión, con el símbolo que deseamos. Aunque el diseño del carácter así obtenido puede parecer, en principio, bastante tosco, no debe preocuparnos, puesto que la ampliación proporcionada por el papel cuadriculado es bastante grande, y su aspecto parecerá mucho más uniforme cuando se vea representado a su tamaño normal en la pantalla.

Concluida la operación de diseño, que sin duda es la más ardua hasta obtener un resultado lo suficientemente satisfactorio, el aspecto de nuestro dibujo debe ser algo similar a un tablero de ajedrez muy particular, en el cual los puntos ocupados tomarán el color del atributo **INK**, y los que permanecen libres el del **PAPER**.

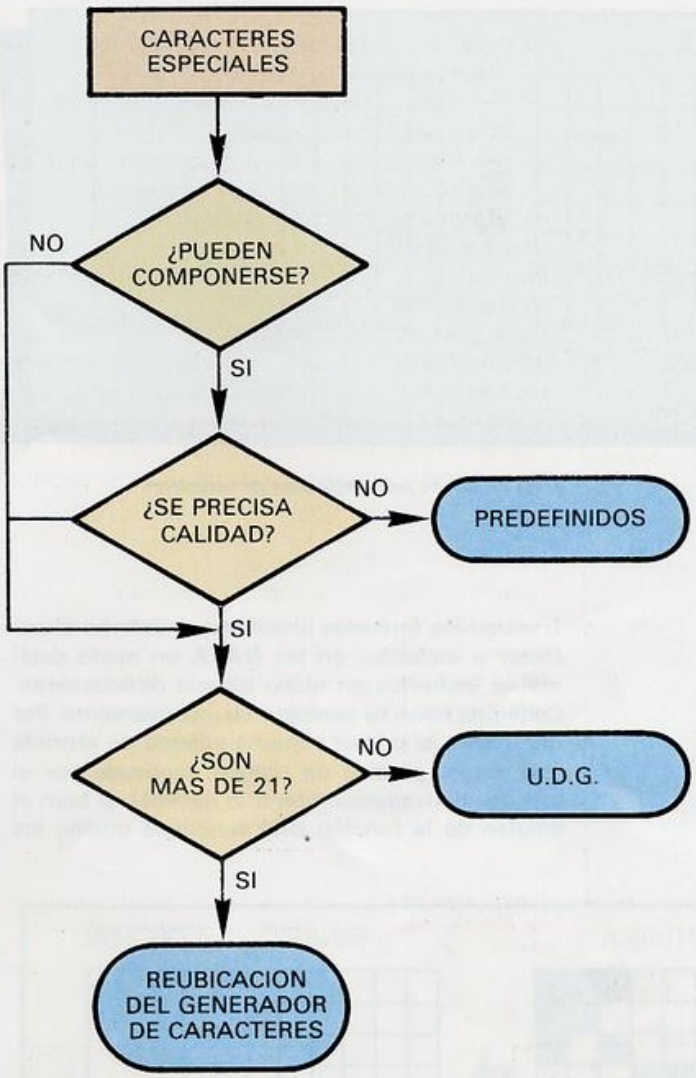
La tarea que debemos emprender a continuación es la de traducir nuestro carácter a un formato inteligible por el Spectrum. Puesto que nuestro ordenador, como todos los de su clase, sólo entiende de «unos» y «ceros», es decir, el lenguaje binario, debemos codificarle el carácter de esta forma.

Esto es mucho más fácil de lo que parece, puesto que se limita a sustituir los puntos coloreados de la cuadrícula por «unos» y los que permanecen en blanco por «ceros». De esta forma, cada una de las ocho líneas que componen el carácter se mostrará como una sucesión de 8 elementos, con valores 1 ó 0.

Así por ejemplo, una «a» acentuada podría tener el siguiente formato:

00001100
00010000
01111100
00000010
00111110
01000010
00111111
00000000

En este proceso, nos pueden servir de ayuda los denominados PROGRAMAS GENERADORES DE CARACTERES. El fabricante de nuestro aparato, consciente de la importancia de esta faceta, ha incorporado en sus cintas de demostración, tanto del ZX Spectrum como del Plus, un programa de este tipo, que no ayudará considerablemente en la fase de diseño.

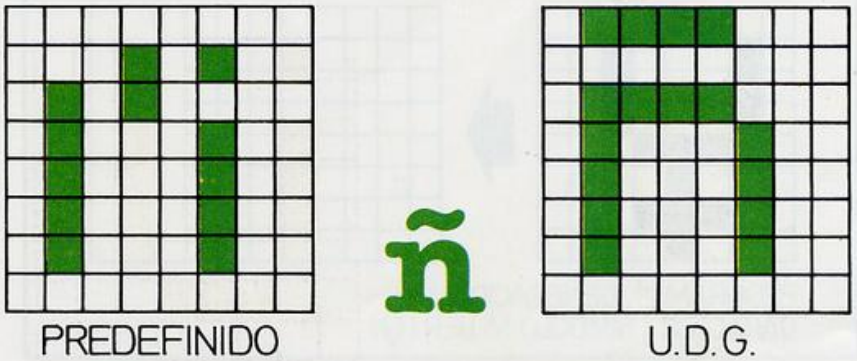


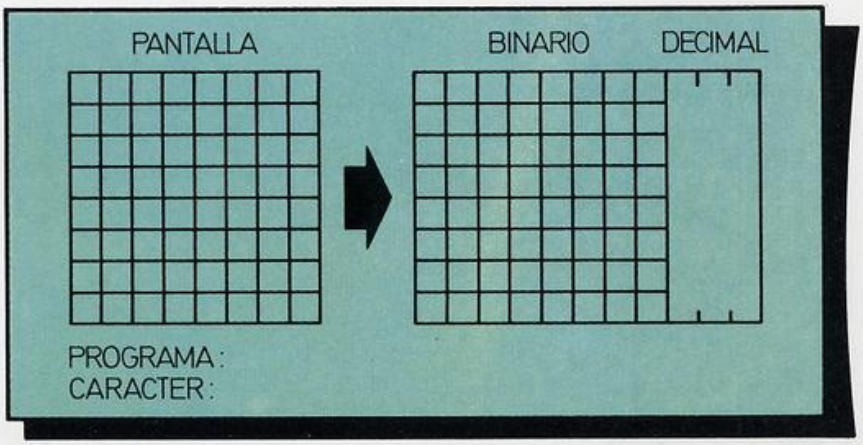
La elección de un método de obtención de caracteres especiales, se basa fundamentalmente en la calidad que se espere obtener, y la cantidad de ellos.

DEFINIENDO CARACTERES

Aun en el caso de poder utilizar combinaciones de caracteres predefinidos, es considerablemente mayor la calidad obtenida con los U.D.G.

La definición de un carácter concreto, nos da paso a la obtención de símbolos que no podemos encontrar en el teclado. El sistema más cómodo para incorporar los datos de los caracteres definidos a un programa es mediante sentencias **DATA**. Así pues, dado que la función BASIC **BIN** nos permite traducir números binarios a decimales, podremos optar por dos soluciones:





Hoja de diseño para definición de caracteres.

Traducir los formatos binarios que definen el carácter e incluirlos en las **DATA** en modo decimal, o incluirlos en modo binario directamente. Cada una tiene su ventaja y su inconveniente. Por una parte, el primer sistema supone de entrada una mayor pérdida de tiempo, motivada por el proceso de traducción binario decimal, si bien el empleo de la función **BIN** simplifica mucho los trámites. Por otra parte, dado que la ocupación del byte en binario es siempre mayor que en decimal, la inclusión de las **DATA** con este formato hace más pesada la labor de introducción del programa, y consume mayor cantidad de memoria.

Nosotros optaremos en nuestros ejemplos por el formato decimal; recurrimos a la estructura **PRINT BIN...** para ir averiguando los valores decimales de las líneas. Siguiendo con el ejemplo anterior:

```
PRINT BIN 00001100
PRINT BIN 00010000
PRINT BIN 01111100
PRINT BIN 00000010
PRINT BIN 00111110
PRINT BIN 01000010
PRINT BIN 00111111
PRINT BIN 00000000
```

Aunque no tengan efecto alguno, añadimos los ceros a la izquierda para representar todos los puntos del carácter. Los resultados de las operaciones del ejemplo serán, respectivamente: 12, 16, 124, 2, 62, 66, 63 y 0.

Una vez incluidas las sentencias **DATA** en el programa, es necesario conocer la forma de ubicar la definición del carácter. Para ello, debemos recurrir a una combinación de funciones del BASIC muy específica, denominada **POKE USR**, que adopta la forma general:

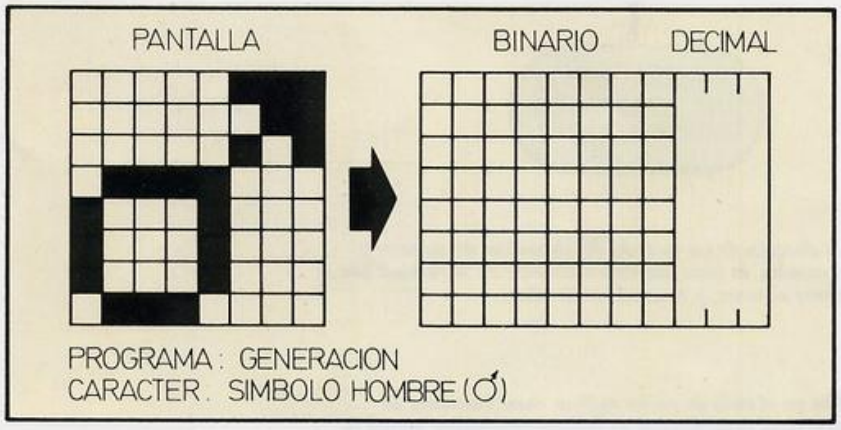
```
POKE USR "X",Y
```

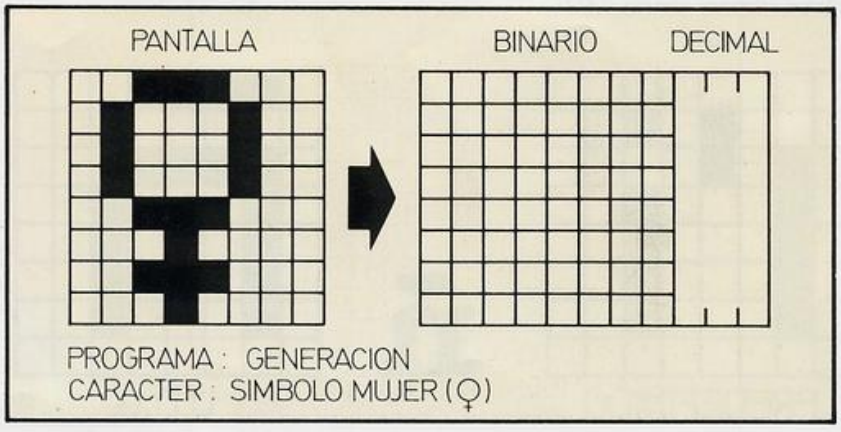
Donde **"X"** indica el carácter gráfico que se desea generar, e **Y** la configuración decimal de cada una de las ocho líneas que definen el carácter, o la binaria precedida de la función **BIN**.

Realmente, el conjunto **USR "X"** (siendo X un carácter de la A a la U), nos ofrece la posición de comienzo en memoria del gráfico correspondiente a la letra. Por lo tanto, en esa posición y las 7 siguientes, deben efectuarse las asignaciones de valores de cada una de las 8 líneas que componen cada carácter gráfico.

Estas asignaciones, tienen lugar por medio de la sentencia **POKE**, que coloca en la posición señalada por el primer parámetro **(USR "X")** el valor del segundo **(Y)**.

Veamos a continuación un ejemplo de ello en el





La primera fase del proceso de generación de gráficos, corresponde al diseño de los caracteres necesarios.

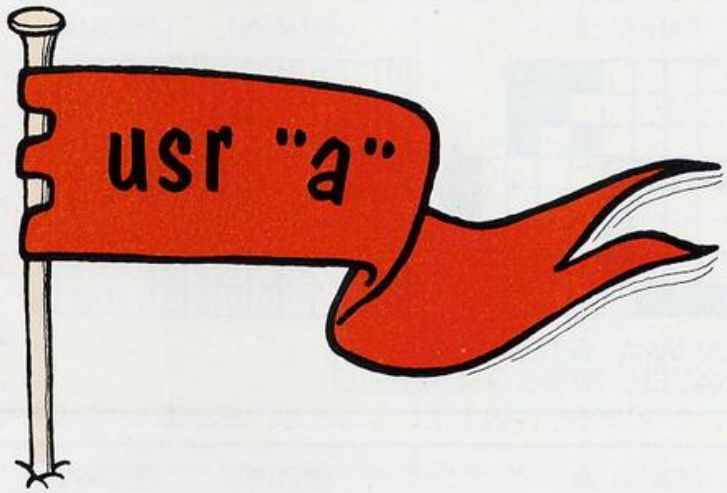
siguiente programa, que asigna a los gráficos de las teclas de la **M** y la **H**, los símbolos representativos de los sexos:

```
10 REM J.M.LOPEZ MARTINEZ - GENE
RACION DE CARACTERES
20 FOR I=0 TO 7: READ X: POKE USR
"M"+I,X: NEXT I
30 FOR I=0 TO 7: READ X: POKE USR
"H"+I,X: NEXT I
40 PRINT AT 11,12;"M=M, H=H"
50 DATA 7,3,5,120,136,136,136,112
60 DATA 56,68,68,68,56,16,56,16
```

En la línea 20 se ejecuta un bucle **FOR NEXT**, que tiene por objeto colocar en la posición **USR "M"** y las 7 siguientes, los valores de las 8 filas del carácter definido, leídas desde la **DATA** de la línea 50, en la variable **X**.
De forma análoga, en la línea 30 se hace lo propio, esta vez a partir de la posición **USR "H"**, y tomando información de la **DATA** de la línea 60.
Por último, en la línea 40 se muestra el resultado obtenido.
Los métodos para asignar la forma a un gráfico definido son sin duda variadísimas, y buena prueba de ello la encontraremos en la sección de PROGRAMA, donde la definición de los caracteres de los distintos programas ha sido llevada a cabo de maneras muy diferentes.

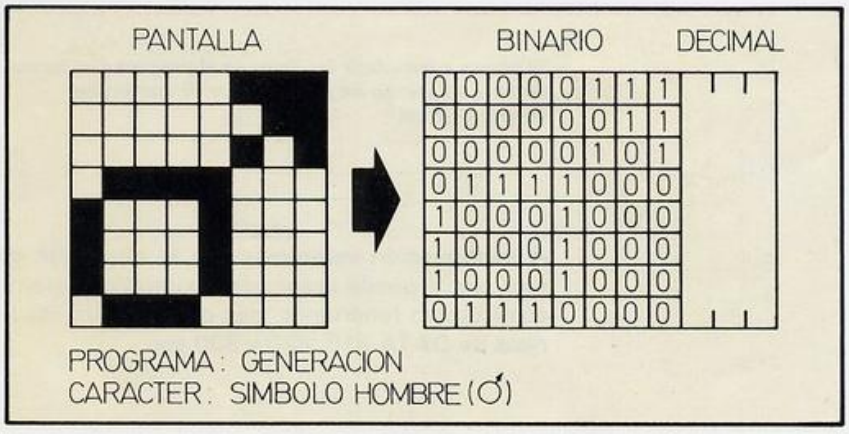
GENERACION DE SPRITES

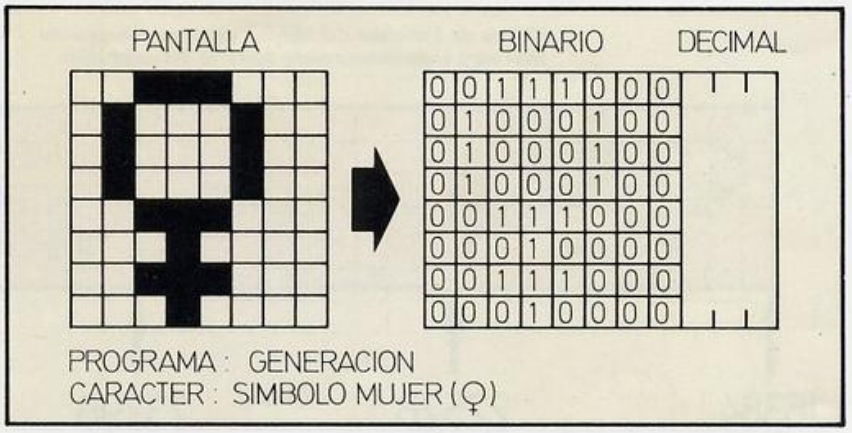
Se conocen como *sprites* los móviles empleados en los programas, cuya forma puede ser redefinida por el usuario. Estos pueden componerse de un sólo carácter pero, generalmente, lo están por un grupo de ellos que se desplazan conjuntamente.
Esto es posible gracias a que las matrices de 8 × 8, que componen los caracteres gráficos, no dejan ninguna separación ni vertical ni horizontal; de este modo, pueden apilarse caracteres definidos unos encima de otros, o de forma lateral, para conseguir un *sprite* de mayor tamaño.
Una muestra de ello, la tenemos en el programa TRUENO AZUL, que ocupa el próximo capítulo de la sección de PROGRAMA. En él, un helicóptero altamente sofisticado es el protagonista de la aventura.

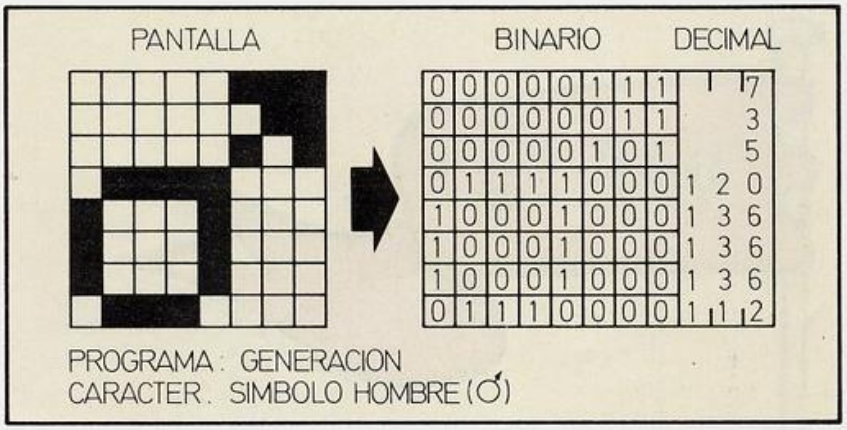


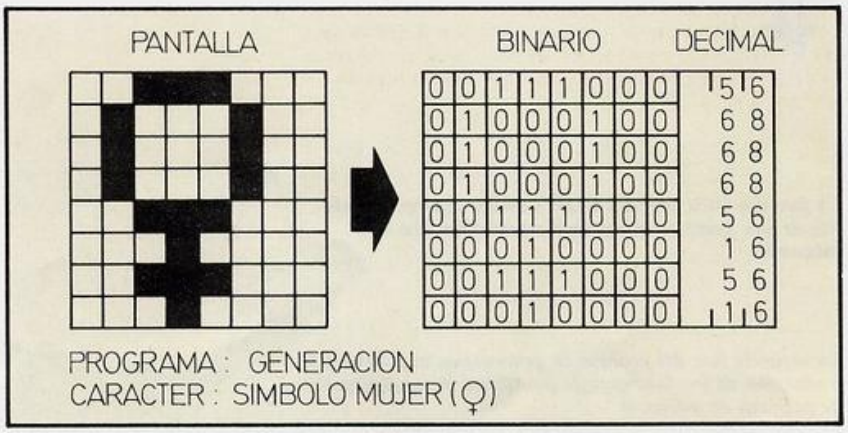
La función **USR** *seguida de un carácter entrecomillado (desde «a» hasta «u») equivale a una dirección de memoria.*

La segunda fase del proceso de generación, concierne a la traducción de los formatos de pantalla a la configuración de patterns *en memoria.*









Si vamos a introducir los datos en el programa en forma decimal, es preciso un paso anterior de conversión binario-decimal.

A continuación veremos como la alteración de este *sprite* puede ocasionar resultados bastante cómicos: no tendremos más que sustituir las líneas de **DATA** 810, 820 y 830 por:

```
810 DATA 0,35,112,255,255,127,48,3
820 DATA 0,255,129,231,255,255,129,255
830 DATA 0,192,1,253,255,253,1,192
```

Apreciaremos al ejecutar nuevamente el programa como el hilicóptero fruto de la ciencia ficción, se ha convertido en un auténtico Foker superviviente de la Primera Guerra Mundial. ¡Las maravillas de la microinformática!

La tecnología de la generación de un *sprite* suele ser algo más compleja, no tanto por lo que es su inclusión en el programa, que resulta idéntica a la de cualquier gráfico definido, sino por el hecho de que cuando se utilizan varios caracteres solidariamente, hemos de tener mucho cuidado para que coincidan perfectamente, formando un todo único.

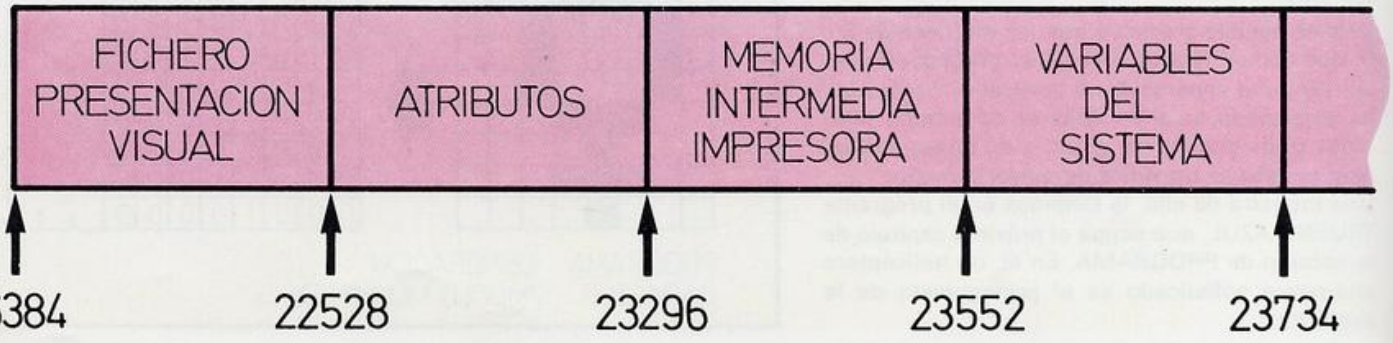
Dado este inconveniente, los programas comunes para generación de gráficos no nos son de gran utilidad en esta aplicación concreta. Así pues, hemos de recurrir, o bien a los programas generadores de pantallas, o bien al método tradicional del lápiz y el papel.

PEEK

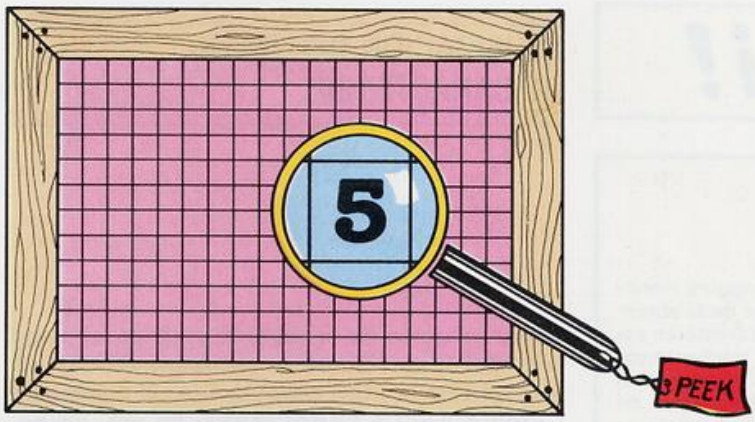
El área de Variables del Sistema contiene información vital para el funcionamiento correcto del ordenador.

Antes de comenzar la explicación del sistema de reubicación de caracteres, conviene que aprendamos una nueva función BASIC: **PEEK**.

Esta realiza la función inversa a la sentencia **POKE**, es decir, averigua el byte contenido en determinada posición de memoria. Así pues, esta función precisa de un sólo argumento numérico, que deberá estar comprendido entre 0 y 65535. Como en el caso de **POKE**, si el argumento de la función **PEEK** es negativo, recibimos el mensaje

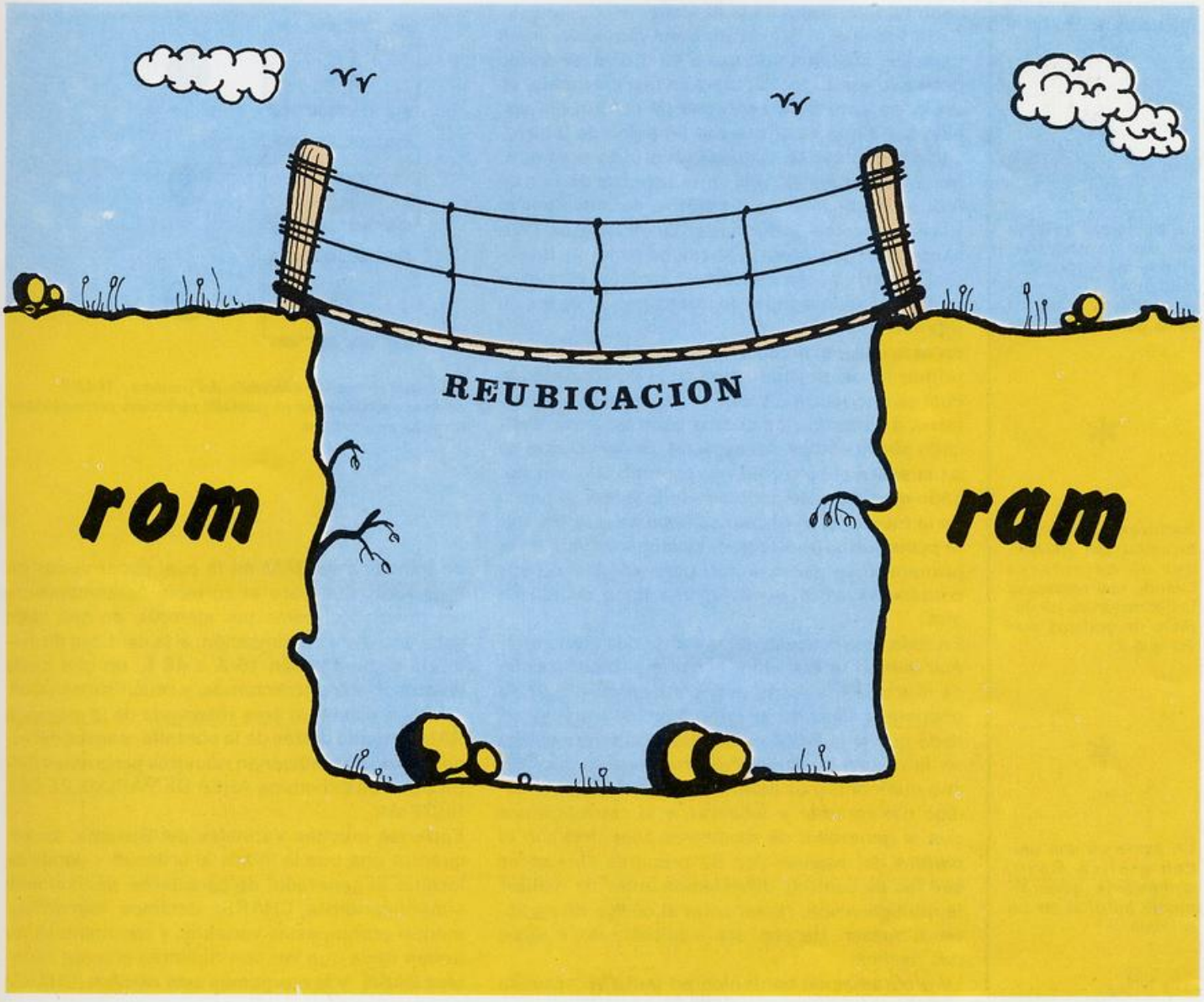


La función **PEEK** *investiga el contenido de una dirección de memoria.*



B Integer out of range, y utilizando direcciones no enteras, se efectúa un redondeo hacia el entero más próximo: los argumentos iguales o superiores a 0.5, al inmediatamente superior, y los que no alcanzan dicha fracción, al inferior.
Lógicamente, el resultado de la función **PEEK** será un número entero comprendido entre 0 y 255. Como es natural, puesto que el contenido de la ROM es inalterable, el **PEEK** de cualquier dirección por debajo de 16384 dará siempre un mismo resultado. Por el contrario, las investigaciones dirigidas a la RAM (por encima de 16383) pueden darnos en cada caso diferentes resultados.

La reubicación del generador de caracteres a RAM es un puente tendido a la creación de un número de caracteres definidos, sólo limitado por la cantidad de memoria de que dispongamos.



¡!

Los principales inconvenientes de la obtención de caracteres especiales mediante caracteres predefinidos son los siguientes: no es posible obtener gran parte de ellos, y si van a ser utilizados frecuentemente en el programa prolongan considerablemente la longitud de éste, además de ser nétamente inferiores en calidad.

*

La principal ventaja del uso de combinaciones de caracteres predefinidos es que no es necesario el proceso de generación.

*

Recurriremos a la reubicación del generador de caracteres cuando sea necesaria la definición de un número de gráficos mayor que 21.

*

Un *sprite* es una unidad gráfica móvil, compuesta generalmente por más de un carácter.

REUBICACION DEL JUEGO DE CARACTERES

Como ya vimos en el capítulo anterior, el juego principal de caracteres, se encuentra definido en una zona de la ROM conocida bajo el nombre de GENERADOR DE CARACTERES. En ella, los *patterns* (forma de los caracteres), se almacenan de la misma manera que lo hacen los gráficos definidos en el RAM; es decir, en conjuntos de ocho bytes, cada uno de los cuales equivale a una línea del carácter, y por tanto, con cada bit representando un punto de la pantalla.

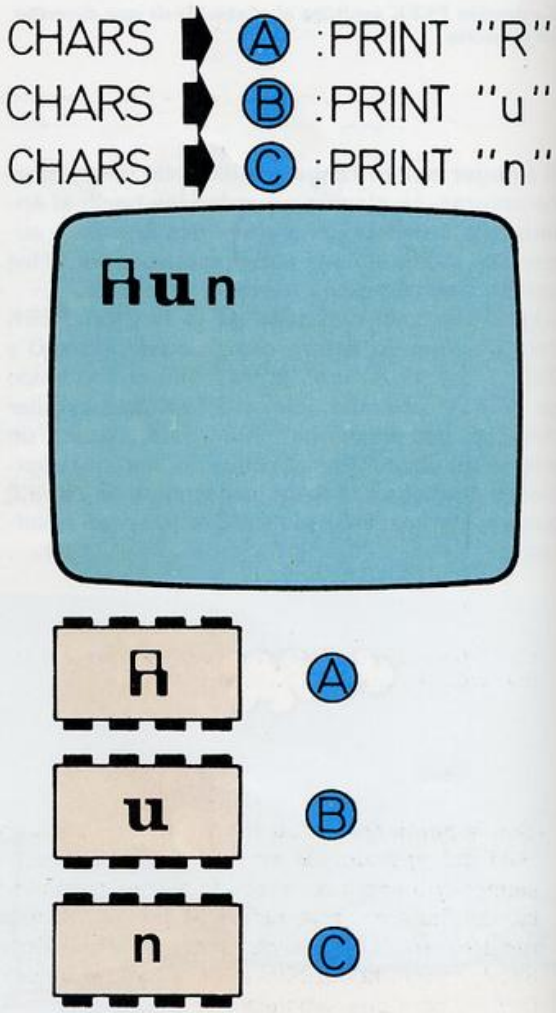
Cada vez que el Spectrum tiene que escribir un carácter, averigua en base a su código de donde debe extraer su forma: si no es representable, es decir, se trata de un carácter de control por debajo del 32, lo sustituye por un signo de interrogación. En caso de que sea un gráfico predefinido, lo construye en una zona especial de la memoria. Tratándose de un gráfico definible por el usuario, localiza su forma actual en la RAM. Pero lo que más nos interesa ahora, es su forma de actuar cuando desea escribir un carácter común, o incluso el mismo signo de interrogación de los no representables.

En este caso, el procedimiento es el siguiente: en primer lugar, averigua cual es la dirección en la cual se encuentra ubicado el generador de caracteres, a continuación cuenta ocho bytes multiplicado por el código del carácter, de forma que se sitúa sobre el *pattern* a representar. Una vez llegado a este punto, todo se limita a leer el punto de la memoria en el cual se encuentra, y los siete posteriores, puesto que inicialmente está en la primera línea del carácter, para ir volcando sus contenidos en la pantalla, una línea debajo de otra.

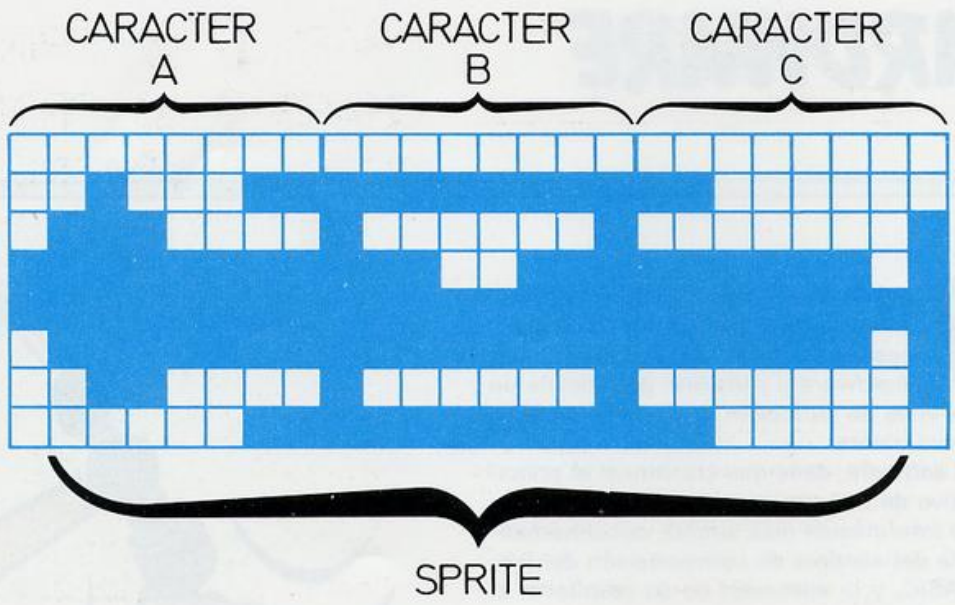
En todo este proceso, debe haber dos cosas que nos llamen la atención: por una parte, el hecho de que sea necesario averiguar en que lugar se encuentra ubicado el generador de caracteres, dado que si la ROM no se altera, siempre estará en la misma posición. Por otra parte, hemos dicho que contamos ocho multiplicando por el código del carácter a localizar, y si consideramos que el generador de caracteres comienza con el *pattern* del espacio (los 32 primeros caracteres son los de control), deberíamos antes de realizar la multiplicación, restar ocho al código de carácter a buscar. He aquí las explicaciones a estos dos hechos.

La programación contenida en la ROM, necesita de una zona de RAM en la cual poder volcar información vital para el correcto funcionamiento del ordenador, como por ejemplo, en que color debe escribir a continuación, si la cantidad de memoria dispinible son 16 K o 48 K, en que modo de cursor nos encontramos, y muchísimas otras. A tal fin, existe un área reservada de la memoria RAM, situada detrás de la pantalla, y antes del lugar donde se almacenan nuestros programas BASIC, que se denomina AREA DE VARIABLES DEL SISTEMA.

Entre las muchas Variables del Sistema, se encuentra una que le indica al ordenador donde se localiza el generador de caracteres, denominada simbólicamente CHARS; decimos «simbólicamente» porque estas variables y sus nombres no tienen nada que ver con nuestras propias variables BASIC, y le otorgamos este nombre (CHARS)



Mediante el manejo adecuado del puntero CHARS podemos simultanear en pantalla caracteres pertenecientes a varios generadores.

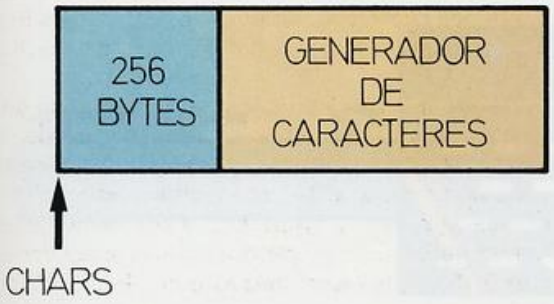


Un **sprite** *es una unidad gráfica móvil, compuesta generalmente por más de un carácter.*

simplemente porque es el que se le da en el manual del aparato. Al encender el ordenador, y siempre que no alteremos el valor de esta Variable del Sistema, esta señala al generador de caracteres en ROM. No obstante, su situación en RAM, nos da la posibilidad de alterar el valor de CHARS para que señale a otro punto.

Este otro punto, lógicamente, es una zona de la RAM, donde nosotros podemos haber depositado los *patterns* de todo un juego de caracteres; a esto se le denomina REUBICACION DEL GENERADOR DE CARACTERES. Naturalmente, si CHARS apunta a un lugar no apunta a otro, es decir, no podemos disponer simultáneamente del juego de caracteres de ROM y del nuestro propio de RAM,

La Variable del Sistema CHARS siempre señala 256 bytes antes del comienzo del generador de caracteres.



aunque dentro de unas líneas tendremos la oportunidad de ver un sistema para simular o. Pero antes queda por resolver una cuestión pendiente. El hecho de que no restemos 32 al código de carácter antes de iniciar la búsqueda del carácter (multiplicar por ocho su código) se debe a que para evitarnos este problema, CHARS no apunta exactamente al comienzo del generador de caracteres, sino concretamente 256 bytes antes: 8 bytes por 32 caracteres suponen 256 bytes, a sabiendas de que el sistema de 256 bytes (32 caracteres) correspondientes a los códigos de control, y que ya hemos comentado con anterioridad.

En cuanto a la forma de simultanear, aunque sólo sea de manera aparente, varios juegos de caracteres (por ejemplo el de la ROM y uno o varios en RAM), se basa en el hecho de que una vez escrita la información en la pantalla, ésta no se altera. Así pues, nosotros podemos escribir en el juego de caracteres que queramos, sólo con modificar previamente CHARS para que apunte al generador correspondiente en cada caso.

Para terminar veámos el sistema de alteración de CHARS. Esta Variable del Sistema se encuentra en las direcciones 23606 y 23607. Por tanto, siendo X la nueva situación deseada para el generador, en primer lugar le restamos 256 (**LET X=X-256**) y luego tan sólo es necesario efectuar un **POKE** en 23607 de la parte entera de X partido por 256 (**POKE 23607,INT (X/256)**, y a continuación, de su resto en 23606 (**POKE 23606,X-256*PEEK 23607**).

En general, podemos utilizar la siguiente línea BASIC, siempre y cuando recordemos asignar a **X** la dirección de comienzo del generador requerido antes de ejecutarla (**LET X=<dirección del generador>**): **LET X=X-256: POKE 23607, INT(X/256): POKE 23606, X-256* PEEK 23607.**

¡!

Gracias a la generación de caracteres se pueden obtener resultados de gran calidad.

*

La Variable del Sistema que indica la situación del generador de caracteres se denomina CHARS.

*

CHARS señala siempre 256 bytes antes de la dirección de comienzo del generador de caracteres.

*

Un *pattern* es el formato en bytes que un carácter tiene en la memoria.

*

El área de Variables del Sistema contiene información vital para el correcto funcionamiento del ordenador.

HARDWARE

¡!

Las puertas lógicas fundamentales realizan las funciones lógicas booleanas: AND, OR, XOR y NOT.

*

Existen dos puertas utilizadas bastante frecuentemente: NAND y NOR. La puerta NAND equivale a una puerta AND cuya salida es negada. La puerta NOR, también conocida como puerta NI, equivale a una puerta OR cuya salida es negada.

HASTA ahora hemos tenido la oportunidad de estudiar los dos componentes esenciales de un ordenador: *hardware* y *software*, pero desde un punto de vista de su misión, y no de su sistema de funcionamiento.

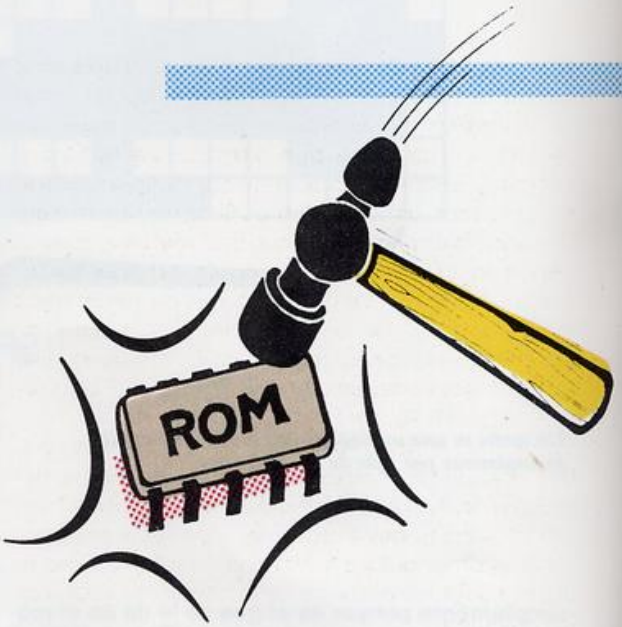
Sobre el *software*, dado que constituye el principal objetivo de nuestro aprendizaje, hemos recibido una información más amplia: concretamente, acerca del sistema de interpretación del lenguaje BASIC, y la ejecución de su resultado: el código máquina.

Ahora bien, el *hardware* fue también estudiado desde la óptica de su misión, descuidando lo referente a su organización interna para conseguir un funcionamiento tan sumamente eficaz. De esto nos ocuparemos en el presente capítulo

Como ya sabemos, el *hardware* es el conjunto de elementos físicos que integran nuestro ordenador. Todos ellos tienen por fin último el tratamiento lógico de una información. Es aquí donde se haya la propia esencia del sistema: en la información.

Lo que nosotros entendemos por información, no se parece prácticamente en nada a lo que el ordenador entiende como tal. Realmente, la información según la vamos a tratar ahora, desde el punto de vista de nuestro Spectrum, no es más que un simple estado de la corriente eléctrica. Dado que el *hardware* no nos sirve para nada sin la información memorizada en su *firmware*, veamos primeramente como funcionan las memorias. Reduciendo enormemente su complejidad técnica, la memoria no es más que un conjunto de circuitos interrelacionados, basados en un elemento capaz de retener un cierto nivel de corriente eléctrica, que nosotros asociamos a un estado lógico (1 ó 0).

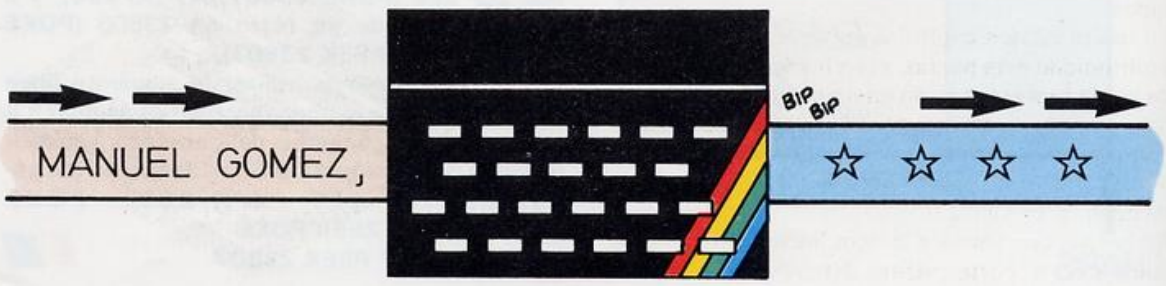
Cada uno de estos elementos es lo que para nosotros constituye un bit, y la relación de ocho de ellos, un byte. Por último, la cantidad total de ellos que integran estos circuitos, ya sea en uno o en varios *chips*, conforman el conjunto de la memoria del ordenador.



El contenido de una memoria ROM es inalterable por medio de la programación. Mucho nos tememos que para su destrucción haya que recurrir al martillo.

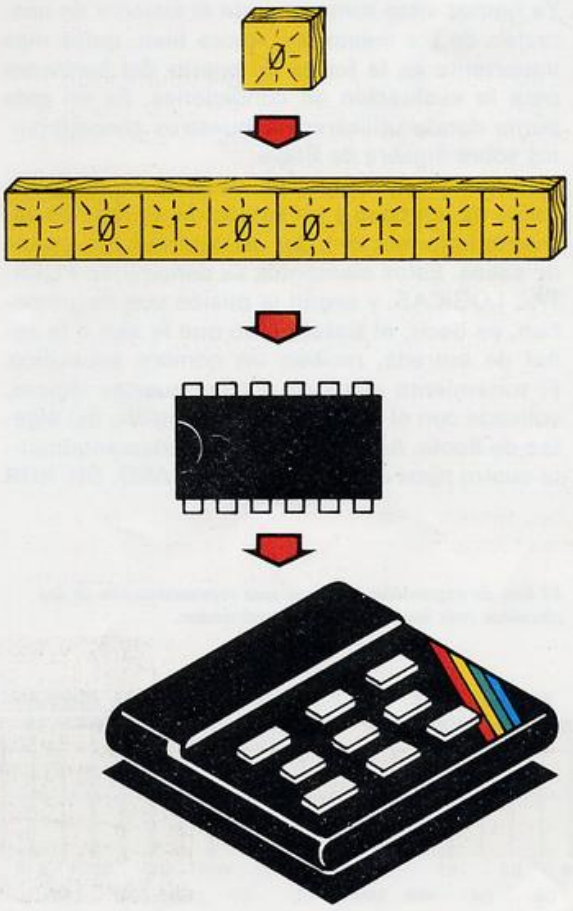
Realmente, existe una distinción entre las memorias RAM y ROM, dado que las primeras pueden alterar el nivel de corriente de estos elementos básicos, mientras que el contenido de una memoria ROM es físicamente inalterable. De ahí que, ni aún intencionadamente, consigamos dañar permanentemente nuestro aparato utilizando la programación.

El **hardware** *trata la información como simples impulsos eléctricos.*



Ni tan siquiera desde el código máquina podemos alterar el contenido de la ROM, puesto que esto es físicamente imposible, y por tanto, cualquier problema, por grave que parezca, se reduce a apagar y encender nuevamente el aparato. Así que ya sabemos; si queremos destruir nuestro ordenador, habremos de recurrir al martillo, porque nunca lo conseguiremos con la programación.
Estos dos estados de corriente, que denominamos 0 y 1, circulan por la circuitería de nuestro aparato de un elemento a otro: de la memoria a la C.P.U. y viceversa, entre los *ports* y la C.P.U. através de la U.L.A., etc...
En definitiva, las pistas del circuito impreso sobre el cual se encuentran los componentes del ordenador, son como las calles de una ciudad extraordinariamente transitada, recorridas por centenares de impulsos eléctricos, con los cuales la inteligencia humana ha sido capaz de codificar información.
Algunas de estas señales eléctricas, tienen el sentido de información, y circulan através de los buses, tanto de datos como direcciones, en los cuales se expresan, respectivamente, el dato a transportar, y el lugar del ordenador al cual va destinado. De nuevo, pese a esta importante misión que le atribuimos a los buses, estos no son más que un conjunto de pistas por las cuales circula la corriente eléctrica.
Por otros circuitos, llamémosles menos especializados, o de segundo orden, circulan señales con otros sentidos, como es propiamente la corriente eléctrica de alimentación, que le da a los componentes la energía necesaria para su funcionamiento.
Un ejemplo muy claro de este hecho, es la transmisión de datos a la grabadora de casetes. En ella vemos claramente, «oímos» este aspecto. Así por ejemplo, esos unos y ceros, que dentro del ordenador adquirirán el sentido de un apasionante juego, o un complicado programa de gestión comercial, no son más que unos estridentes sonidos.
En los próximos capítulos estudiaremos más en profundidad este hecho, y tendremos oportunidad de ver la representación en un osciloscopio de estas señales. De momento, nos bastará con saber, que cuando se emite o se recibe desde la grabadora, un cero o un uno no son más que señales de distinta frecuencia.
De hecho, con vistas a la comunicación con otras máquinas, la parte trasera del ordenador deja al descubierto una parte de la tarjeta de circuito impreso, denominada BUS DE EXPANSION. En el se encuentran representados los principales circuitos, con sus correspondientes señales, para que cualquier aparato que las comprenda, pueda leer o enviar señales (información) a través de los mismos, pasando a formar parte, como periférico, del sistema ordenador.
Hasta el momento, el *hardware* se nos ha revelado como un conjunto de elementos bastante estúpido, y nos parece increible que el simple hecho de enviar corriente en dos estados a través de unos circuitos, que no dejan de ser meros cables, pueda llegar a dar resultados que en ocasiones pueden parecer inteligentes.

La asociación de los componentes ' cos de la memoria, capaces de memorizar un estado de corriente (bit), conducen a la construcción de u conjunto de varios Kbytes.



¡!

En los esquemas lógicos, las puertas NAND se representan como puertas AND con un círculo antes de su salida, y las NOR con el círculo antes de la salida de una puerta OR.

*

La lógica triestado admite tres estados diferentes de corriente: dos de ellos se asignan a los niveles lógicos (1 y 0), y el tercero a la ausencia de éstos.

*

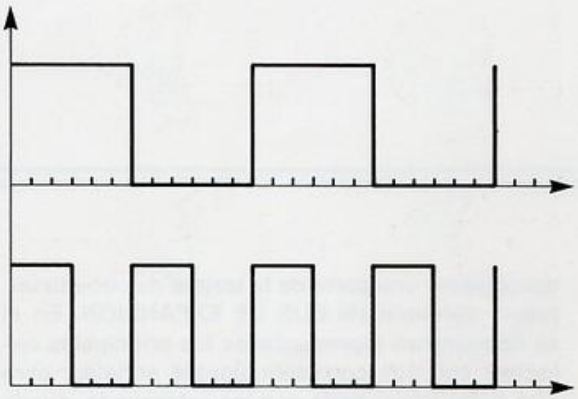
La cadencia de operación del ordenador viene marcada por el reloj.

¡!

Existe una imposibilidad física para alterar el contenido de una memoria ROM mediante la programación; por tanto, no nos debe importar experimentar con los **POKEs**, ni desde el BASIC, ni desde el propio código máquina.

*

Las puertas lógicas son elementos del *hardware* capaces de dar resultados distintos en base a sus entradas.



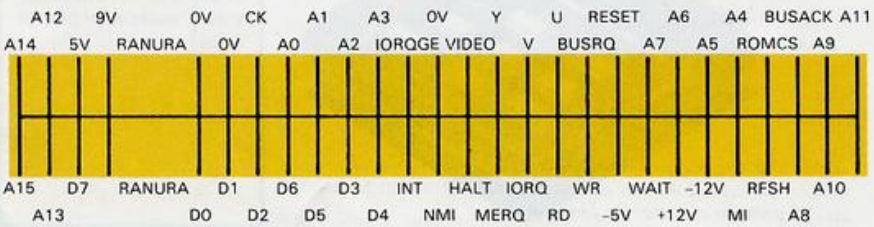
Las informaciones almacenadas en un casete no son más que ondas de diferente frecuencia.

La «inteligencia» del ordenador reside fundamentalmente en dos puntos: su capacidad de memorizar estados de corriente (información), y su capacidad para alterar estos estados (información) de acuerdo con determinadas condiciones.
Ya hemos visto someramente el sistema de operación de las memorias. Ahora bien, quizá más importante es la forma de operar del *hardware* para la evaluación de condiciones. Es en este punto donde utilizaremos nuestros conocimientos sobre álgebra de Boole.
Existen unos elementos del *hardware* que son capaces de, en base a un estado de entrada de corriente (información), dar un determinado estado de salida. Estos elementos se denominan PUERTAS LOGICAS, y según la misión que desempeñen, es decir, el tratamiento que le den a la señal de entrada, reciben un nombre específico. El tratamiento otorgado por las puertas lógicas, coincide con el de las funciones lógicas del álgebra de Boole. Así pues, existen fundamentalmente cuatro tipos de puertas lógicas: AND, OR, XOR y NOT (conocida generalmente como puerta INVERSORA).
En los esquemas de circuitería, cada una de estas puertas se haya representada mediante un símbolo, que podemos observar en las figuras adjuntas.
Estas puertas, son capaces de tratar un bit de información por cada entrada, generando una señal (bit) de salida. Lógicamente, podemos combinar ocho puertas de un mismo tipo para efectuar una función lógica sobre un byte completo.
Como vimos anteriormente, mediante el álgebra de Boole, podemos tratar matemáticamente problemas que aparentemente no tienen nada que ver con esta ciencia; como por ejemplo, si nuestro coche puede o no atravesar un paso a nivel. Gracias a ello, el ordenador puede gestionar la información de una manera aparentemente «inteligente», basando sus decisiones en el resultado de salida de las puertas lógicas.
Además de estas cuatro puertas lógicas fundamentales, se suelen emplear dos más: NAND y NOR. La primera equivale a la combinación de una puerta AND con un NOT a su salida, y la segunda, a una puerta OR a cuya salida se ha aplicado también una inversora.
En el último capítulo, estudiamos un importante teorema del álgebra de Boole: el Teorema de De Morgan. Su aplicación principal se da en la construcción de circuitos. Según él, podemos construir una función AND negando las entradas aplicadas a una puerta NOR, e igualmente, es posible construir una función OR mediante la negación de las entrada de una puerta NAND. Esto debe servirnos a modo puramente anecdótico, a no ser que nos dediquemos al diseño de algún circuito para conectar al Spectrum.

SINCRONISMO

Otro punto importante en el comportamiento del *hardware*, es el sincronismo de sus acciones. No debemos pensar que la nube de señales eléctricas (informaciones) que fluyen por las pistas, deambulan de una manera anárquica e incontrolada. Existe por el contrario un sistema de regulación, que sincroniza todos los tratamientos de información.
Este sistema que ya pudimos estudiar con anterioridad se denomina SINCRONISMO, y el núcleo fundamental del mismo, se encuentra en un oscilador (cristal de cuarzo), conocido bajo el nombre de RELOJ.

El bus de expansión contiene una representación de los circuitos más importantes del ordenador.

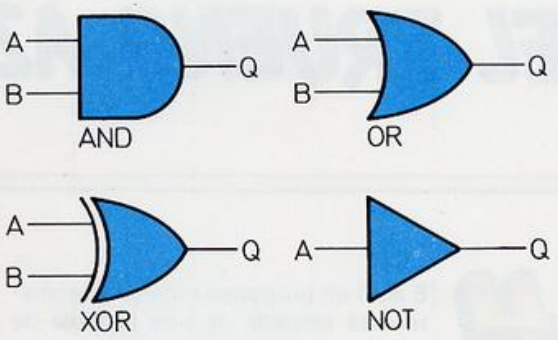


Este reloj envía una señal a intervalos iguales de tiempo, y esta señal es aguardada por el *hardware* para realizar todas las operaciones.

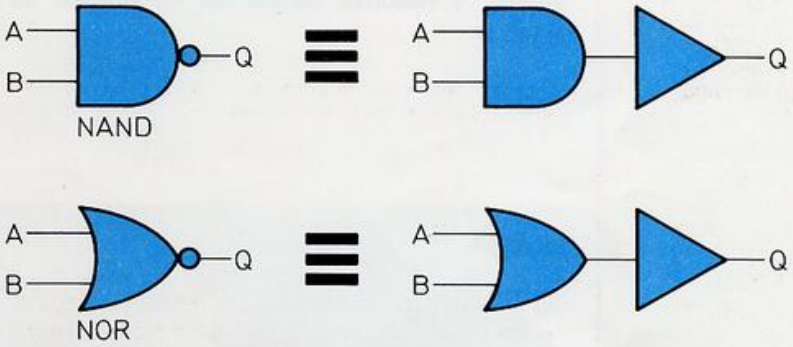
Supongamos que nos encontramos ante una puerta AND, a la cual llegan dos informaciones; sin embargo, el resultado no aparece a su salida ¿qué es lo que ocurre? si ya tiene los dos operandos, ¿a qué espera la puerta AND para procesar la información? Simplemente, a la llegada del impulso del reloj (señal de sincronismo), que le indique que puede trabajar.

En cierto modo, el reloj se puede comparar con los golpes de bombo, que en las antiguas galeras de esclavos marcaban la cadencia de las paladas. Por cada golpe, se remaba una palada; cuanto más seguidos eran los golpes, más rápido se remaba.

Análogamente, la velocidad de ejecución de un ordenador depende directamente de la cadencia que marque su reloj. En el caso concreto del Spectrum es de aproximadamente 3.5 Mhz (Megahercios), lo cual supone que se emite una señal de sincronismo unos tres millones y medio de veces por segundo.



Las puertas lógicas conciden con las funciones lógicas del álgebra de Boole.



Una puerta NAND equivale a la combinación de una puerta AND y una puerta NOT. Análogamente, una puerta NOR equivale a la combinación de una puerta OR y una puerta NOT.

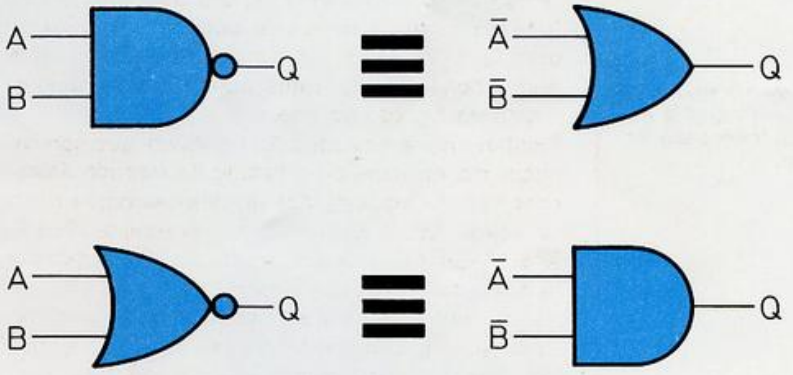
LOGICA TRIESTADO

Para finalizar diremos que, hasta ahora, hemos supuesto que existen dos estados de corriente, cuyo significado lógico es uno o cero; sin embargo, esto es cierto sólo en parte. Supongamos que somos un periférico reclamándole una información al Spectrum. Acudimos a su bus de expansión y comprobamos el estado lógico de determinado circuito.

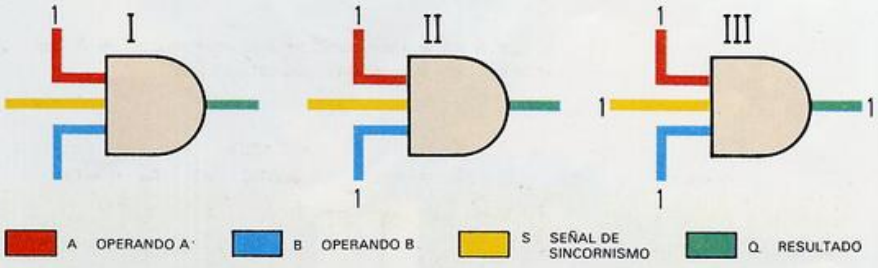
Si el nivel es uno, tendremos claro que esa es la información que nos desea transmitir. Ahora bien, si es un cero, ¿cómo saber si ese es el bit que quería transmitir el ordenador? pudiera ser que le hubiéramos sorprendido en un momento de «distracción», y simplemente no deseara transmitirnos ningún dato.

De esta circunstancia nace la necesidad de un tercer estado de corriente, esta vez sólo al nivel del *hardware*, de forma que además de los dos estados de corriente asociados a los bits (1 ó 8), exista un tercero que indique la falta de nivel lógico.

Este sistema, en el cual se utilizan tres estados: dos para representar los lógicos (1 ó 8) y el tercero para indicar la falta de estado lógico, se le denomina LOGICA TRIESTADO, y es la empleada por el *hardware* de nuestro ordenador.



En la figura aparece la representación del Teorema de De Morgan mediante símbolos lógicos para esquemas electrónicos.



Debido al sincronismo, una puerta lógica no sólo debe esperar la llegada de sus operandos, sino también la señal que le permite ejecutar su función.

EL TRUENO AZUL

HE aquí un programa cuyo protagonista está extraído de una película de ciencia ficción: el Trueno Azul *(Blue Thunder)*. Este es el avión más veloz, sofisticado, destructivo, poderoso, y temido del mundo, y nosotros somos los encargados de pilotarlo.

Nuestra misión es destruir lo más rápidamente posible las edificaciones que el enemigo ha construido en nuestro territorio, con el objetivo de utilizar el terreno como pista de aterrizaje para el aparato, y efectuar desde allí nuevas incursiones. A tal fin, el Trueno Azul ha sido equipado con bombas antimateria. Estas peligrosas armas son

capaces de destruir toda una columna de edificios, dejándola reducida a cenizas con solo tocarla. Debido a la gran velocidad que posee nuestro avión, el selector automático de disparo sólo nos permite efectuar tres lanzamientos por pasada, por lo cual no debemos desaprovechar ninguna bomba.

La casi totalidad de los microordenadores, cuentan en su biblioteca de programas con un juego similar al que hoy presentamos.

Pilotamos una nave que vuela cada vez más bajo, y debe destruir todos los bloques de edificios que se encuentran debajo de ella, para poder aterrizar sin chocar con ninguno.

La altura de las casas es aleatoria, y el número de bombas de que disponemos está limitado a tres por pasada, puesto que de no ser así, el juego sería muy sencillo. Para poder efectuar el bombardeo debemos pulsar **SPACE** (barra espaciadora en el Plus). Por cada bloque de edificio destruido, el Spectrum nos concede un punto. Si el arrasamiento ha sido total, los puntos obtenidos (puntos totales) son conservados y sumados a los que vayamos consiguiendo en la nueva pantalla.

Para hacer más complicado el juego, el avión que pilotamos desciende de dos en dos filas por cada pasada, debido a lo cual, no sólo debemos tener buena puntería para destruir los edificios, sino seleccionar cuidadosamente los blancos de nuestra acción; de no ser así, lo más seguro es que no duremos mucho tiempo sin toparnos con un edificio.

El programa posee además un marcador de records, en el cual queda almacenada la mayor puntuación lograda hasta el momento.

EL PROGRAMA

Como es habitual, la introducción del programa no plantea más problema que la presencia de los caracteres subrayados. Estos sutituyen a los gráficos definidos cuya tecla tiene por carácter más significativo el subrayado. Así por ejemplo, la A subrayada se introduce como el carácter gráfico de la tecla A.

El disparador de nuestro «Trueno Azul» es la tecla de **SPACE.**

¡!

El *Blue Thunder*, antes de posarse en el suelo, necesita destruir todos los edificios que aparecen en la pantalla.

*

El lanzamiento de las bombas antimateria se consigue pulsando **SPACE**.

*

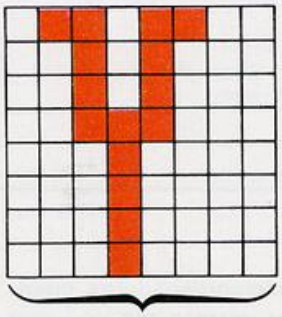
En el listado del programa, los caracteres subrayados corresponden a los gráficos definidos de usuario.

*

A la hora de grabar el programa, utilizaremos el siguiente comando: **SAVE "B.THUNDER"**. Si optásemos por la opción de autoejecución, teclearíamos la secuencia **SAVE "B.THUNDER" LINE 10.**

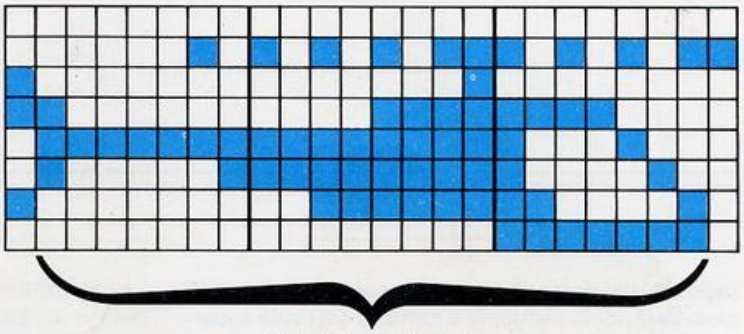
*

Para hacer más difícil el juego, nuestro avión desciende dos líneas por pasada.



BOMBA

En el gráfico definido de la C se encuentra la bomba antimateria.



TRUENO AZUL

El «Trueno Azul» viene definido por los tres primeros gráficos de usuario.

Recordemos que para introducir un carácter gráfico, hemos de pasar el cursor a modo G; o bien pulsando la tecla **GRAPH** en el Plus, o bien pulsando simultáneamente las teclas **CAPS SHIFT** y **9**, sistema éste válido tanto para el ZX Spectrum, como para el Plus. Una vez introducido el gráfico o gráficos precisos, debemos desconectar el modo **G**, para continuar con la introducción del resto del listado. Ello se consigue, ya sea repitiendo los pasos dados para la conexión, como pulsando simplemente la tecla **9** (válida para los dos modelos).

La estructura del programa es francamente clara, y ayudados por los **REM** que se incluyen en el mismo, se comprensión no estriba inconveniente alguno. Ahora bien, es destacable el sistema que hemos seguido para la generación de edificios, dado que gracias a un único grafico definido, podemos crear «rascacielos» de la altura que deseemos.

A la hora de la grabación del programa, sigamos el sistema habitual: **SAVE "B.THUNDER"**. Si deseamos que el programa se ejecute al finalizar su carga, debemos grabarlo mediante **SAVE "B.THUNDER" LINE 10**. ¡Buen aterrizaje!

```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 REM * BLUE THUNDER   1985 *
 50 REM ************************
 60 REM   INIC. VARIABLES
 70 LET REC=0
 80 LET REC1=0
 90 LET SW1=0
100 LET PT=0
110 LET PUN=0
120 REM   COMIENZO
130 GO SUB 790
140 GO SUB 260
150 GO TO 700
160 LET X=20-FILA
170 FOR Q=0 TO X
180 PRINT  INK 5;AT FILA,COL;" ABC"
190 PRINT  INK 6;AT FILA+Q,COL;"E"
200 BEEP .001,50
210 PRINT AT FILA+Q,COL;" "
220 PRINT  PAPER 4;AT 20,B;" "
230 IF ATTR (FILA+Q+1,COL)=58 THEN  LET PUN=PUN+1: P
RINT "PUNTOS ";PUN
240 NEXT Q
250 RETURN
260 REM   DIBUJO EDIFICIOS
270 LET F1=20
280 FOR W=1 TO 31
290 PRINT  PAPER 4;AT F1,W;" "
300 NEXT W
310 PRINT  PAPER 2;AT 21,18;"  RECORD=     ";AT 21,2
8;REC
320 REM
330 REM
340 FOR Z=4 TO 28
350 FOR J=INT (RND*7)+12 TO 19
360 PRINT  PAPER 0;AT J,Z;" "
370 PRINT ; INK 2; PAPER 7;AT J,Z;"D"
380 NEXT J
390 NEXT Z
400 LET CERO=0: LET CAB=0
410 REM  MOVIMIENTO HELICOPTERO
420 REM
430 FOR A=0 TO 20 STEP 2
440 LET SW=1
450 FOR B=0 TO 28
460 LET FILA=A: LET COL=B
470 PRINT  INK 5;;AT FILA,COL;" ABC"
480 BEEP .001,40:
490 REM COPMR.IMPACT & VICT.TOT
500 IF ATTR (FILA+CAB,COL+CERO+4)=58 THEN  LET CERO=
CERO-2: LET FILA=FILA: IF FILA=FILA THEN  GO TO 580
510 IF FILA=20 AND COL=28 THEN  LET SW1=1: PRINT  FL
ASH 1;AT 10,10;" BUEN PILOTO ": LET PT=PT+PUN: PRINT
 INK 9;AT 20,0;"PUNTOS TOTALES ";PT: GO SUB 990: GO T
O 710
520 IF SW>=4 THEN  GO TO 540
530 IF INKEY$=" " THEN  LET SW=SW+1: GO SUB 160
540 LET FILA=FILA: LET COL=COL
550 NEXT B
560 PRINT  PAPER 1;AT FILA,29;"  "
570 NEXT A
580 FOR D=FILA TO 20
590 REM   CHOQUE
600 REM
610 BEEP .001,50
620 PRINT  INK 2;AT FILA,COL+2;"AB"; FLASH 1;"D"
630 PRINT AT FILA,COL+1;" "
640 NEXT D
650 REM  TRASVASE DE PUNTOS
660 LET PT=PUN+PT
670 RETURN
680 REM   FIN DE JUEGO
690 REM
700 PRINT AT 5,8;"PUNTUACION= ";PT
710 IF SW1 THEN  GO TO 930
720 GO SUB 990
730 PRINT  PAPER 2; FLASH 1;AT 21,28;REC
740 PRINT AT 7,7; PAPER 6;" OTRA PARTIDA ? "
750 LET K$=INKEY$
760 IF K$="S" OR K$="s" THEN  BEEP .2,40: GO TO 90
770 IF K$="N" OR K$="n" THEN  GO TO 10000
780 GO TO 750
790 REM   GRAFICOS USUARIO
800 RESTORE 810
810 DATA 0,2,128,192,127,65,128,0
820 DATA 0,171,1,15,255,255,63,0
830 DATA 0,171,0,240,136,132,194,254
840 DATA 255,255,137,137,255,137,137,137
850 DATA 108,40,40,56,16,16,16,16
860 FOR K=USR "A" TO USR "E"+7
870 READ A
880 POKE K,A
890 NEXT K
900 PAPER 1: BORDER 1: INK 9
910 CLS
920 RETURN
930 FOR N=1 TO 100
940 BEEP .01,20: BEEP .01,40
950 NEXT N
960 LET SW1=0
970 GO TO 110
980 REM   COMPROBACION RECORD
990 IF PT>REC THEN  LET REC=PT: GO TO 1010
1000 RETURN
1010 PRINT AT 0,4; FLASH 1; PAPER 7;"ESTABLECIO UN NU
EVO RECORD";AT 1,10;"EN "; INK 7; PAPER 9;REC; PAPER
7; INK 9;" PUNTOS"
1020 RETURN
```