4

RUN

Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

¡¡MENUDO CAMBIO!!

Tráenos tu

SPECTRUM

y llévate un

SPECTRUM PLUS

Renuévate con INVESTRONICA.

Ahora INVESTRONICA te da la oportunidad de hacerte con el microordenador más moderno del mercado: EL SPECTRUM PLUS.
Sólo tendrás que entregarnos tu ZX SPECTRUM...
...lo demás será visto y no visto, el Spectrum Plus ya es tuyo.
Tener un ordenador Sinclair es la garantía de estar siempre a la última.

Apúntate a lo más nuevo.

El Spectrum Plus es lo más nuevo del mercado. Si tu Spectrum es estupendo; el Plus es fabuloso. Podrás disfrutar de un teclado profesional; 17 teclas más que el Spectrum, es decir 17 ventajas más... y por supuesto lo podrás utilizar con todos los programas y periféricos que ya tienes, puesto que **el SPECTRUM PLUS es totalmente compatible con todo el software y accesorios del spectrum.** Además INVESTRONICA, al realizar el cambio, **te da de nuevo 6 meses de garantía,** una nueva cassette de demostración y un libro de instrucciones a todo color.

No te lo pienses... cámbiate a lo último, tienes las de ganar.

Tenerlo, muy fácil

Manda tu ZX Spectrum (sin cables, ni fuente de alimentación) a tu Servicio Técnico Oficial (HISSA) más cercano, bien personalmente o por agencia de transportes (los gastos son por cuenta de INVESTRONICA) y en 48 horas ya podrás disfrutar de tu nuevo Spectrum Plus.
Sólo tienes que abonar (contra reembolso) 12.000 Pts. (*)

OPERACION CAMBIO

investronica

(*) 18.000 pts. si es de 16 K.

J. M. PUBLICIDAD

Dirígete a cualquiera de las delegaciones HISSA

C/. Aribau, n.º 80, Piso 5.º 1.º
Telfs. (93) 323 41 65 - 323 44 04
08036 BARCELONA

P.º de Ronda, n.º 82, 1.º E
Telf. (958) 26 15 94
18006 GRANADA

C/. San Sotero, n.º 3
Telfs. 754 31 97 - 754 32 34
28037 MADRID

C/. Avda. de la Libertad, n.º 6
bloque 1.º Entl. izq. D.
Telf. (968) 23 18 34
30009 MURCIA

C/. 19 de Julio, n.º 10 - 2.º local 3
Telf. (985) 21 88 95
33002 OVIEDO

C/. Hermanos del Río Rodriguez, n.º 7 bis
Tel: (954) 36 17 08
41009 SEVILLA

C/. Universidad n.º 4 - 2.º 1.º
Telf. (96) 352 48 82
46002 VALENCIA

C/. Travesia de Vigo, n.º 32, 1.º
Telf. (986) 37 78 87
6 VIGO

Avda. de Gasteiz, n.º 19 A - 1.º D
Telf. (954) 22 52 05
01008 VITORIA

C/. Atares, n.º 4 - 5.º D
Telf. (976) 22 47 09
50003 ZARAGOZA

COMPUTIQUE

Te da más

Y también SPECTRUM 48K por sólo 31.500

Al comprar tu Spectrum

te regalamos

Y además 6 programas

COMPUTIQUE

Abrimos sábados por la tarde

Embajadores, 90
28012 Madrid
Tfno. 2270980

PRINCIPIOS DE PROGRAMACION EN BASIC

PARA comenzar, debemos decir que el BASIC es un lenguaje de alto nivel, es decir, no excesivamente difícil de manejar y comprender por nosotros. Por medio de un conjunto no muy grande de instrucciones, podemos realizar programas que resuelven problemas muy variados e incluso complicados.

Hasta ahora, hemos trabajado en el modo directo o comando. Con este sistema hemos visto que las instrucciones que introducíamos a través de la línea 24 de la pantalla del ordenador, se ejecutaban de forma inmediata al pulsar la tecla **ENTER**.

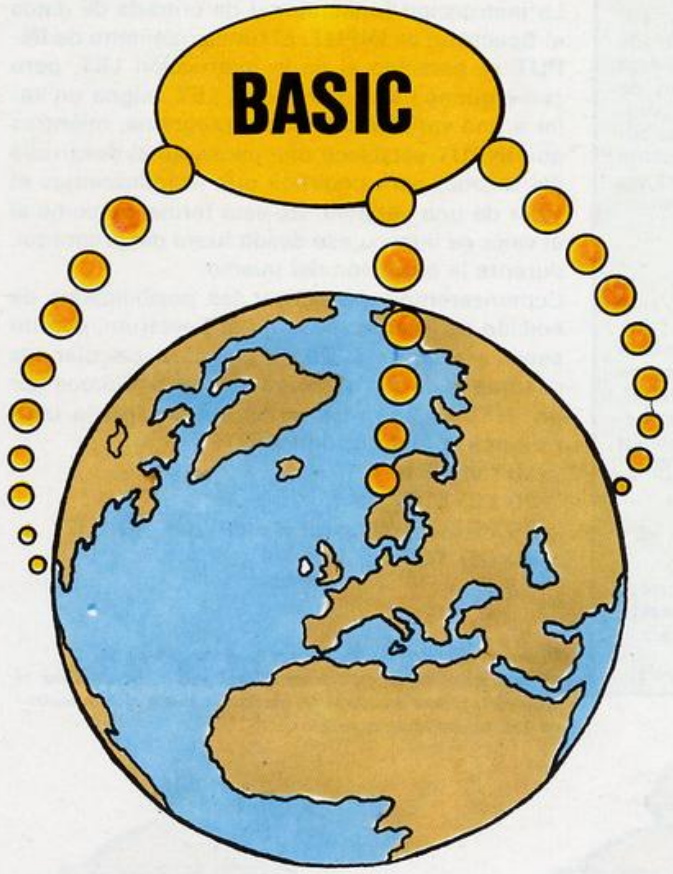
Para consultas rápidas sobre el valor o de una variable, o para averiguar el resultado de una operación matemática, el modo directo cumple perfectamente su cometido; pero cuando se trata de resolver problemas más complicados, que requieren de una entrada de datos, la realización de ciertos cálculos y una salida impresa, la utilización de este modo es imposible.

El modo PROGRAMA existe, precisamente, para facilitar la construcción de estructuras de programación más complejas, que permitan almacenar gran número de instrucciones, guardando un determinado orden de ejecución. Un programa BASIC no es más que una colección de instrucciones, numeradas para indicar al Spectrum el orden exacto en que deben ser ejecutadas.

Cuando anteponemos un número a lo que sería una instrucción en modo directo, el ordenador entiende que no la debe ejecutar de forma inmediata, sino almacenarla dentro del programa y colocarla en la posición que le corresponde.

Normalmente, las instrucciones de programa se numeran de 10 en 10. Esto se hace para permitir intercalar nuevas instrucciones entre las ya escritas. Cada vez que el Spectrum acepta una nueva línea para el programa, lo primero que se pregunta es si ese número de instrucción ya existe. De ser así, reemplaza la antigua línea por la nueva que le ha llegado con el mismo número. Si la instrucción no existía antes, la incluye dentro del programa, entre las líneas a que corresponde de acuerdo con su número.

El Spectrum nos permite, a través de su línea de edición (la última), la introducción, modificación y borrado de líneas de programa; contando para ello con la ayuda de la función **EDIT**, (**SHIFT** y **1** en el modelo antiguo), y las teclas de desplazamiento del cursor.



El BASIC es el lenguaje de programación, más ampliamente difundido en el mundo microinformático.

COMENZANDO A PROGRAMAR

Dado que el repertorio de palabras BASIC que manejamos es bastante limitado, vamos a ampliarlo algo en este capítulo. Un programa cons-

Los lenguajes de programación más utilizados en el Spectrum son el BASIC, el ensamblador y el código máquina. De entre ellos, el BASIC es el de más alto nivel, es decir, el de mayor facilidad de comprensión por los humanos.

¡!

El *syntax marker*, señala en la línea de edición el punto hasta el cual la sintaxis de la instrucción a introducir se considera correcta. Se representa mediante un carácter de interrogación en intermitencia.

*

La zona de la pantalla que se emplea para la introducción y modificación de las instrucciones, se denomina LINEA DE EDICION.

*

El comando fundamental para la modificación de programas es **EDIT**.

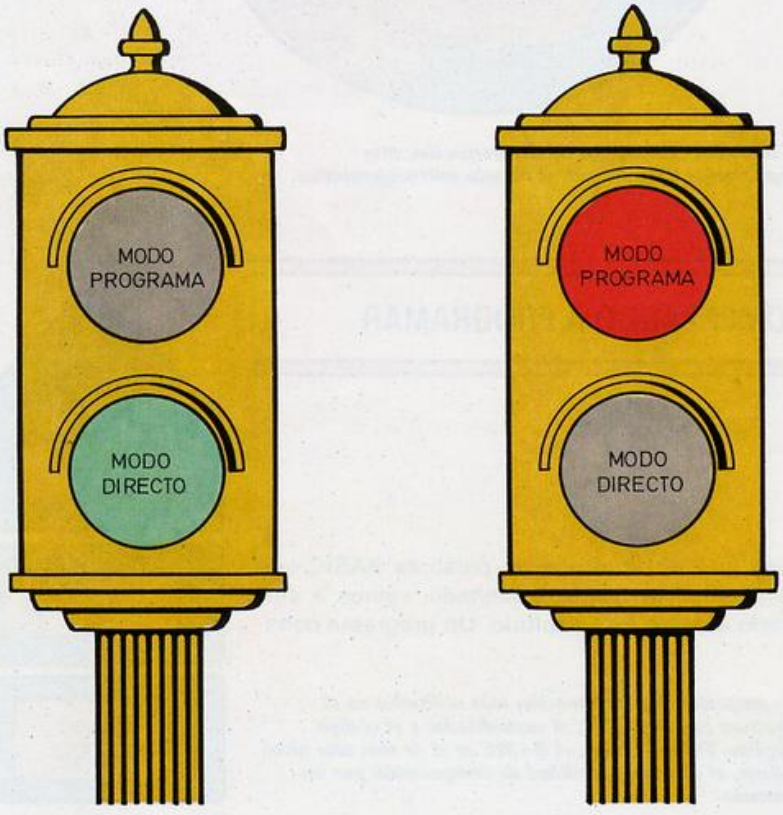
ta de tres partes fundamentales: entrada, cálculo y salida. Conocemos ya algunas palabras BASIC que nos permiten efectuar la salida de resultados **(CLS, PRINT)**, y operadores aritméticos con los que realizar los cálculos (+, -, *, /). Nos falta, por tanto, conocer alguna instrucción para introducir datos en el ordenador.

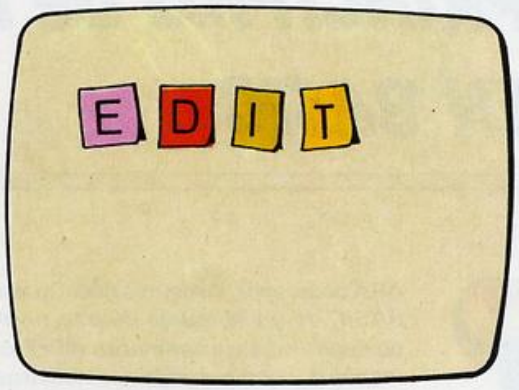
La instrucción fundamental de entrada de datos al Spectrum es **INPUT**. El funcionamiento de **INPUT** es parecido al de la instrucción **LET**, pero con algunas particularidades. **LET** asigna un valor a una variable dentro del programa, mientras que **INPUT** establece una pausa en el desarrollo del mismo, para pedirnos que introduzcamos el valor de una variable. De esta forma, es como si el valor se introdujese desde fuera del programa, durante la ejecución del mismo.

Comenzaremos a practicar las posibilidades de edición de instrucciones en el Spectrum, con un sencillo programa. En él, vamos a calcular los cuadrados de los números que introducimos por un **INPUT**; para mostrarlos en la pantalla utilizaremos la instrucción **PRINT**:

```
10 INPUT N
20 LET C=N*N
30 PRINT "El cuadrado de";N;" es:";C
40 GO TO 10
```

El modo directo, da vía libre a la ejecución de las instrucciones según entran en el ordenador. En cambio, el modo programa pospone su ejecución hasta el momento en que nosotros lo deseemos.





El comando* EDIT *se utiliza para editar cualquier línea de programa en el área de modificación (zona inferior de la pantalla).

Debemos recordar pulsar **ENTER** después de cada nueva línea de instrucción. Veremos que, según vamos introduciendo las líneas, éstas en vez de ejecutarse de forma inmediata, pasan a formar parte ordenada del programa y son transferidas a la parte superior de la pantalla. Hagamos una pequeña revisión de los conocimientos empleados en el programa.

Punto y coma (;) es un parámetro de la palabra clave **PRINT**, que actúa como separador entre variables y literales, indicando a un mismo tiempo que el siguiente dato impreso en la pantalla, debe situarse inmediatamente después del que ha precedido al punto y coma. **GO TO** es una instrucción que transfiere el control del programa a la línea que indica.

Una vez que hemos almacenado el programa, tenemos que ejecutarlo, o sea, hacerlo funcionar. Esto se logra empleando la palabra clave **RUN** en modo directo. **RUN** borra los contenidos de las variables que pudieran existir con anterioridad al inicio del programa, limpia la pantalla y comienza a ejecutar el programa a partir de su primera línea.

De no existir en el programa ninguna sentencia de transferencia de control **(GO TO)**, el programa pasaría por las líneas 10, 20, y por último 30, para detenerse a continuación por haber llegado al final del programa. Esto no sucede, porque el **GO TO** de la línea 40, indica al ordenador que el programa debe continuar a partir de la instrucción 10. Todos los programas están formados por una secuencia de instrucciones numeradas, de forma que al teclear **RUN** el ordenador ejecuta una a una las líneas que lo componen, por riguroso orden, hasta llegar a la última, donde se detiene.

La instrucción **GO TO** es una declaración imperativa. Esto quiere decir, que no está condicionada a ninguna otra circunstancia y que, cada vez que el programa pase por ese número de línea, debe continuar la ejecución a partir del nuevo número de instrucción que se le indica.

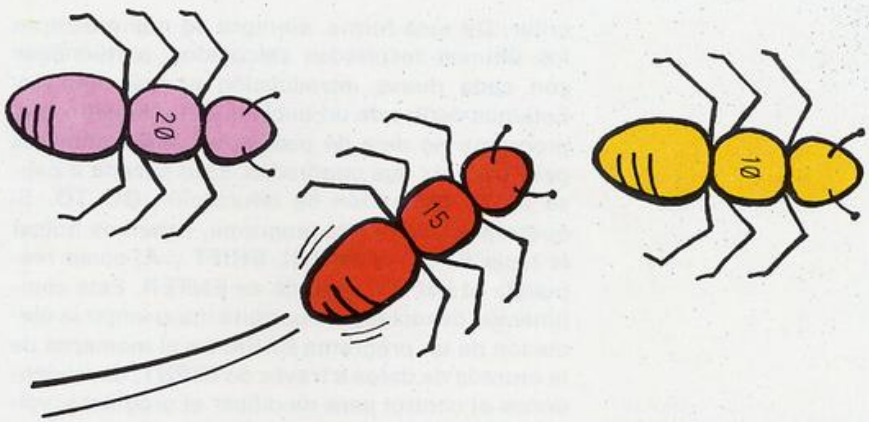
En nuestro ejemplo, al teclear **RUN** (palabra clave de la tecla **R**), el programa comienza a ejecutar la primera línea (10), demandándonos que introduzcamos un número. En la última línea de la pantalla, aparece el cursor en el modo **L**, para indicarnos que espera una secuencia de caracteres; debemos entonces teclear el número del que deseamos calcular el cuadrado, seguido de **ENTER**.

Hecho esto, el Spectrum pasa a interpretar el siguiente número de instrucción (20). Asigna a la variable numérica C, por medio de la instrucción **LET**, el resultado del producto del número introducido por nosotros en el **INPUT N**, por sí mismo; es decir, asigna a C, el valor N al cuadrado. Una vez realizado el cálculo, continua con la interpretación de la siguiente instrucción (30), que le indica como mostrar los resultados obtenidos en la pantalla. Como el programa no ha escrito aún ningún texto, la instrucción **PRINT** comienza a ejecutarse, a partir de la primera columna de la primera línea de la pantalla.

Lo primero que se escribe es el literal (la secuencia de caracteres entrecomillados) "El cuadrado de". Después imprime el valor de la variable N (inmediatamente a la derecha del literal por existir un ;), seguidamente el literal " es:", e inmediatamente el resultado almacenado en la variable C.

El **INPUT** *es como un buzón del programa, en el que el usuario puede depositar los datos con destino a éste. Dispone de dos entradas diferentes: una para variables numéricas y otra para variables de cadena.*

Observemos que, hasta aquí, toda la instrucción **PRINT** ha estado condicionada por el punto y coma (;), por lo que cada nuevo módulo de la misma se ha colocado a la derecha del anterior. Como al final de la instrucción (después de C), no



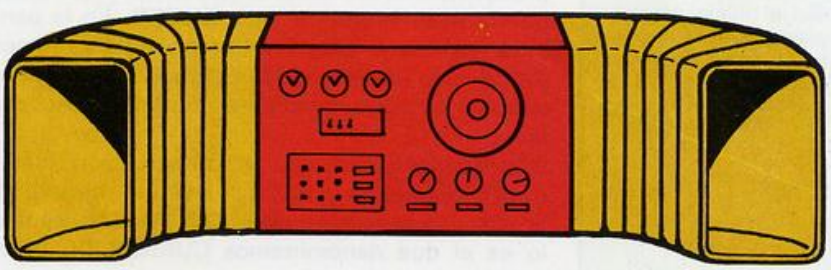
Las líneas de instrucción ocupan su sitio en el programa, no por su orden de introducción, sino según su número de instrucción.

aparece este signo de puntuación, el programa interpreta que las próximas instrucciones **PRINT** deben ejecutarse en la siguiente línea de la pantalla, por lo que se sitúa en ella cediendo el control de programa a la siguiente instrucción (40). De no existir esta línea de instrucción, el programa detendría aquí su ejecución, al haber rebasado el número de instrucción más alto en secuencia. Sin embargo, como hemos incluido una instrucción de bifurcación incondicional (**GO TO**), al llegar el programa a la línea 40, lo que hace es volver a comenzar con la interpretación de la línea 10.

De esta forma hemos creado un «bucle» de programa, que se repite una y otra vez, pidiéndonos un número, calculando su cuadrado e imprimiendo el resultado del cálculo en la pantalla. Por cada pasada por la instrucción 30 se genera una nueva línea de resultados.

Después de llenar con los resultados la primera pantalla, observamos que el ordenador efectua un *scroll* de la misma. Esta función automática, suprime la línea superior de la pantalla y desplaza las demás una línea hacia arriba, para dejar sitio en la última a la nueva línea que se va a es-

El proceso seguido por un programa, se desarrolla en tres fases: entrada de datos, cálculo en base a los mismos y salida de los datos calculados.



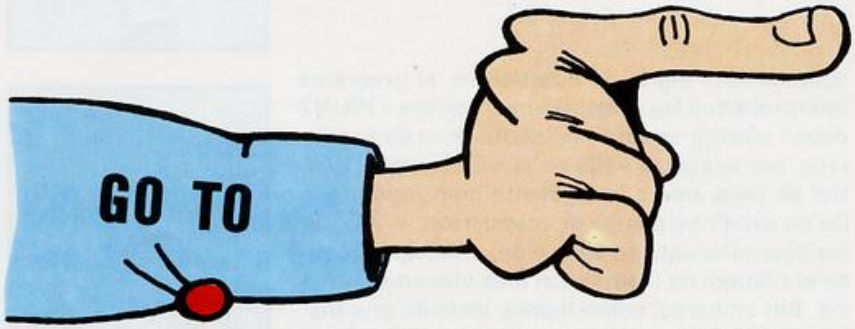
ENTRADA CALCULO SALIDA

Cuando una instrucción es escrita en la línea de edición sin ser precedida por un número, se ejecuta inmediatamente al pulsar **ENTER**. A este tipo de instrucciones se les denomina de MODO DIRECTO o MODO COMANDO.

*

El MODO PROGRAMA facilita la construcción de estructuras de programación más complejas, que permiten guardar un gran número de instrucciones siguiendo un determinado orden de ejecución.

cribir. De esta forma, siempre se nos muestran los últimos resultados calculados, perdiéndose con cada nueva introducción el más antiguo. Estamos dentro de un bucle que no tiene fin, y el programa no deja de pedirnos nuevos números para calcular sus cuadrados. Esto sucede a causa de la instrucción de bifurcación **GO TO**. Si queremos detener el programa, debemos pulsar la tecla **STOP** (**SYMBOL SHIFT** y **A**) como respuesta al **INPUT**, seguida de **ENTER**. Esta combinación de teclas, nos permite interrumpir la ejecución de un programa BASIC en el momento de la entrada de datos a través de **INPUT**, devolviéndonos el control para modificar el programa, volver a ejecutarlo, o suprimirlo de la memoria para comenzar a introducir otro nuevo.



La sentencia **GO TO** *obliga a continuar la ejecución del programa, a partir de determinado punto del mismo, que se señala en forma de número de línea a continuación de la sentencia.*

La instrucción fundamental de entrada de datos al Spectrum es **INPUT**. Esta sentencia establece una pausa en el desarrollo del programa, para que introduzcamos desde el teclado el valor de una variable determinada por el mismo. **RUN** borra las variables que pudieran existir con anterioridad al inicio del programa, limpia la pantalla y comienza a ejecutarlo a partir de su primera línea.

MODIFICANDO NUESTRO PRIMER PROGRAMA BASIC

Vamos ahora a detener el programa para incluir en él algunas modificaciones. Lo primero que hacemos, después de abandonar el **INPUT** por el procedimiento explicado en el párrafo anterior, es pulsar nuevamente la tecla **ENTER**. En la pantalla aparece el listado del programa que hemos introducido (la relación de instrucciones ordenadas con su correspondiente número de secuencia).

Observamos ahora, que en la instrucción número 40, aparece el símbolo > entre el número de la misma y la palabra clave **GO TO**. Este símbolo es el que denominamos CURSOR DE PROGRAMA.

Este cursor de programa apunta siempre al último número de instrucción introducido, modificado o listado. En nuestro caso, está presente en la línea 40 porque ha sido la última que hemos introducido.

Podemos probar ahora a desplazar este cursor por medio de **SHIFT** y las teclas **6** y **7** (flechas verticales en el Plus). Vemos que por cada pulsación de las teclas **SHIFT** y **7** (flecha hacia arriba), el cursor se desplaza una línea hacia arriba, hasta llegar al número más bajo (10). Del mismo modo, empleando ahora las teclas SHIFT y **6** (flecha hacia abajo), el efecto es el contrario, hasta llegar nuevamente a la línea 40.

Nos vamos a situar ahora en la línea 10, para introducir una pequeña modificación. Queremos que, además de que el ordenador nos pida un número, nos indique con un texto la palabra "Número:", aclarando qué tipo de dato se intenta recoger en el **INPUT**. Para poder hacer esto, basta con colocar un texto entre comillas entre la palabra clave **INPUT** y la variable N.

Hasta ahora la instrucción tenía el aspecto:

10 INPUT N

Para situarnos en la línea 10, vamos a emplear los cursores de programas y a teclear **SHIFT** y **1** (**EDIT**). Ahora podemos ver que en la última lí-

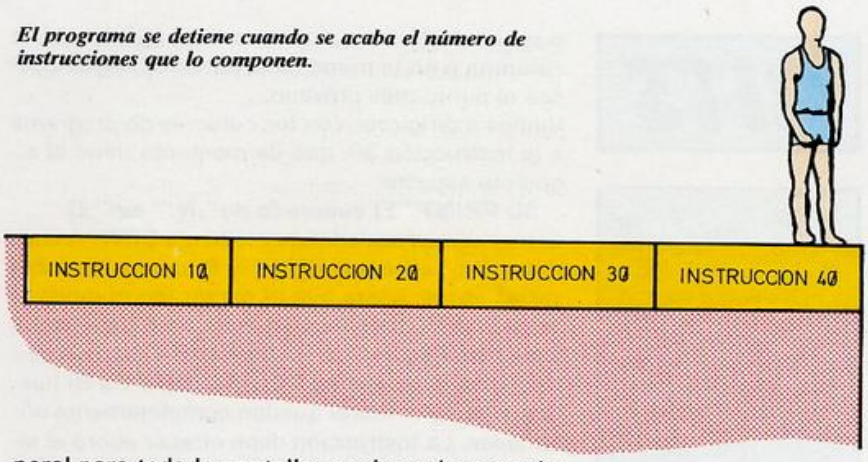
El comando **RUN** *se utiliza para ejecutar las instrucciones, que han sido almacenadas en forma de programa.*

nea de la pantalla se encuentra una copia de la instrucción número 10, lista para ser modificada, con el cursor en el modo **L** inmediatamente situado entre el número de instrucción y la primera palabra clave BASIC.

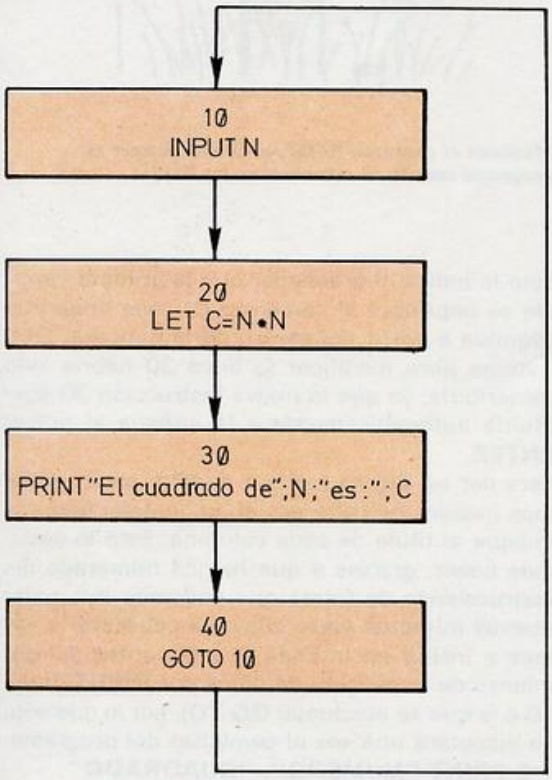
Esta vez, debemos emplear las teclas de desplazamiento horizontal del cursor **SHIFT 5** y **8** (**flechas horizontales**), para colocarnos en el lugar adecuado e introducir la modificación. Pulsamos **SHIFT** y **8** para colocarnos entre **INPUT** y **N**, y tecleamos **"Numero:"**;. Según hemos ido introduciendo el literal, la **N** que quedaba a su derecha, se ha ido desplazando para dejar sitio a los caracteres recién llegados. Seguidamente, pulsamos **ENTER** para validar la instrucción (no importa en qué punto se encuentre el cursor al pulsar **ENTER**), y podemos ver como el nuevo contenido de la instrucción 10 ha reemplazado al anterior, desapareciendo la línea de edición.

Podemos ejecutar nuevamente el programa, por medio de la palabra clave **RUN**, para ver los resultados. Efectivamente, el programa funciona de la misma manera que antes, pero pidiéndonos los números a introducir con el texto deseado. Para interrumpir la ejecución debemos emplear el sistema descrito anteriormente, pulsando **ENTER** una vez más si queremos listar el programa. Vamos a continuar ahora con las modificaciones, para conseguir que se imprima una cabecera general para toda la pantalla, en dos columnas denominadas "NUMERO" Y "CUADRADO". Pero para poder hacer ésto, necesitamos saber como escribir columnas de números de forma fácil. Del mismo modo que el punto y coma (;) le indica a **PRINT**, que lo próximo que escriba debe situarse inmediatamente a continuación, la coma (,) tiene el efecto de producir un salto desde la columna 0 a la 16 en la pantalla, es decir, de la primera a la segunda mitad de la misma, ya que la pantalla del Spectrum tiene 32 columnas. De este modo, tenemos solucionado el problema. Así pues, podemos decir que tanto el caracter punto y coma (;), como la coma (,), actúan como separadores de literales y variables, pero mientras que el primero indica que los resultados de **PRINT** se deben escribir seguidos, el segundo los

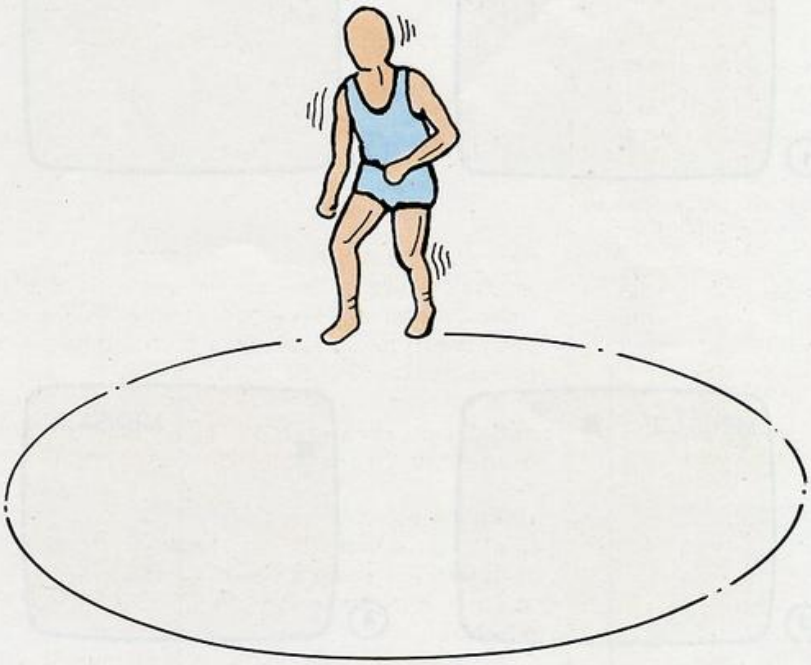
El programa se detiene cuando se acaba el número de instrucciones que lo componen.



El programa «CUADRADOS» sigue el camino señalado por las flechas en el gráfico.



En el programa «CUADRADOS», la instrucción **GO TO 10** *de la línea 40, crea un círculo vicioso en el que se desarrolla el programa. Este tipo de estructuras se denominan BUCLES.*



BITS

Para borrar una línea de instrucción, basta con teclear su número y pulsar **ENTER**.
Para detener un programa durante un **INPUT**, debemos introducir el comando **STOP**.

*

El cursor de programa se desplaza hacia arriba, mediante las teclas **CAPS SHIFT** + **7** (tecla especial en el Plus), y hacia abajo, mediante **CAPS SHIFT** + **6** (tecla especial en el Plus).

dispone de forma que se escriban en la primera columna o en la mitad de la pantalla, según cual sea el punto más próximo.
Vamos a dirigirnos con los cursores de programa a la instrucción 30, que de momento tiene el siguiente aspecto:

30 PRINT "El cuadrado de";N;" es:";C

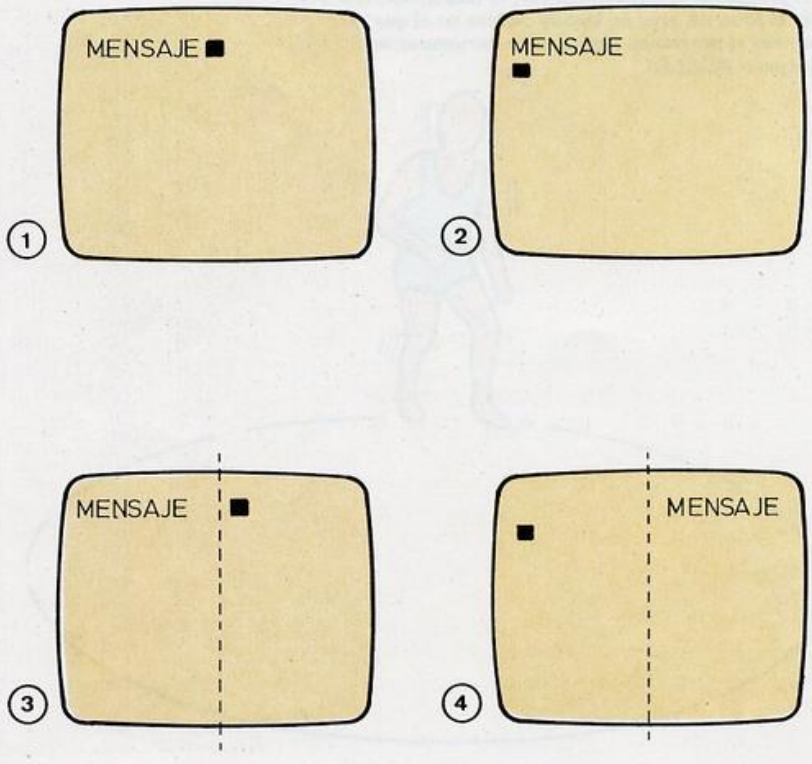
Comencemos por editarla mediante **EDIT**. A continuación, pulsamos **SHIFT** y **8** (cursor derecho) varias veces, hasta que el cursor (en el modo **L**) se sitúe inmediatamente después del ;, que sigue a las comillas finales del literal «El cuadrado de». Hecho esto, pulsamos **DELETE** (**SHIFT** y **0**) hasta que el ; y el literal quedan completamente eliminados. La instrucción debe ofrecer ahora el siguiente aspecto, con el cursor situado inmediatamente antes de la **N**:

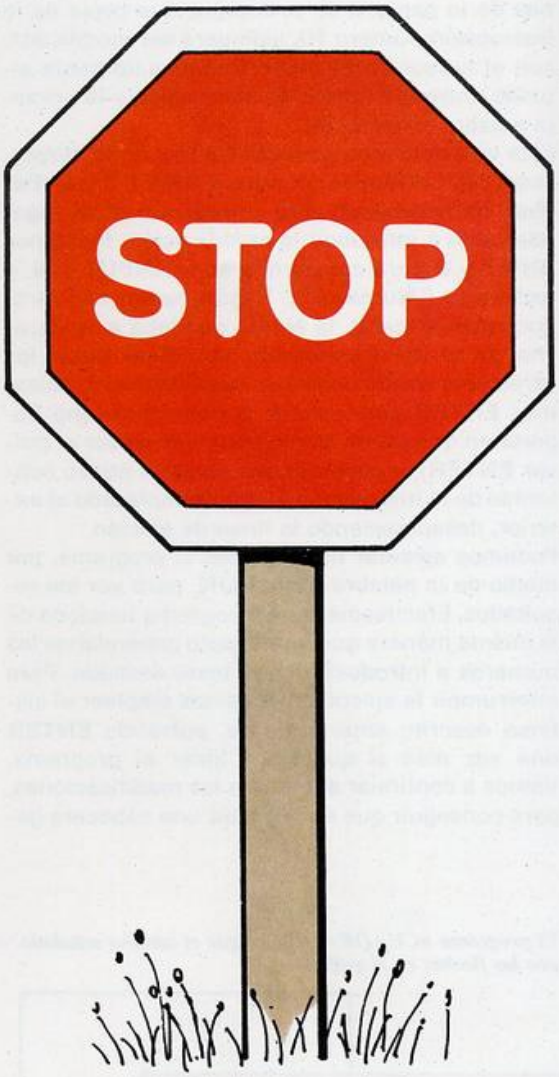
30 PRINT N;" ES:";C

Avanzamos ahora nuevamente el cursor hacia la derecha, para situarnos justamente delante de la **C**. Acto seguido, utilizamos 8 veces la función **DELETE**, para hacer desaparecer el literal que resta y los ;, y tecleamos un caracter ,. Ahora la instrucción debe aparecer como:

30 PRINT N,C

Los separadores punto y coma (;) y coma (,) afectan, a la posición en que se escribirá el próximo literal o resultado (■). I. **PRINT "MENSAJE";...** ***II.*** **PRINT "MENSAJE"...** ***III.*** **PRINT "MENSAJE",...** ***(cuando la impresión anterior termina en la mitad izquierda de la pantalla) IV.*** **PRINT "MENSAJE",...** ***(cuando la impresión anterior termina en la mitad derecha de la pantalla).***





***Mediante el comando* STOP, *podemos detener el programa cuando se encuentra en un* INPUT.**

Esto le indica al ordenador que la primera variable se imprimirá al comienzo de cada línea y la segunda a partir del centro de la pantalla. Otro sistema para modificar la línea 30 habría sido reescribirla, ya que la nueva instrucción 30 sustituiría automáticamente a la antigua al pulsar **ENTER**.
Para dar un último toque a nuestra labor, podemos incluir una cabecera en la pantalla que nos indique el título de cada columna. Esto lo podemos hacer, gracias a que hemos numerado las instrucciones de forma que podemos intercalar nuevos números entre ellas. La cabecera la vamos a incluir en la línea 5, justo antes del comienzo de la petición de datos por **INPUT** (línea 10 a la que se efectua el **GO TO**), por lo que sólo se ejecutará una vez al comienzo del programa.

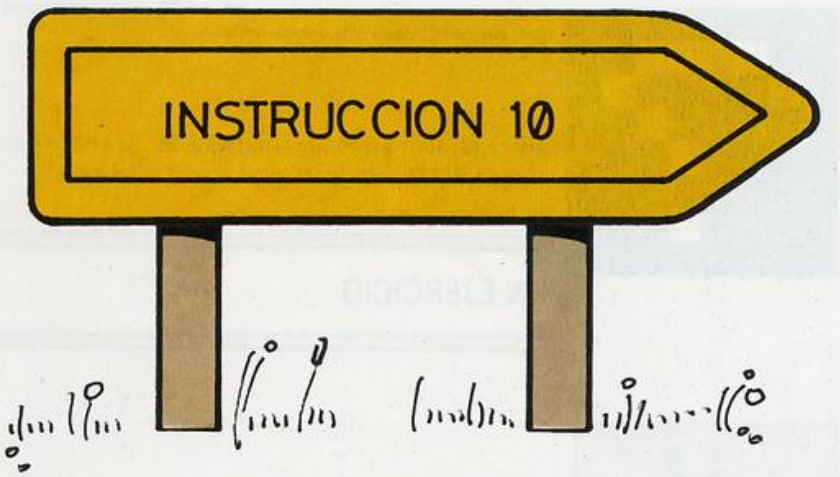
5 PRINT "NUMERO", "CUADRADO"

Después de todas las modificaciones, el programa debe presentar el siguiente aspecto:

```
5 PRINT "NUMERO","CUADRADO"
10 INPUT "Numero:";N
20 LET C=N*N
30 PRINT N,C
40 GO TO 10
```

Ha llegado el momento deseado. Podemos teclear **RUN** para obtener la compensación a nuestros esfuerzos.

PROBLEMAS DE SINTAXIS



El cursor de programa (>) indica la instrucción que aparecerá en la línea de edición al utilizar el comando **EDIT**.

Hasta ahora nos hemos limitado a copiar ejemplos de programas, y por tanto no hemos tenido problemas de ningún tipo; sin embargo, cuando empecemos a hacer nuestros propios programas, la inexperiencia nos hará cometer pequeños errores. El más frecuente es el error de sintaxis, es decir, el que se produce al introducir una instrucción que el ordenador no es capaz de comprender. El Spectrum está dotado de un sistema de detección de errores de este tipo, extremadamente útil para los que estamos aprendiendo. Gracias a él, cualquier instrucción, ya sea en modo directo o programa, no será admitida por el ordenador si contiene algún error de sintaxis. Este sistema se conoce como *syntax checker* (comprobador de sintaxis).

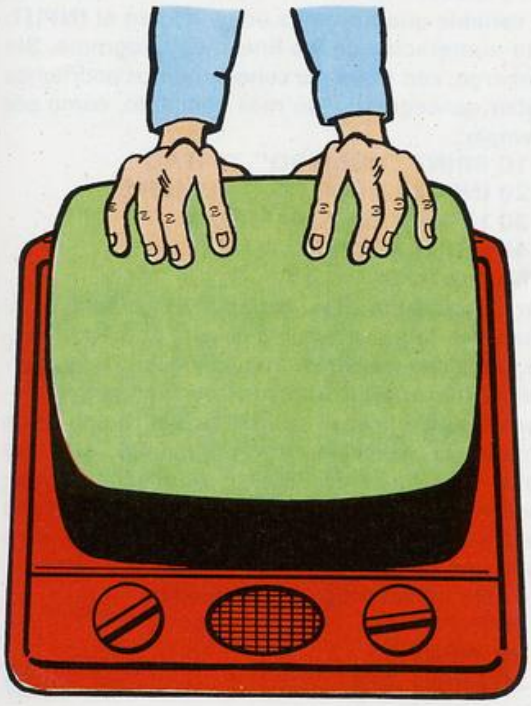
Para estudiar el funcionamiento de este sistema, intentaremos introducir en comando directo una instrucción con un error de este tipo. Por ejemplo, **PRINT "correcto**. Si ahora pulsamos **ENTER**, la línea de edición no desaparecerá, sino que se verá sustituida por la instrucción introducida por nosotros, más un caracter de interrogación en intermitencia, situado entre la instrucción tecleada y el cursor.

¿Qué significa ese caracter? ¿Es un nuevo tipo de cursor que desconocemos? No. Simplemente se trata del sistema por el cual, el *syntax checker* nos indica que se ha producido un error de sintaxis. La posición del indicador, señala el punto hasta el cual la sintaxis se considera correcta. En nuestro ejemplo, el indicador se encuentra al final de la instrucción, ya que el error consiste en no haber completado la misma con el caracter comillas.

En el momento en que se pulse cualquier tecla (distinta de **ENTER**, por supuesto), el indicador de sintaxis (*syntax marker*) desaparecerá, permitiéndonos la modificación de la instrucción.

Para terminar, veamos otro ejemplo. Intentemos introducir **PRINT "2+2= "2+2**. Esta vez, el indicador de error aparecerá entre las comillas que cierran el literal y la constante **2+2**. Esto es debido a que, como ya sabemos, entre los diferentes componentes de una instrucción **PRINT**, debe emplearse un separador (ya sea un punto y coma o una coma). Si retrocedemos con el cursor, la interrogación desaparecerá, y podremos introducir

El scroll consiste en el desplazamiento de la información de la pantalla una línea hacia arriba.



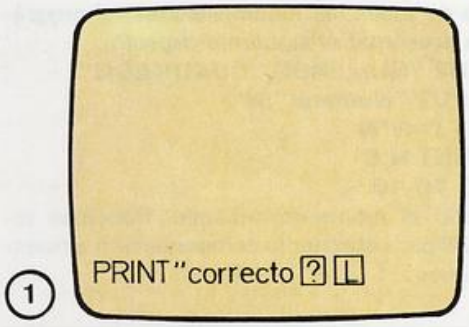
¡!

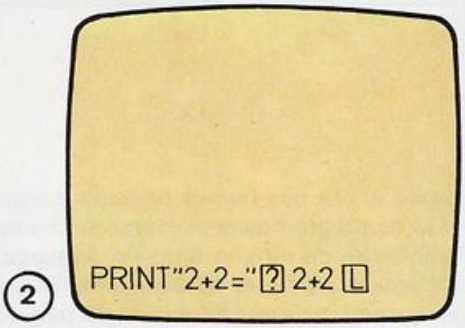
El separador coma (,), desplaza la posición de la próxima impresión a la primera columna de la siguiente línea, o la columna 16 de esa misma línea, según haya sobrepasado o no esta última. El Spectrum está dotado de un sistema de detección de errores de sintaxis. Gracias a él, cualquier instrucción, ya sea en modo directo o en modo programa, no será admitida por el ordenador si contiene algún error de este tipo. Este sistema se conoce como *syntax checker* (comprobador de sintaxis).

El Spectrum utiliza el syntax marker *(?), para indicarnos que no comprende una instrucción.*

un separador en el lugar adecuado. Si ahora pulsamos **ENTER** la instrucción será admitida.

UN EJERCICIO





I. Disposición de la pantalla cuando se produce un error de sintaxis, al intentar introducir la instrucción **PRINT "correcto.**
II. Disposición de la pantalla cuando se produce un error de sintaxis, al intentar introducir la instrucción **PRINT "2+2= "2+2.**

¡!

Normalmente, las instrucciones de programa se numeran de 10 en 10. Esto se hace para permitir intercalar nuevas instrucciones entre las ya escritas.

*

Cuando se introduce una nueva línea de instrucción con el mismo número que alguna ya existente, la nueva pasa a sustituir a la antigua en el programa.

*

Cuando anteponemos un número a lo que sería una instrucción en modo directo, el ordenador entiende que no la debe ejecutar de forma inmediata, sino almacenarla dentro del programa y colocarla en la posición que le corresponde.

Por fin ha llegado el momento de ejercitar nuestros conocimientos. Encendamos nuestro ordenador e intentemos diseñar un programa similar al "CUADRADOS" de ejemplo. Nuestro programa ha de calcular la mitad de los números que introduzcamos, es decir, dividir el número introducido por dos. Ahora vamos a ver unas pequeñas pistas que nos ayudarán a resolver el ejercicio: El número a dividir podríamos introducirlo en una variable mediante un **INPUT.** En cuanto al resultado, se puede calcular como la variable entrada dividida por dos, pudiéndose a su vez asignar este valor a una variable. Una vez completada la entrada y el cálculo, sólo resta realizar la impresión en la pantalla del resultado. Ya podemos ponernos manos a la obra; cuando hayamos conseguido un programa que cumpla las premisas anteriormente establecidas, podremos comprobar las diferentes soluciones que aparecen a continuación.

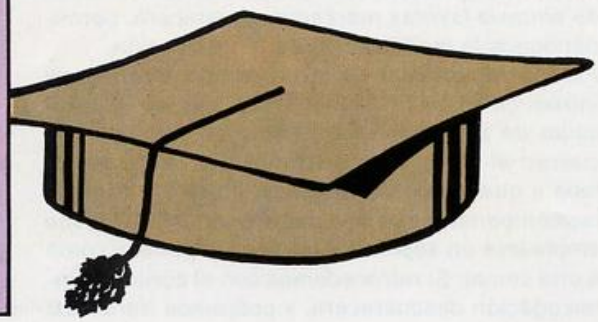
Sin duda, una de las soluciones más simples es:

```
10 INPUT A
20 PRINT A/2
```

También podíamos habernos esforzado un poquito más, y haber programado algo parecido a:

```
10 INPUT A
20 PRINT "La mitad de ";A;" es ";A/2
```

El sistema comprobador de sintaxis (syntax checker) *nos ayudará en la realización de programas, al no admitir líneas que contengan errores de este tipo.*



Por supuesto, no tiene importancia el nombre de la variable que hayamos empleado en el **INPUT**, o la numeración de las líneas del programa. Sin embargo, con nuestros conocimientos podríamos haber conseguido algo más completo, como por ejemplo:

```
10 PRINT "NUMERO","MITAD"
20 INPUT "NUMERO:";NUMERO
30 LET MITAD=NUMERO/2
40 PRINT NUMERO,MITAD
50 GO TO 20
```

Para estar contentos, deberíamos conseguir un programa lo más parecido posible al último, desde luego con nuestras propias ideas, pero procurando utilizar todos nuestros conocimientos. Podemos ejercitarnos a partir de ahora con otros programas parecidos, como por ejemplo, uno que halle el doble de un número, o incluso uno que realice la suma de dos números, aunque en ese caso deberemos utilizar dos **INPUT**, uno para cada uno de los sumandos. Ahí va una pista sobre este último programa propuesto:

```
10 INPUT A
20 INPUT B
30 PRINT A+B
```

DISEÑO DE PANTALLAS

TODOS sabemos, que la pantalla es el medio fundamental que el ordenador utiliza para comunicarse con nosotros. Por ello, a la hora de confeccionar buenos programas, hemos de poner gran empeño en hacer de la pantalla un agradable vehículo de información; esto se consigue cuidando la estética de la misma en sus diferentes aspectos:

— Diseño de la pantalla.
— Utilización del color.
— Dinamismo.

En los tres puntos anteriormente expuestos, se pueden concretar nuestras acciones sobre la pantalla para conseguir una presentación agradable de la misma. A continuación, vamos a dar repaso a una serie de trucos y consejos acerca de este asunto.

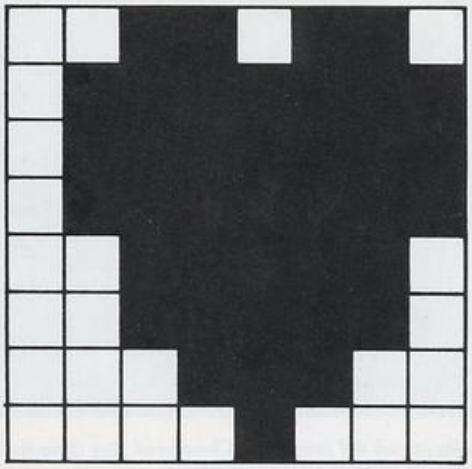


Gráfico definido de un corazón, que puede ser empleado para la confección de naipes, en cualquier juego basado en la baraja francesa.

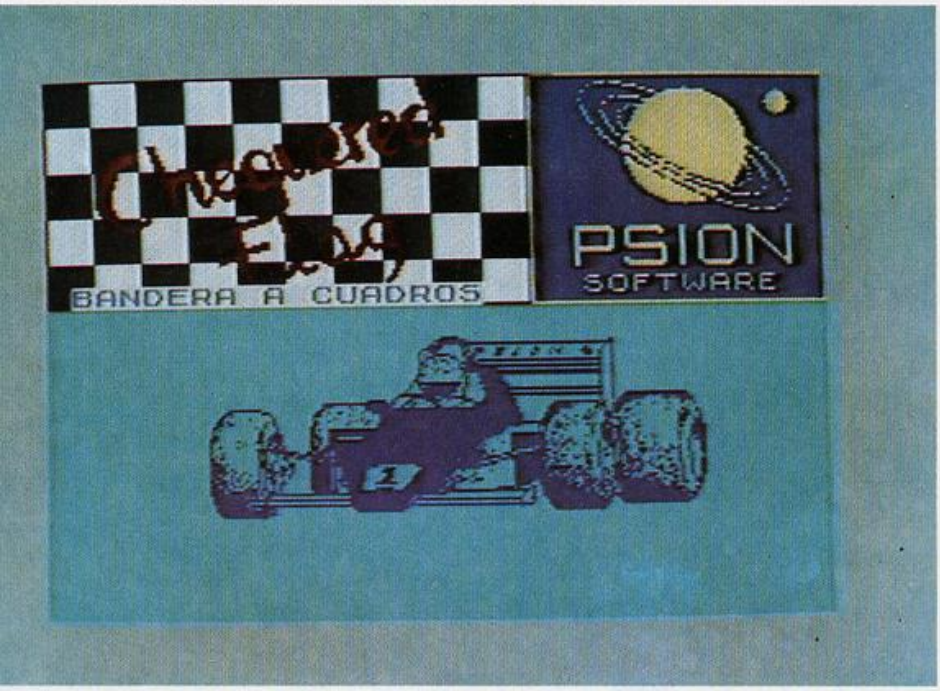
DISEÑO DE LA PANTALLA

Indudablemente, el diseño de la pantalla en sí es una parte esencial dentro de la estética de la misma, pero no tanto por la complejidad de las figuras que la integran, como por su distribución en el espacio reservado a la presentación visual. Durante estas primeras semanas de nuestra andadura por el BASIC del Spectrum, hemos tenido oportunidad de ver muchas figuras de considerable belleza, cuya realización no entraña por el contrario complejidad alguna, sino que son llevadas a cabo con tan solo una instrucción.

Llegados a este punto, debemos hacer una distinción entre dos aspectos fundamentales que contribuyen a un buen diseño de la pantalla: por una parte, las figuras que se presentan en ella y por otra, su distribución en la pantalla.

En el primer punto, influirán muy directamente los medios de que dispongamos para el diseño de las figuras y, desde luego, el tiempo que podamos invertir en su realización. En el mercado existen programas de utilidad, destinados precisamente a la realización de dibujos en la pantalla. Desde los modestos generadores de caracte-

ULTIMATE, PLAY THE GAME, *es una firma británica de reconocido prestigio, que destaca por la gran calidad de las pantallas de sus programas.*



Pantalla inicial del programa **Chequered flag** *(Bandera a cuadros) de* **PSION COMPUTERS.**

res, hasta los complejos programas conocidos como diseñadores de pantallas. Estos facilitarán enormemente nuestra labor estética, aunque sin ellos, pero con un mayor esfuerzo, también se pueden conseguir interesantes resultados.
Sin embargo, otro de los puntos importantes en la realización de un bonito trabajo, es la distribución inteligente de los «bultos» en la pantalla. Veamos esto mediante un ejemplo: supongamos que queremos realizar una pantalla de presentación de un juego basado en una baraja francesa. La primera fase del diseño es puramente imaginativa. En ella decidiremos cuál es la pantalla que deseamos realizar. Sin un gran esfuerzo, podemos llegar a la conclusión que una de las alusiones más claras a los juegos de este tipo, es el tapete de juego (verde) sobre el que se disponen los cuatro ases de la baraja.
Una vez que sabemos el tipo de diseño que vamos a realizar, debemos poner manos a la obra. El dibujo del tapete no será cosa complicada; bastará con asignar un código de color 4 al marco y al fondo de la pantalla, con lo que tras una instrucción **CLS** generaremos un perfecto tapete de juego. Llevémoslo a la práctica:

```
20 BORDER 4: PAPER 4: BRIGHT 0: CLS
```

Seguro que en la instrucción hemos observado un pequeño detalle: la desconexión del modo **BRIGHT.** De no ser así, y puesto que el brillo suplementario no afecta al marco de la pantalla, tendríamos un tapete verde claro por su zona interior, y verde oscuro en su periferia, cosa no muy común entre los paños de este tipo.
Un último toque podría aportarlo una línea roja de delimitación de los bordes del tapete. Aunque, según lo considere el programador, ésta puede tener un efecto negativo: reducir las dimensiones aparentes de la pantalla. Por tanto, vamos a dejar la línea de delimitación en una instrucción independiente, con el fin de poder suprimirla en el momento que nos interese.

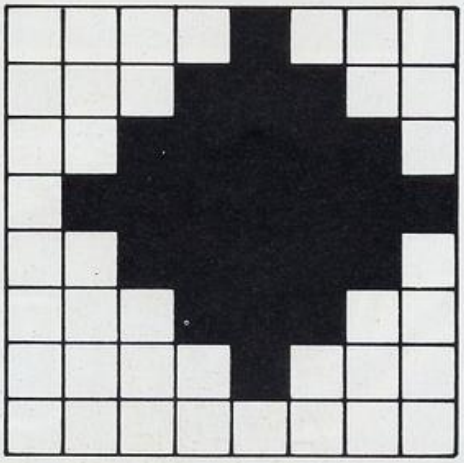
```
30 INK 2: PLOT 10,0: DRAW -10,10,PI/2: DRAW 0,155: DRAW 10,10,PI/2: DRAW 235,0: DRAW 10,-10,PI/2: DRAW 0,-155: DRAW -10,-10,PI/2: DRAW -235,0
40 PLOT 12,2: DRAW -10,10,PI/2: DRAW 0,151: DRAW 10,10,PI/2: DRAW 231,0: DRAW 10,-10,PI/2: DRAW 0,-151: DRAW -10,-10,PI/2: DRAW -231,0
```

La instrucción 40 traza otra línea de delimitación, un **pixel** más adentro que la dibujada por la 30.

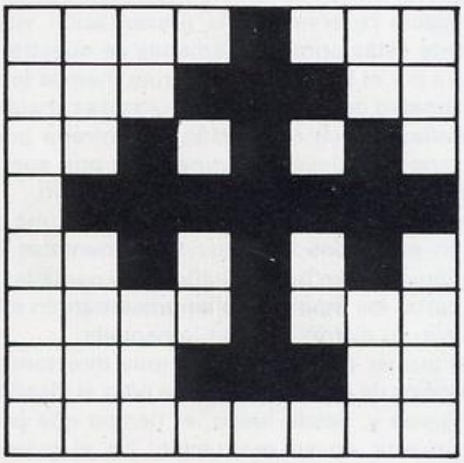
BITS

Una de las muchas aplicaciones de **OVER,** es conseguir la edición de mensajes parpadeantes, mejorando el efecto estético que se consigue mediante el modo **FLASH.** He aquí un ejemplo de ello.

```
10 PRINT AT 10,5;"MENSAJE PARPADEANTE"
20 OVER 1: PAUSE 25: GO TO 10
```



Para conseguir la simetría en el gráfico del diamante, ha sido necesario dejar libres la primera columna y la última fila.



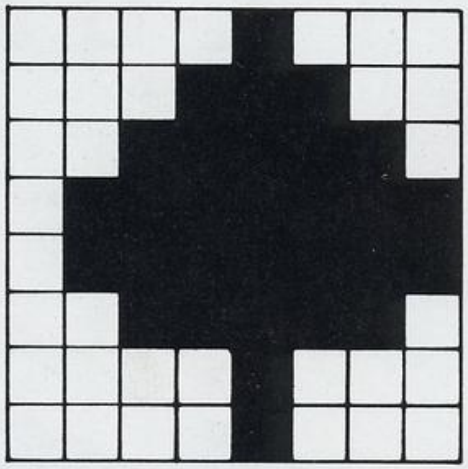
De toda la baraja, el gráfico más complicado de diseñar es el trébol, debido al detalle necesario para sus hojas.

La eliminación de ambas instrucciones producirá la supresión de la doble raya del tapete, sin afectar a ninguna otra instrucción posterior.

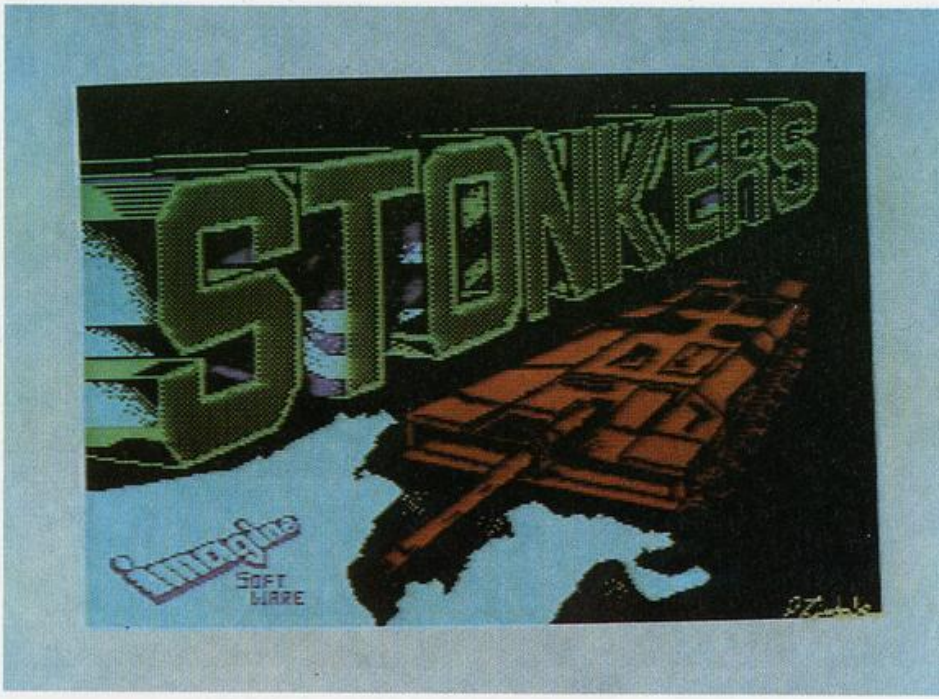
LOS NAIPES

Por fin, ha llegado la hora de perfilar el motivo central de la pantalla. Como en el caso del tapete, la realización del fondo blanco de la carta no supone mayores dificultades. Sin embargo, otro asunto es la realización del dibujo interior y la adecuada proporción de los naipes. La política más acertada, en este caso sería emplear para la pantalla los mismos ases que se utilizarán más tarde en el juego, con lo que nos ahorraremos una buena cantidad de trabajo. Las dimensiones de las cartas elegidas para nuestro ejemplo son 9 de alto y 7 de ancho, expresadas en posiciones de carácter (9 filas × 7 columnas).
Lo mejor sería, desde luego, plasmar el fondo de las cartas mediante subrutina, pues será una tarea que habrá que realizar muy frecuentemente en el juego. Además, la posición donde se debe efectuar la impresión variará, por lo que ha de ser indicada a la subrutina mediante unos parámetros; por ejemplo, F (fila) y C (columna).

```
999 STOP: REM Fin de programa principal
1000 FOR I=0 TO 8: PRINT AT F+I,C; PAPER 7; "       ": NEXT I
1020 RETURN
```



La pica es el último de los gráficos definidos para la baraja, con unas características muy similares al corazón en cuanto a dimensiones.



***Pantalla de presentación del programa de estrategia militar* Stonkers.**

Antes de pasar al dibujo de las cartas, vamos a completar la subrutina que dibuja el naipe con un detalle más: un marco que evite la difuminación del blanco sobre el fondo verde del tapete. Para ello utilizaremos una línea muy fina, del color de la carta; es decir, roja para corazones y dia-

Penetrator, *una versión para el Spectrum del popular* Scramble.

Pantalla inicial del programa Atic-Atac.

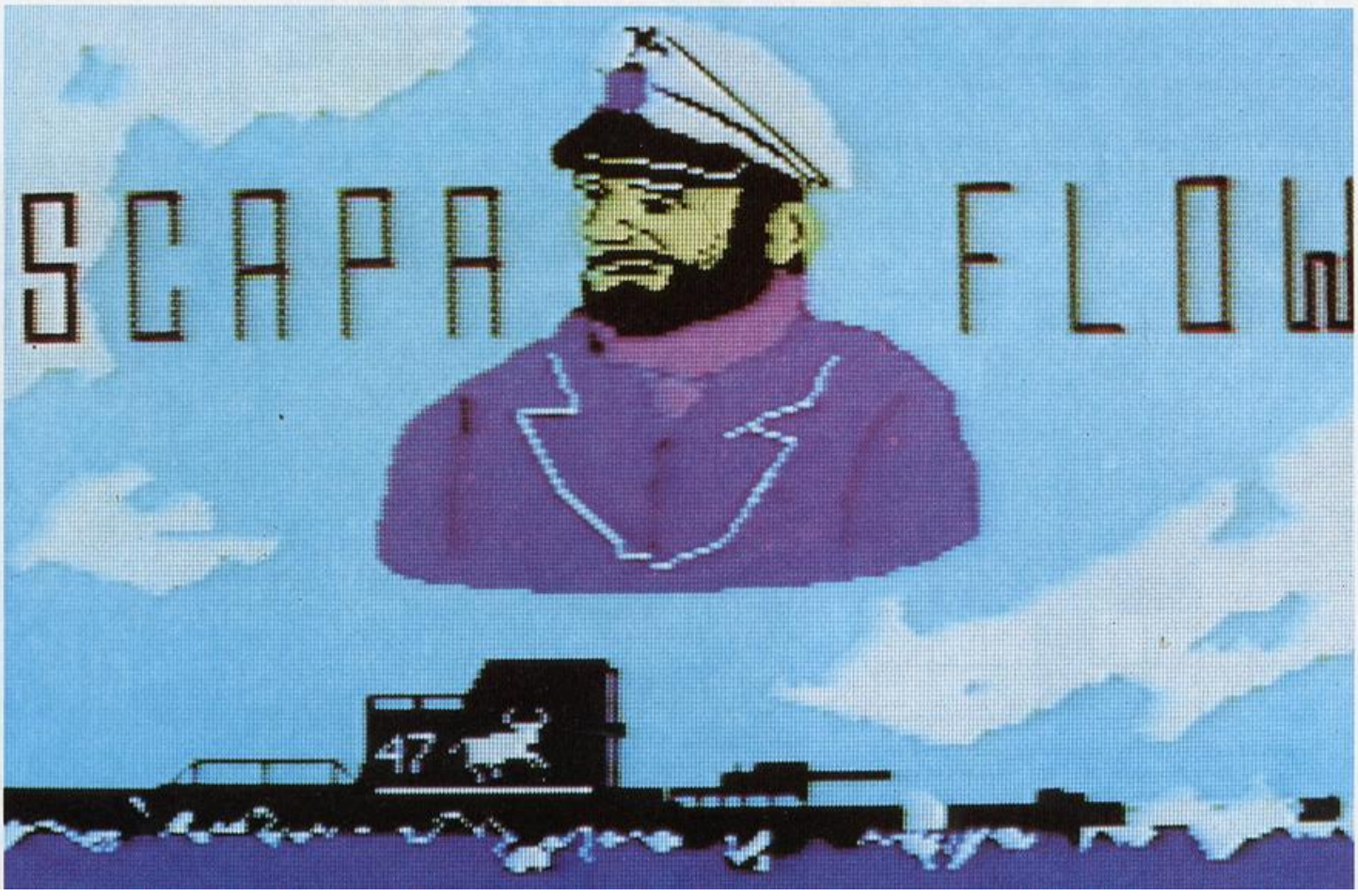
Nosotros también podemos diseñar nuestras propias pantallas. He aquí una muestra de ello.

mantes, y negra para tréboles y picas. Evidentemente, se nos hace necesario un tercer parámetro, que indique a la subrutina el color de la carta. Utilicemos por ejemplo la variable CC, que contendrá el código de color de la carta (0=negro y 2=rojo).

```
1020 LET FP=F*8
1030 LET CP=175-C*8
1040 INK CC: PLOT FP,CP
1050 DRAW 56,0: DRAW 0,-72: DRAW -56,0: DRAW 0,72
1060 RETURN
```

El siguiente paso es, sin duda alguna, la definición de unos caracteres gráficos, que representen cada uno de los palos de la baraja. Esto será trabajo fácil, ya que la cinta de demostración que acompaña al Spectrum contiene un programa generador de caracteres, que nos ayudará en esta tarea. Por tanto, en el programa principal incluiremos la definición de los caracteres, que en nuestro ejemplo se han localizado en las letras A, B, C y D, para corazón, diamante, trébol y pica, respectivamente.

```
50 FOR I=0 TO 31: READ A: POKE USR "A"+I,A: NEXT I
60 DATA 54,127,127,127,62,62,28,8
70 DATA 0,8,28,62,127,62,28,8
80 DATA 8,28,42,127,42,8,28,0
90 DATA 8,28,62,127,127,62,8,8
```



Para evitar problemas de color, se pueden difuminar los contornos de las figuras.

Para los que no dispongamos de más facilidades que el programa generador de caracteres, la primera parte de la labor de diseño ya ha terminado. Sólo es necesario situar el correspondiente gráfico en el centro de la carta y en las esquinas superior izquierda e inferior derecha de la misma. Por el contrario, los más afortunados en cuanto a los medios de diseño gráfico, podremos acometer el dibujo de unos ases de grandes dimensiones. El sistema de codificación de los mismos variará a gusto del programador, pero una manera bastante cómoda de almacenar la forma de los ases puede ser unas sentencias **DATA**, que contengan direcciones de **PLOT** y coordenadas de **DRAW**.
Con todo lo visto hasta el momento, seremos capaces de diseñar nuestra propia pantalla. Sólo tenemos que ensamblar adecuadamente las diferentes partes que hemos ido estudiando. Por supuesto, las rutinas aquí expuestas son meras sugerencias, que pueden ser complementadas y mejoradas a gusto del programador; por ejemplo, buscando sistemas de dinamismo en la pantalla, para que ésta no resulte tan estática. Una idea en este sentido, es establecer la entrada de mensajes con el título del programa, nuestro nombre o las instrucciones.

ULTIMOS CONSEJOS

Para terminar, vamos a dar un pequeño repaso, a todos los aspectos importantes que hemos visto para el diseño de una pantalla.
* Es muy importante que la pantalla sea representativa de la idea que se desea comunicar.
* Debemos procurar, en lo posible, enmarcar con una línea fina las masas de color superpuestas sobre fondos distintos. Cuanto más pequeña sea la zona a enmarcar, más necesaria será esta medida. Sin llegar, por supuesto, al extremo de delimitar zonas de un solo carácter.
* La delimitación mediante línea del marco de la pantalla puede ser un arma de doble filo, ya que inevitablemente reduce las dimensiones aparentes de la pantalla.

La combinación de colores en la pantalla, requiere bastante experiencia para poder obtener resultados como el de la foto.

* Es fundamental la distribución de las figuras. Estas deben guardar simetría consigo mismas y con la pantalla. Es necesario eliminar, en lo posible, los espacios vacíos respecto a espacios excesivamente cargados.
Bien, esperamos que estos consejos hayan sido lo suficientemente aleccionadores, como para convertirnos en unos grandes diseñadores de pantallas. El tema aún nos reserva algunas agradables sorpresas, como por ejemplo, cuando abordemos los espectaculares efectos de dinamismo de la pantalla, que se pueden conseguir con unos pocos trucos.

SATURNO JET

COMO piloto más experimentado de la Armada Interplanetaria, se te ha encomendado la difícil misión de tripular la nave Saturno Jet, en misión de reconocimiento por los anillos del planeta Saturno.

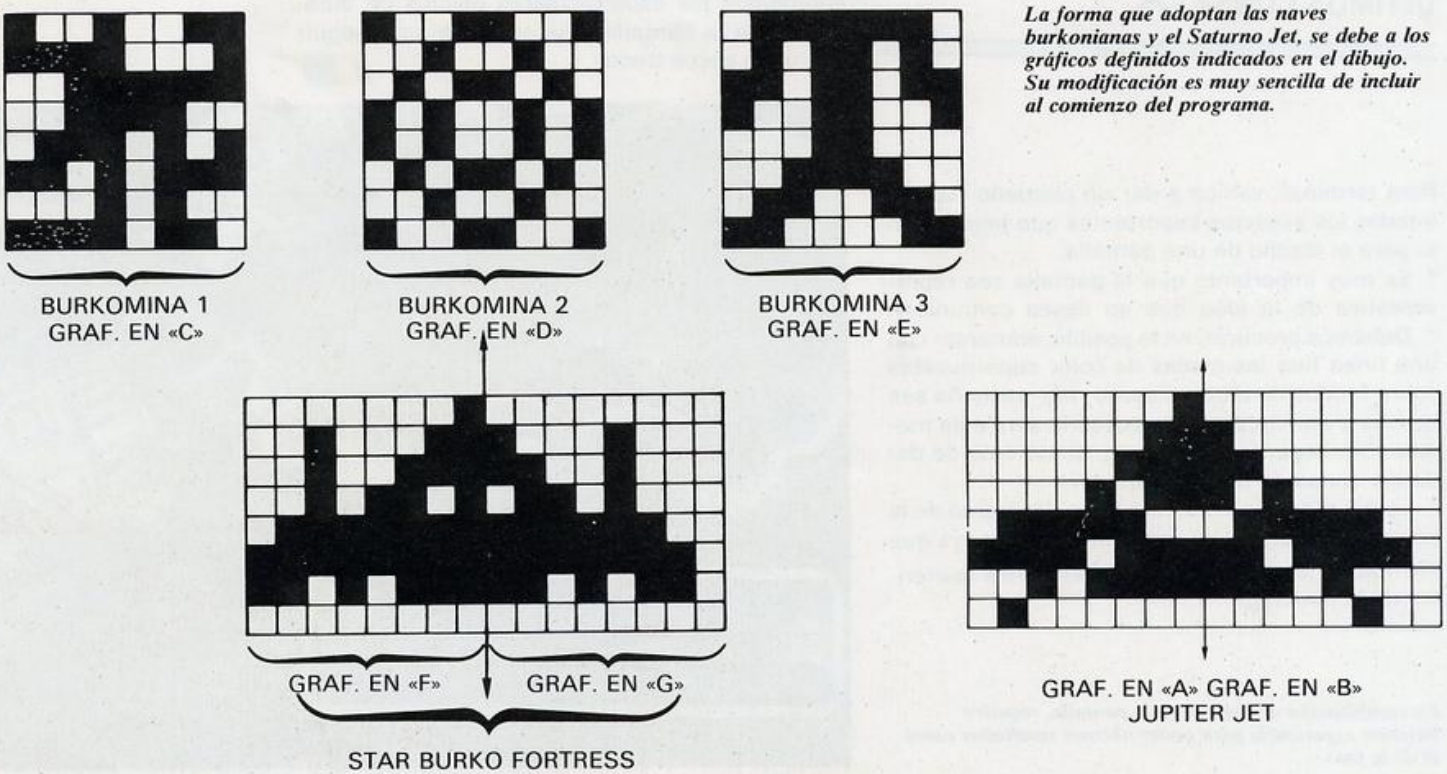
Las últimas noticias llegadas de la base Lambda 4 en el satélite Titán, son bien poco alentadoras: los malvados habitantes del planeta Burko (según se sale de Tau Ceti, el tercer planeta a la izquierda), han enviado su flota de conquista a nuestro sistema solar.

La única manera de hacerles frente con ciertas posibilidades de victoria, es conocer su poderío concreto; por ello, han distribuido sus naves en la zona de los anillos de Saturno, haciéndolas invisibles a los medios habituales de detección terrestres. Tu objetivo es tripular por control remoto la nave de reconocimiento Saturno Jet, introduciéndote en las líneas burkonianas, para informar de la composición de su escuadra.

Cuanto más te adentres en los anillos, tanto más difícil será tu misión, pero más importante será la información que los ojos electrónicos de tu nave transmitan. No dudes que para evitar tu acción de espionaje, los pérfidos burkonianos utilizarán todos los medios a su alcance.

Iniciarán sus operaciones disuasorias mediante el lanzamiento de Burkominas y, si consigues esquivarlas, pondrán en juego sus Star Burko Fortress (naves burkonianas de invasión masificada), en un último esfuerzo por conseguir lo que las Burkominas no lograron llevar a buen término. Recuerda siempre que tu nave es de reconocimiento y, como tal, no dispone de ningún mecanismo de ataque o defensa, sino simplemente de una gran maniobrabilidad. Deberás evitar siempre el choque con cualquiera de las naves burkonianas, o su zona de influencia (una columna de pantalla intergaláctica a cada lado), ya que ésto supondría la inmediata destrucción de la nave que tripulas.

Para el control del Saturno Jet, emplearás las teclas **Z** (izquierda) y **X** (derecha), presentes en el panel de control de tu ordenador, en el Cuartel General Terrestre.



La forma que adoptan las naves burkonianas y el Saturno Jet, se debe a los gráficos definidos indicados en el dibujo. Su modificación es muy sencilla de incluir al comienzo del programa.

El programa se basa en la utilización de la variable del sistema SCR CT **(SCRoll CounTer),** localizada en la dirección decimal 23692. La misión de esta variable, es contar el número de líneas desplazadas fuera de la pantalla, desde el último mensaje de **Scroll?** emitido. Evitando mediante POKEs que alcance el valor cero, conseguiremos que las oleadas de «estorbos espaciales» no se

BITS

El control de la variable del sistema **SCR CT**, *permite evitar la aparición del desagradable mensaje* **Scroll?**, *en plena batalla.*

Al introducir el programa, no olvidemos sustituir los caracteres subrayados, por los gráficos definidos correspondientes a las teclas indicadas.

*

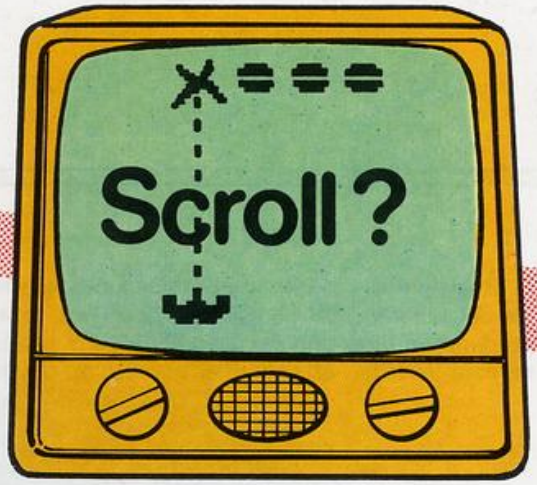
Para el movimiento del Saturno Jet, emplearemos las teclas **Z** (izquierda) y **X** (derecha). Tengamos siempre bien presente, que debemos evitar la colisión con las naves burkonianas, puesto que no disponemos de dispositivos de defensa y ataque.

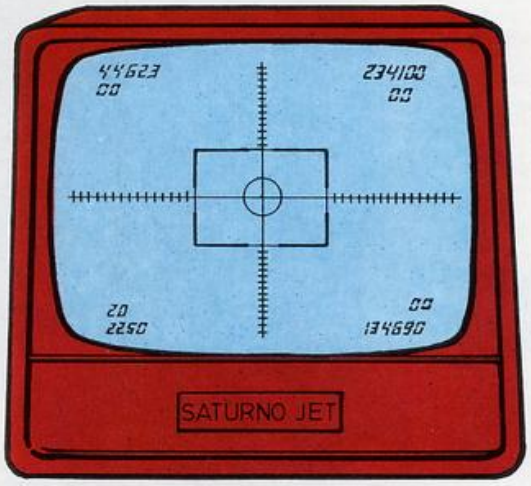
detengan en ningún momento, con el mensaje de desplazamiento de la pantalla.

La forma de simular el movimiento de las naves burkonianas ha sido bien simple: todo se ha limitado a escribir por cada ciclo de programa, una nave enemiga en un punto aleatorio de la última línea de la pantalla; al forzarse el carácter **RETURN** después de la escritura de cada nave, se efectúa un desplazamiento automático de la pantalla **(scroll)**, por haberse realizado la representación en la última línea de ésta.

Para la introducción del programa, hay que seguir la norma habitual de sustituir los caracteres que aparecen subrayados en el listado, por los gráficos definibles correspondientes a las letras subrayadas. Una vez que el listado se encuentre en la memoria, podemos grabarlo mediante **SAVE "SATURNO" LINE 1**.

El momento en que nuestra nave pase a convertirse en chatarra espacial, llegará tarde o temprano, y entonces el juego se dará por terminado; pero antes informará sobre cuántas naves burkonianas hemos conseguido evitar. Con este marcador, podemos establecer competiciones con nuestros amigos, para contrastar nuestra pericia ante los mandos del Saturno Jet.





Los ojos electrónicos del Saturno Jet, transmitirán una preciada información sobre la potencia de la flota burkoniana.

```
 10 REM   ******************
 20 REM   ******************
 30 REM   **  J.M.MAYORAL  **
 40 REM   ******************
 50 REM   ******************
 60 BORDER 0: PAPER 0: INK 9
 70 POKE 23658,8: CLS
 80 LET a$="ABCDEFG"
 90 RESTORE 160
100 FOR K=1 TO 7
110 FOR N=0 TO 7
120 READ P
130 POKE USR A$(K)+N,P
140 NEXT N
150 NEXT K
160 DATA 1,3,7,27,124,247,47,64
170 DATA 0,128,192,176,124,222,232,4
180 DATA 147,241,63,60,117,215,20,246
190 DATA 153,36,90,165,165,90,36,153
200 DATA 126,90,219,153,24,60,102,195
210 DATA 1,35,39,45,127,255,215,0
220 DATA 0,136,200,104,252,254,214,0
230 LET PAR=6: LET A$="CDEFG"
240 LET K=1
250 LET N=0
260 LET CERO=0: LET BTM=0: LET COL=0: LET DES=0
270 LET T=1
280 LET X=2*PAR
290 LET ALEAT=INT (RND*32)
300 IF N>=300 THEN  GO TO 320
310 LET B$=A$(K)
320 PRINT AT 21,ALEAT; INK INT (RND*7)+1;B$
330 POKE 23692,255
340 PRINT AT 21,31;"  "
350 REM
360 PRINT AT 10,X-2;"      "
370 PRINT AT 11,X;"AB"
380 PRINT AT 21,31;"  "
390 LET N=N+T
400 IF N=100*3 THEN  LET T=2: LET B$="FG": GO TO 420
410 IF N=100*K THEN  LET K=K+1
420 LET PUM=DES: LET DES=COL: LET COL=BTM: LET BTM=C
ERO: LET CERO=ALEAT
430 PRINT AT 10,X-2;"      "
440 PRINT AT 11,X;"AB"
450 IF X>=PUM-2 AND X<=PUM+T THEN  GO TO 500
460 REM
470 IF INKEY$="Z" THEN  LET X=X-T
480 IF INKEY$="X" THEN  LET X=X+T
490 GO TO 290
500 FOR V=1 TO 5
510 PRINT  INK INT (RND*7)+1; OVER 1;AT 11,X;A$(V);A
T 11,X+1;A$(V)
520 BEEP .1,-10
530 NEXT V
540 PRINT  PAPER 7;AT 21,0;" PUNTUACION="; PAPER 6;N
550 PRINT  PAPER 6; FLASH 1; BRIGHT 1;AT 0,3;"PULSA
UNA TECLA PARA JUGAR"
560 IF INKEY$="" THEN  GO TO 560
570 RUN 230
```