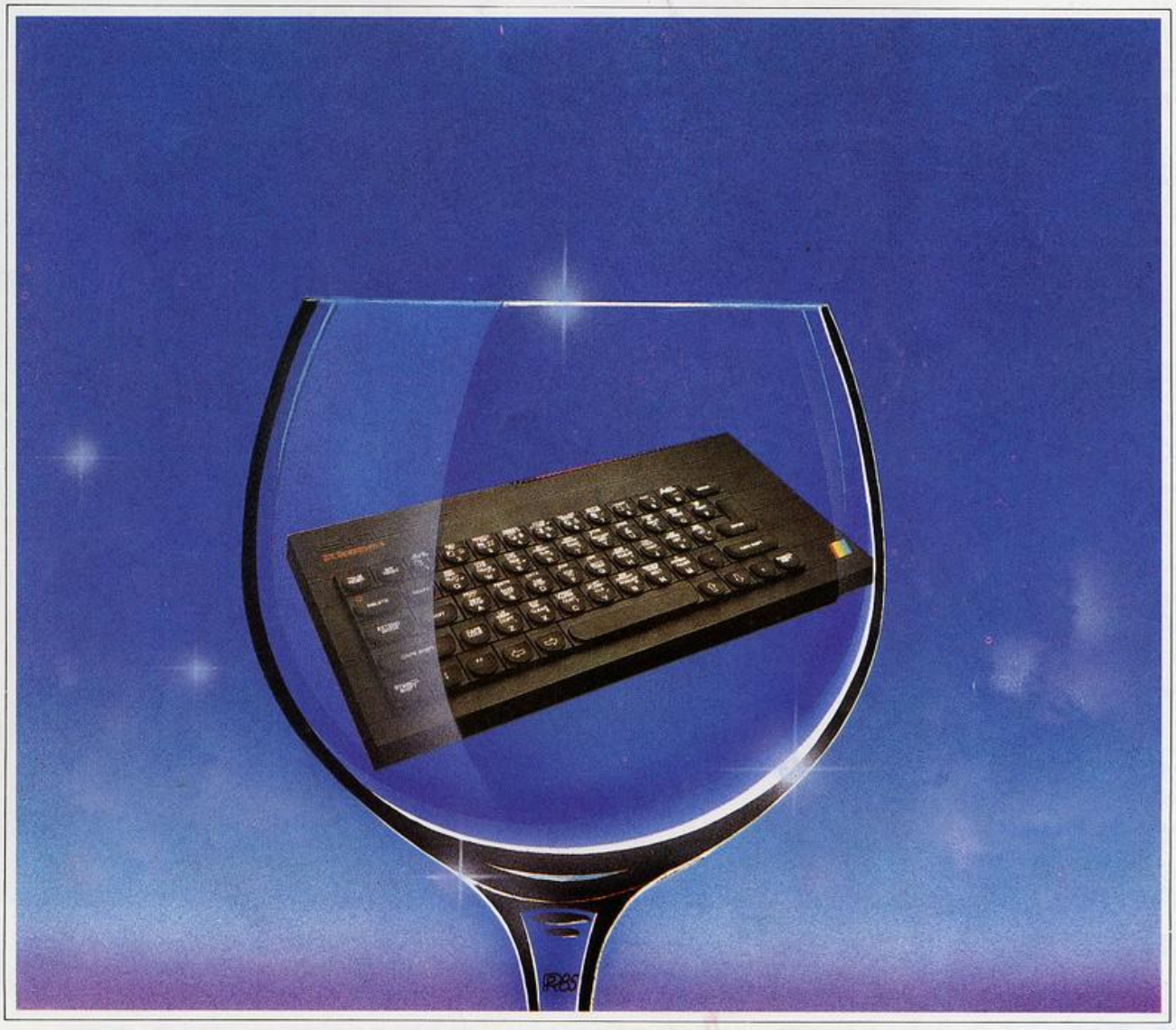
14
150 pts.

RUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

RUN

Enciclopedia Práctica del Spectrum

una publicación de

EDICIONES NUEVA LENTE, S. A., Y EDICIONES INGELEK, S. A.

Director editor por NUEVA LENTE
MIGUEL J. GOÑI
Director editor por INGELEK
ANTONIO M. FERRER
Director de producción
RICARDO ESPAÑOL
Jefe de producción
SANTOS ROBLES
Director de la obra
FERNANDO LOPEZ MARTINEZ
Coordinación
VICENTE ROBLES
Colaboradores
JUAN MANUEL LOPEZ MARTINEZ
CARLOS DE LA OSSA
JUAN MANUEL MAYORAL
Diseño gráfico
JOSE OCHOA
Maquetación
CARLOS GONZALEZ-AMEZUA
Ilustraciones
JOSE OCHOA
ALFONSO MENDEZ
ANTONIO PERERA
Fotografía
(Equipo Gálata)
ALBINO LOPEZ y
EDUARDO AGUDELO
Ediciones Nueva Lente, S. A.
Dirección y Administración:
Benito Castro, 12
28028 Madrid. Tel. 2454598
Ediciones Ingelek, S. A.
Números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Publicidad:
LOLA GONZALEZ
CARMINA FERRER
Tel. 4576923

Plan general de la obra:
52 fascículos de aparición semanal
encuadernables en cuatro tomos
de 13 fascículos

Distribución en España:
COEDIS, S. A. Valencia, 245.
08007 Barcelona
Distribución en Argentina:
Capital: Ayerbe
Interior: DGP
Distribución en Colombia:
DISUNIDAS, Ltda.
Distribución en Chile:
Alfa Ltda.
Distribución en Ecuador:
Muñoz Hermanos, S. A.
Distribución en México:
IMTERMEX, S. A.
Lucio Blanco, 435
México D. F.
Distribución en Paraguay:
Selecciones, SAC.
Distribución en Perú:
DISELPESA
Distribución en Puerto Rico:
Agencia de Publicaciones de Puerto Rico, Inc.
Distribución en Uruguay:
Ledian, S. A.
Distribución en Venezuela:
CONTINENTAL
Editor para Chile:
Editorial Andina, S. A.
La Concepción, 311. Santiago-9
Importador exclusivo Cono Sur:
CADE, SRL
Pasaje Sud América 1532. Tel. 212464
Buenos Aires-1.290. Argentina.

Fotomecánica: OCHOA
Ricardo Ortiz, 74. Madrid.
Impresión: Gráficas Reunidas, S. A.
Avda. de Aragón, 56. Madrid
ISBN de la obra: 84-7534-118-7
ISBN del fascículo: 84-7534-119-5
ISBN del tomo primero: 84-7534-120-9
Depósito legal: M-9896-1985
PRINTED IN SPAIN

Ediciones Nueva Lente, S. A. y Ediciones Ingelek, S. A. garantizan la publicación de todo los fascículos que componen esta obra y el suministro de cualquier número atrasado o tapa mientras dure la publicación y hasta un año después de terminada. El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra si las circunstancias del mercado así lo exigen.

JUNIO 1985

EL AZAR INFORMATIZADO

ARROJAR una moneda al aire, es una práctica común para la resolución de situaciones conflictivas: quién paga la merienda, quién inicia el partido, y otras muchas de extraordinaria trascendencia. Se utiliza en definitiva el azar como factor de decisión, pues nos parece algo absolutamente involuntario. Sin embargo, una ciencia tan exacta como las matemáticas, también se ha ocupado del estudio de la «suerte», y mediante el CALCULO DE PROBABILIDADES, aunque no es capaz de asegurarnos si saldrá cara o cruz, sí puede informarnos de nuestra probabilidad de empezar el partido o pagar la merienda, aunque evidentemente, puede ser que el azar nos juegue una mala pasada.

Al tipo de sucesos que dependen de la suerte, como los que hemos visto hace un momento, se les conoce como SUCESOS ALEATORIOS, y el BASIC es el único lenguaje de programación, entre los de mayor difusión, que tiene capacidad para emularlos; en pocas palabras, el BASIC es capaz de arrojar una moneda al aire, tirar un dado, o darle la vuelta a un bombo de lotería. Más adelante comprobaremos, gracias al ordenador, cómo los sucesos aleatorios siguen determinadas tendencias, que pueden ser estudiadas matemáticamente (probabilidades); pero antes vamos a observar, la forma en que el lenguaje de programación trata los sucesos aleatorios.

Como en la mayoría de los casos, el tratamiento que el ordenador da a la «suerte», es numérico, es decir, la habilidad aleatoria del BASIC se concreta en la generación de números al azar. Esto es suficiente, y es tarea del programador interpretar estos números que la máquina proporciona. Así por ejemplo, si queremos simular la tirada de un dado, pediremos a nuestro Spectrum que encuentre un número natural (entero y sin decimales) al azar entre 1 y 6; si lo que queremos es arrojar una moneda al aire, el número podría ser 1 ó 2 (1=Cara, 2=Cruz), etc...

Aparentemente, no debería ser nada difícil para un ordenador producir un número aleatorio, dada su extraordinaria facilidad para la realización de complejos cálculos numéricos, pero si nos fijamos un poco, observaremos que los resultados de todos esos cálculos y operaciones complejas a que nos referimos, son siempre muy concretos y en absoluto aleatorios.

La técnica para obtener un número al azar, como el que produce el BASIC, es bastante sofisticada.

Dos son las sentencias que controlan el proceso de generación de números aleatorios: **RANDOMIZE** (abreviado en el teclado a **RAND**) y **RND**. Antes de entrar en materia, hemos de hacer una pequeña aclaración sobre lo «trucados» que están los dados y monedas del BASIC.

PSEUDOALEATORIEDAD

Bajo el complicado nombre con que titulamos este epígrafe, se esconde una característica importante de los números que el BASIC genera al azar. Estos números, se denominan PSEUDOALEATORIOS, es decir, PROXIMOS a la aleatoriedad, porque dado su método de obtención, no dependen completamente de la suerte.

Básicamente, para la extracción de números aleatorios, el Spectrum toma un número inicial, denominado SEMILLA (en inglés *seed),* a partir del cual realiza una serie de cálculos de relativa complejidad, que desfiguran este valor inicial; sin embargo, dada la exactitud matemática, a una misma semilla siempre corresponde un mismo número pseudoaleatorio.

En cualquier caso, la definición de pseudoaleatorio es bastante estricta, puesto que, aunque a no-

Una máquina tan precisa como nuestro Spectrum, también puede simular ciertos sucesos que se producen al azar.

¡!

El algoritmo para la obtención de números naturales aleatorios entre límites, es de gran importancia: **R=A+INT (RND*B);** donde R es el número aleatorio natural, A el límite inferior y B la diferencia entre el límite superior y el inferior. Tengamos en cuenta que en esta fórmula el límite superior sí se alcanza.

*

Los números que se obtienen mediante la función **RND** no son completamente aleatorios, aunque en la práctica se pueden considerar como tales.

El azar informático no se puede considerar completamente aleatorio.

sotros nos parezca que la generación de números es realmente fruto exclusivo de la casualidad, no menos que una auténtica lotería, se trata realmente de la selección de números de una serie de 65536 semillas, bastante grande para aparentar que los números no se repiten pero, desde luego, no infinita.

Una vez conocidas las limitaciones que la informática presenta en el campo de la aleatoriedad, pasaremos a estudiar el formato en que el Spectrum nos proporciona los números al azar. Estos se encuentran siempre comprendidos entre 0 y 1, de la forma: **0 <= R < 1;** siendo **R** un número real comprendido en el intervalo, que puede llegar a adoptar el valor mínimo (0) pero no el máximo (1).

Sin embargo, en la mayoría de los casos, debemos elegir un número entre 1 y 90 (bingo), entre 1 y 6 (tirada de un dado), entre 2 y 12 (tirada de dos dados), entre 1 y 40 (extracción de una carta de la baraja española), etc... ¿Cómo solucionamos entonces el problema?

Puesto que el BASIC nos suministra un número decimal cada vez que recurrimos a la generación aleatoria, nos es necesario introducir algunos cálculos para conseguir que la secuencia de números se produzca entre determinados valores, generalmente dentro del conjunto de los naturales (positivos no decimales, incluyendo el cero).

Ahora que ya disponemos de los conocimientos básicos necesarios, entraremos en el estudio de la forma en que el BASIC efectúa la extracción de números aleatorios, y más concretamente, en cómo introducir el factor suerte en nuestros propios programas.

LA FUNCION RND

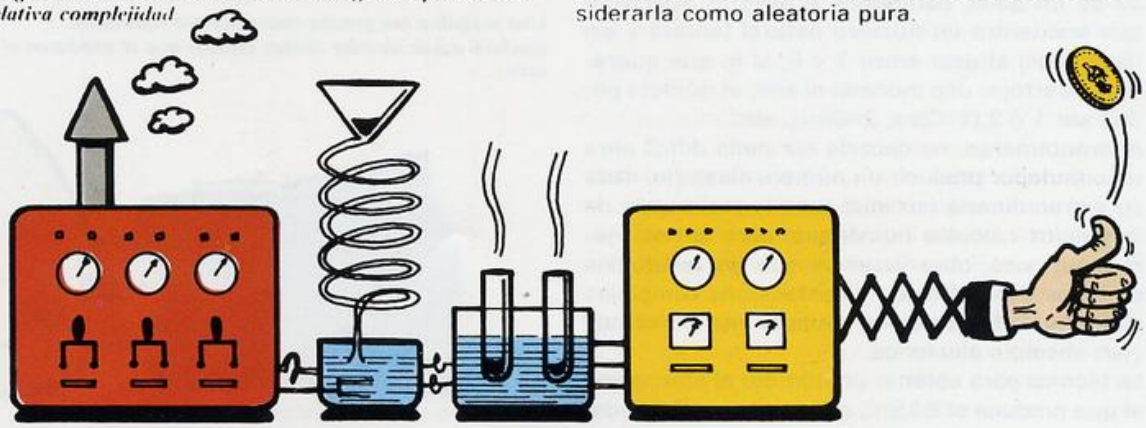
La función BASIC encargada de la generación de números aleatorios es **RND,** y carece de argumentos. Basta con recurrir a ella por medio de la sentencia **LET,** si queremos almacenar su resultado, o con **PRINT,** si lo que deseamos es imprimir directamente en la pantalla. Aunque estos dos sistemas citados son los más habituales, es obvio que puede ser utilizada como cualquier función BASIC.

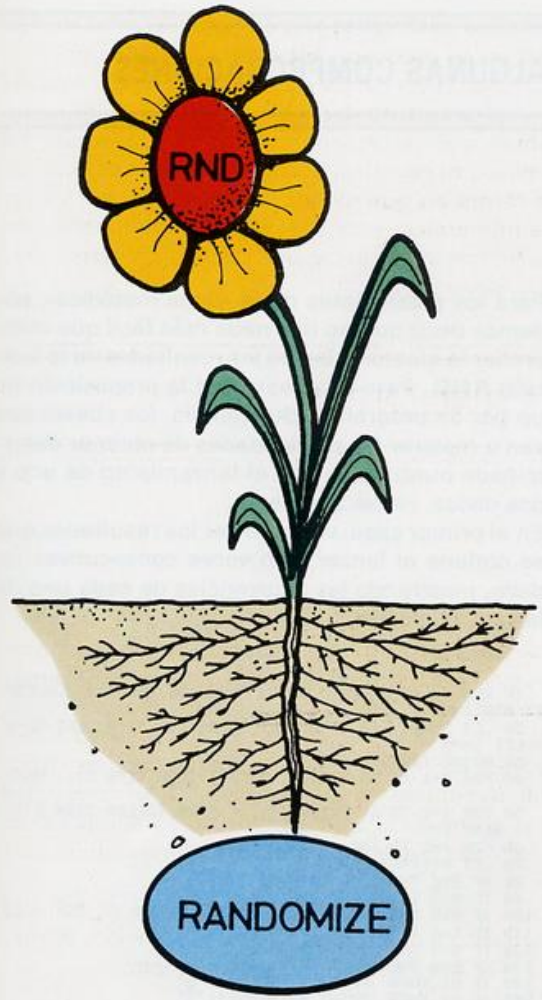
Pero, para comprender gráficamente el aspecto de los números que produce esta función, es mejor que pasemos al siguiente ejemplo:

```
10 REM - GENERACION ALEATORIA
20 FOR I=1 TO 44
30 PRINT RND,;
40 NEXT I
```

Con este programa, llenamos la pantalla de una verdadera sopa de «números», en lugar de la tradicional sopa de «letras». Como ejercicio mental, podemos plantearnos el buscar alguna relación lógica entre los resultados obtenidos, aunque nuestro esfuerzo será en vano. La serie es lo suficientemente complicada para que podamos considerarla como aleatoria pura.

La generación de números aleatorios sigue un proceso de relativa complejidad





La sentencia RANDOMIZE *sitúa la base (semilla) de la que se parte para la extracción de los números aleatorios.*

Ahora vamos a introducir algunas modificaciones en el programa, para que el ámbito de aplicación (desde cero hasta casi uno) se amplíe de acuerdo con nuestras necesidades:

```
10 REM - GENERACION CON LIMITE SUPERIOR
20 INPUT "Limite Superior:";B
30 FOR I=1 TO 44
40 PRINT RND*B,;
50 NEXT I
```

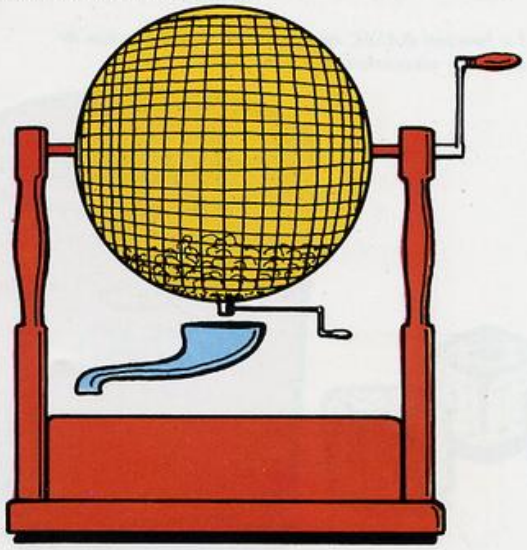
En primer lugar, en la línea 20, pedimos el límite superior de generación de números. En la línea 40, multiplicamos el número aleatorio generado, por el límite superior introducido por nosotros. De esta forma, obtenemos un número que oscila entre cero y el límite superior deseado, aunque sin alcanzarlo en ningún momento.
Como casos especiales, diremos que si introducimos cero como límite superior, todos los números generados serán cero; o si introducimos uno, nos encontraremos ante el mismo caso del programa ejemplo anterior, es decir, la generación aleatoria sin ningún tipo de conversión (intervalo $0 <= R < 1$).
Si seleccionamos un número cualquiera, por ejemplo 10, encontraremos que las partes enteras de los valores obtenidos, oscilan entre 0 y 9. Este hecho puede darnos una pista de cómo obtener números naturales, a partir de las posibilidades de la función generadora de números aleatorios, y en combinación con una función que vimos anteriormente: **INT**, que nos permite convertir los números decimales en enteros.
Vamos a completar ahora el programa haciendo uso de la función **INT** *(INTeger)*, que acabamos de mencionar:

```
10 REM - GENERACION NATURAL CON LIMITE SUPERIOR
20 INPUT "Limite Superior:";B
30 FOR I=1 TO 44
40 PRINT INT(RND*B),;
50 NEXT I
```

Si seleccionamos once como límite superior, veremos que se obtiene una serie de números, que oscilan entre cero y el número indicado por nosotros como límite superior menos uno, es decir, de cero a diez. Más de una vez hemos llegado a pensar que algún profesor ponía las calificaciones por «lotería», y quizá sea este un claro ejemplo de que no andábamos demasiado descaminados.
Por último, podemos incluir un límite inferior a la serie de aleatorios. Puesto que ésta parte siempre de cero, vamos a sumar el número que deseemos como límite inferior, obteniendo la fórmula general para el cálculo de números naturales aleatorios entre límites:

```
PRINT A+INT(RND*B)
```

En el «bombo» de la generación de números aleatorios, entran 65536 bolas (semillas).



¡!

Para la generación de números aleatorios, se parte de un número base (semilla), el cual se somete a un complejo proceso de transformación.

*

Los números aleatorios generados por **RND** se encuentran en el intervalo cero a uno, pudiendo alcanzar el primer valor, pero nunca el último ($0<=R<1$).

*

Para utilizar una semilla determinada en la obtención de **RND**, se utiliza la sentencia **RANDOMIZE** seguida del número que se desea emplear.

¡!

Ejecutando **RANDOMIZE 0**, o simplemente **RANDOMIZE**, la semilla que se utiliza para los números aleatorios, se obtiene a partir del tiempo que la máquina lleva conectada.

*

Para la generación de números aleatorios se emplea la función **RND**, que carece de argumento.

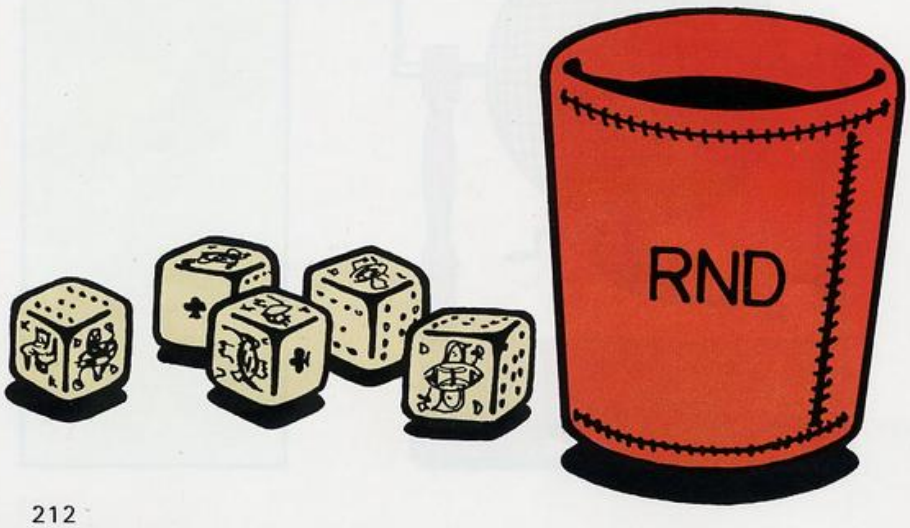
Los números aleatorios generados por el Spectrum son decimales que nunca alcanzan el valor uno, aunque sí el número cero.

Siendo **A** el límite inferior y **B** la diferencia entre límite superior e inferior. Podemos hacer una prueba de esto con el siguiente programa:

```
10 REM - GENERACION NATURAL ENTRE LIMITES
20 INPUT "Limite Inferior:";A
30 INPUT "Limite Sup.-Inf.:";B
40 FOR I=1 TO 44
50 PRINT A+INT(RND*B),;
60 NEXT I
```

Conviene que no pasemos por alto la importancia de la fórmula que acabamos de ver, puesto que ella nos permitirá resolver los problemas planteados en la simulación de la mayoría de los juegos de azar: loterías, cartas, aventuras, etc...

La función BASIC encargada de la generación de números aleatorios es RND.



ALGUNAS COMPROBACIONES

Para los practicantes de la «duda metódica», podemos decir que no hay nada más fácil que comprobar la aleatoriedad de los resultados de la función **RND**. Para ello basta con la proposición de un par de programas de ejemplo, los cuales nos van a mostrar las posibilidades de obtener determinada puntuación con el lanzamiento de uno y dos dados, respectivamente.
En el primer caso, vamos a ver los resultados que se obtiene al lanzar cien veces consecutivas un dado, mostrando las ocurrencias de cada una de las posibilidades:

```
  10 REM - LANZAMIENTO DE UN DADO 100 VECES - J.M.LOPEZ MARTINEZ
  20 LET S1=0: LET S2=0: LET S3=0: LET S4=0: LET S5=0: LET S6=0
  30 PRINT TAB 2;
  40 FOR I=1 TO 30: PRINT (STR$ I)(LEN STR$ I);: NEXT I
  50 FOR I=1 TO 6: PRINT ''(' '+STR$ I)(LEN STR$ I TO ): NEXT I
  60 FOR I=1 TO 100
  70 LET R=1+INT (RND*6)
  80 IF R=1 THEN  LET S1=S1+1
  90 IF R=2 THEN  LET S2=S2+1
 100 IF R=3 THEN  LET S3=S3+1
 110 IF R=4 THEN  LET S4=S4+1
 120 IF R=5 THEN  LET S5=S5+1
 130 IF R=6 THEN  LET S6=S6+1
 140 IF S1 THEN  PRINT AT 2,S1+1;"3"
 150 IF S2 THEN  PRINT AT 5,S2+1;"3"
 160 IF S3 THEN  PRINT AT 8,S3+1;"3"
 170 IF S4 THEN  PRINT AT 11,S4+1;"3"
 180 IF S5 THEN  PRINT AT 14,S5+1;"3"
 190 IF S6 THEN  PRINT AT 17,S6+1;"3"
 200 NEXT I
```

Estudiemos rápidamente el programa: en la línea 20, se asigna el valor cero a los contadores de sucesos de las seis posibilidades. En las líneas 30 a 50, se construye la regla de la parte superior de la pantalla, para medir las ocurrencias, que indica la numeración desde 1 hasta 30. En la línea 60, se establece un bucle de 100 tiradas. En la línea 70, se simula la tirada del dado **R**. En las líneas 80 a 130, se suman la ocurrencias. En las líneas 140 a 190, se imprime, en la fila y columna correspondientes, la valoración de las ocurrencias y, por último, en la línea 200, se cierra el bucle de 100 tiradas.
La técnica para simular el diagrama de barras, que es como se conoce a este tipo de gráfico, es bien sencilla. Si nos fijamos un poco en el programa, veremos que, después de contabilizar las ocurrencias, se imprimen mediante sentencias **PRINT AT**, los caracteres que se obtienen por pulsación de **SHIFT** y **3**, con el cursor en el modo gráfico **G**.
Los números de fila (primer parámetro de **AT**) son

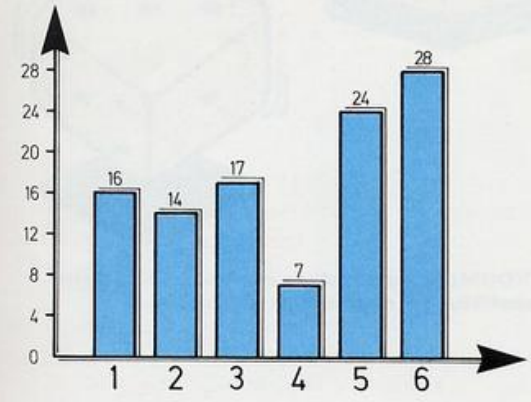
La función **INT** *colabora con* **RND** *en la simulación de ciertos sucesos aleatorios.*

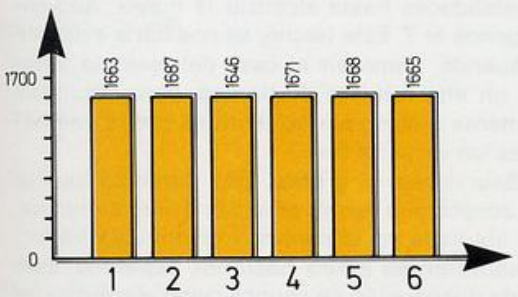
fijos, y corresponden a las líneas en que hemos situado los números del 1 al 6, en el margen izquierdo de la pantalla, para señalar qué suceso representa cada barra.

Los números de columna (segundo parámetro de **AT**), se calculan como el número de ocurrencias más uno. Dado que cada instrucción de este tipo la comenzamos con la comparación **IF S*n***, siendo **n** de 1 a 6, sólo se produce la impresión cuando **S*n*** es mayor que cero. De este modo, como sumamos uno, conseguimos que las impresiones se produzcan desde la columna tercera de la pantalla **(TAB 2)** hasta la columna 32 **(TAB 31)**, si llega a alcanzarse esta posición, consiguiendo el efecto de una barra continua.

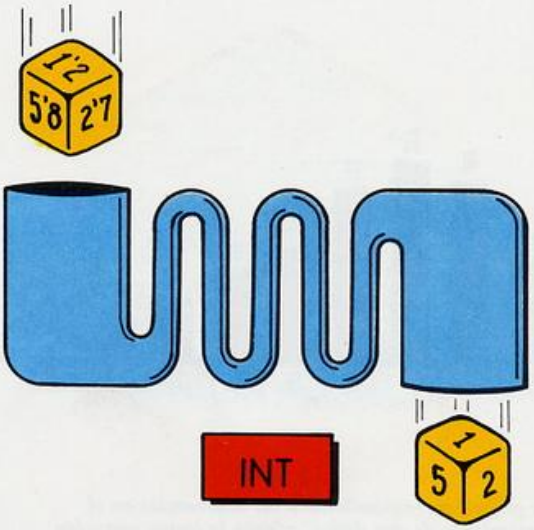
Observaremos que con cada nueva ejecución del programa por medio de la sentencia **RUN**, se obtienen proporciones diferentes de cada una de las posibilidades. Esto sucede porque aplicamos un número bajo de lanzamientos; puesto que, estadísticamente, es idéntica la posibilidad de salir cada una de las caras del dado hacia arriba, es de esperar que, con un número mucho mayor de lanzamientos, las longitudes de las barras se vayan igualando ante nuestros propios ojos.

Vamos a ver lo que sucedería elevando el número de ocurrencias hasta diez mil, aunque a base de armarnos de paciencia, porque el programa tardará en ejecutarse unos veinticinco minutos.





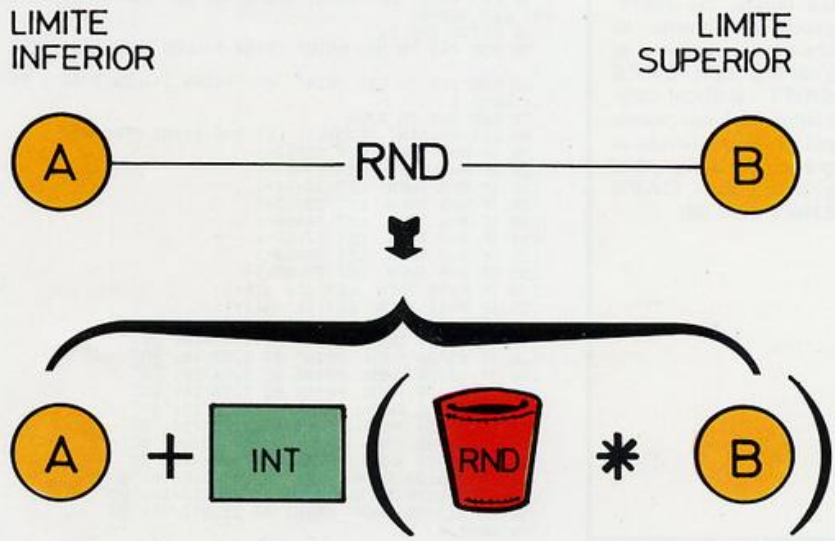
Gráficos obtenidos por el lanzamiento de un dado 1679 y 10000 veces.



No obstante, nuestro afán investigador se verá recompensado al comprobar, de una forma experimental, que existen casi las mismas probabilidades para cada una de las ocurrencias (la longitud de las barras es similar), considerando un número alto de tiradas:

```
  10 REM - LANZAMIENTO DE UN DADO 10000 VECES - J.M.L
OPEZ MARTINEZ
  20 LET S1=0: LET S2=0: LET S3=0: LET S4=0: LET S5=0
: LET S6=0
  30 PRINT TAB 2;
  40 FOR I=1 TO 30: PRINT (STR$ I)(LEN STR$ I);: NEXT
 I
  50 FOR I=1 TO 6: PRINT ''(' '+STR$ I)(LEN STR$ I TO
 ): NEXT I
  60 FOR I=1 TO 10000
  70 LET R=1+INT (RND*6)
  80 IF R=1 THEN  LET S1=S1+1
  90 IF R=2 THEN  LET S2=S2+1
 100 IF R=3 THEN  LET S3=S3+1
 110 IF R=4 THEN  LET S4=S4+1
 120 IF R=5 THEN  LET S5=S5+1
 130 IF R=6 THEN  LET S6=S6+1
 140 IF S1>100 THEN  PRINT AT 2,S1/100+1;"■"
 150 IF S2>100 THEN  PRINT AT 5,S2/100+1;"■"
 160 IF S3>100 THEN  PRINT AT 8,S3/100+1;"■"
 170 IF S4>100 THEN  PRINT AT 11,S4/100+1;"■"
 180 IF S5>100 THEN  PRINT AT 14,S5/100+1;"■"
 190 IF S6>100 THEN  PRINT AT 17,S6/100+1;"■"
 200 NEXT I
```

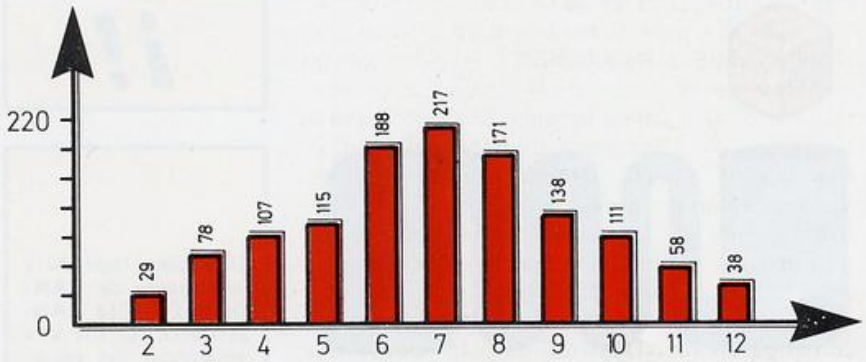
El algoritmo para la obtención de un natural entre límites es muy importante: **R=A+INT (RND*B).**



¡!

El empleo repetido y continuado de **RANDOMIZE 0** (o **RANDOMIZE**), en vez aleatorizar los resultados de **RND**, produce efectos desastrosos de repetición de resultados.

La forma de empleo de **RND** es similar a la de cualquier otra función BASIC.



La gráfica correspondiente a las ocurrencias en el lanzamiento de dos dados, adopta la forma conocida como «campana de Gauss».

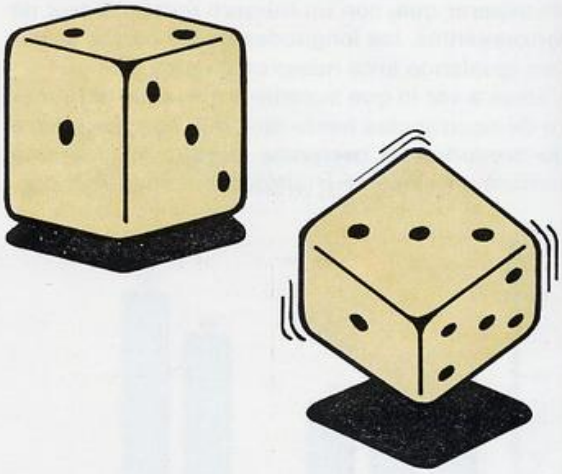
Las modificaciones que hemos introducido en el programa no son muchas. En primer lugar, el número de lanzamientos es de diez mil, y por tanto se ha alterado el **FOR** de la línea 60. Por otra parte, el número de ocurrencias es dividido por cien antes de su presentación en la pantalla; al ser esta proporcional, cada una de las posiciones de la regla de la parte superior de la pantalla, representa cien ocurrencias en lugar de una.
Comenzamos ahora con el segundo de los experimentos propuestos: el caso del lanzamiento de dos dados. Esta vez, la tirada mínima es de dos y la máxima de doce, pero no son iguales las probabilidades de que aparezca cada una de ellas. Pasaremos primero a la práctica y después comentaremos los resultados:

¡!

Los números subrayados que se emplean en los listados de ejemplo, corresponden a los gráficos de las teclas; los subrayados doblemente, se obtienen mediante el gráfico con **CAPS SHIFT** (gráfico cambiado). Así por ejemplo, 3, corresponde al gráfico del 3 (▬); y 8 al gráfico con **CAPS SHIFT** de 8 (■).

```
  10 REM - LANZAMIENTO DE DOS DADO 1250 VECES - J.M.L
OPEZ MARTINEZ
  20 LET S2=0: LET S3=0: LET S4=0: LET S5=0: LET S6=0
  30 LET S7=0: LET S8=0: LET S9=0: LET S10=0: LET S11
=0: LET S12=0
  40 PRINT TAB 2;
  50 FOR I=1 TO 30: PRINT (STR$ I)(LEN STR$ I);: NEXT
I
  60 FOR I=2 TO 12: PRINT "(" "+STR$ I)(LEN STR$ I TO
): NEXT I
  70 FOR I=1 TO 1250
  80 LET R=1+INT (RND*6): LET R=R+1+INT (RND*6)
  90 IF R=2 THEN  LET S2=S2+1
100 IF R=3 THEN  LET S3=S3+1
110 IF R=4 THEN  LET S4=S4+1
120 IF R=5 THEN  LET S5=S5+1
130 IF R=6 THEN  LET S6=S6+1
131 IF R=7 THEN  LET S7=S7+1
132 IF R=8 THEN  LET S8=S8+1
133 IF R=9 THEN  LET S9=S9+1
134 IF R=10 THEN  LET S10=S10+1
135 IF R=11 THEN  LET S11=S11+1
136 IF R=12 THEN  LET S12=S12+1
150 IF S2>20 THEN  PRINT AT 1,S2/10;"▬"
160 IF S3>20 THEN  PRINT AT 3,S3/10;"▬"
170 IF S4>20 THEN  PRINT AT 5,S4/10;"▬"
180 IF S5>20 THEN  PRINT AT 7,S5/10;"▬"
190 IF S6>20 THEN  PRINT AT 9,S6/10;"▬"
200 IF S7>20 THEN  PRINT AT 11,S7/10;"▬"
210 IF S8>20 THEN  PRINT AT 13,S8/10;"▬"
220 IF S9>20 THEN  PRINT AT 15,S9/10;"▬"
230 IF S10>20 THEN  PRINT AT 17,S10/10;"▬"
240 IF S11>20 THEN  PRINT AT 19,S11/10;"▬"
250 IF S12>20 THEN  PRINT AT 21,S12/10;"▬"
310 NEXT I
```

Puesto que la técnica de programación es similar a la de los programas ejemplos anteriores, no merece la pena que recalquemos nada especialmente sobre su estructura. Sólo diremos que cada posición de la escala equivale a diez ocurrencias, y que a la cabeza de las barras hemos añadido una marca del valor máximo.
Sí es interesante observar que, esta vez, se aprecian sensibles diferencias entre las ocurrencias de cada uno de los sucesos, sobre todo entre las barras centrales y las situadas en los extremos superior e inferior, respectivamente.
Esto es debido a que las probabilidades no son realmente las mismas. De hecho, existe una sola posibilidad de obtener en una tirada tanto el 2 como el 12, que se corresponde con el 1 y el 6 de los dos dados, respectivamente.
En cambio, la probabilidad de obtener una puntuación de 3 o de 11, es exactamente el doble, ya que existen dos combinaciones válidas para cada caso: para la puntuación primera, el 1 en el primer dado y el 2 en el segundo, o el 2 en el primer dado y el 1 en el segundo. De forma análoga, se explica la doble probabilidad de obtener la puntuación de 11



RANDOMIZE *seguido de un numero, se utiliza para desaleatorizar los resultados de* **RND**.

De esta manera progresiva, van aumentando las probabilidades hasta alcanzar la mayor, que corresponde al 7. Este hecho, se nos haría evidente efectuando, como en el caso del ejemplo anterior, un alto número de tiradas, pero resultaría realmente tedioso por su lentitud, por lo que haremos un «acto de fe».
Si observamos el gráfico, nos daremos cuenta que adopta una forma parecida a una campana, más abultada en el centro, y disminuida uniformemente en su altura hacia los extremos. Este tipo de distribución se conoce como «campana de Gauss», en honor al prestigioso matemático que investigó sobre las leyes de la probabilidad.

RANDOMIZE

La sentencia **RANDOMIZE** (en abreviatura **RAND**) tiene por objetivo hacer comenzar la generación de aleatorios de la función **RND**, en un punto preciso de la secuencia de números semilla, es decir, sentar el origen para esta función. Si se ejecuta una sentencia **RANDOMIZE 1** seguida de varios **RND**, se obtiene una serie de valores aleatorios; pero si se vuelve a ejecutar de nuevo **RANDOMIZE 1**, los valores calculados a continuación por dicha función, serán idénticos a los calculados anteriormente, es decir, nada aleatorios:

```
10 REM - REPETICION DE SECUENCIAS
20 RANDOMIZE 1
30 FOR I=0 TO 21:PRINT AT I,0;RND:NEXT I
40 RANDOMIZE 1
50 FOR I=0 TO 21:PRINT AT I,16;RND:
NEXT I
```

Existe una segunda forma de utilizar **RANDOMIZE**, esta vez para conseguir todo lo contrario. **RANDOMIZE** o **RANDOMIZE 0**, provocan un cambio en el elemento de la secuencia de aleatorios a utilizar por la función **RND**, en base al reloj interno del Spectrum. En pocas palabras, sitúan la semilla según el tiempo transcurrido desde la conexión de la máquina.

Esto es muy útil cuando se trata de obtener un número realmente aleatorio, pero desastroso cuando un programa pasa repetidas veces, y en corto espacio de tiempo, por una instrucción que contenga un **RANDOMIZE** de este tipo, ya que su efecto es una repetición grande de los resultados.

Una muestra clara de la aplicación de **RANDOMIZE**, la tenemos en el programa MASTER CUBOS, del que tuvimos oportunidad de disfrutar hace poco. En la línea 1290, que encabeza la presentación del juego, se ejecuta una instrucción **RANDOMIZE 2**; a partir de allí, los colores para los cubos que formarán en la pantalla la palabra gigante CUB, se obtienen mediante la función **RND**; a ello se debe que estos sean siempre del mismo color. Si no hubiéramos empleado este sistema, nos habríamos visto obligados a malgastar memoria y variables para almacenar los colores correspondientes a los cubos.

ALGUNOS EJERCICIOS

Con lo dicho hasta el momento, tenemos la base suficiente para obtener y manejar la generación de números aleatorios. Así pues, vamos a hacer algunos programas a modo de ejercicio:

1. Extracción de una carta de la baraja española (40 cartas).

```
10 REM BARAJA ESPA\OLA
20 REM R=NUMERO ALEATORIO
30 REM C=CARTA
40 REM P=PALO
50 LET R=INT (RND*40)
60 LET P=INT (R/10)
70 LET C=R-10*P
80 IF C=7 THEN  LET C$="SOTA": GO TO 130
90 IF C=8 THEN  LET C$="CABALLO": GO TO 130
100 IF C=9 THEN  LET C$="REY": GO TO 130
110 IF C=0 THEN  LET C$="AS": GO TO 130
120 LET C$=STR$ (C+1)
130 IF P=0 THEN  LET P$="OROS": GO TO 160
140 IF P=1 THEN  LET P$="COPAS": GO TO 160
150 IF P=2 THEN  LET P$="ESPADAS": GO TO 160
160 IF P=3 THEN  LET P$="BASTOS"
170 PRINT C$;" DE ";P$
```

2. Lanzamiento de dos dados (dados de 6 caras).

```
10 REM LANZAMIENTO DE DOS DADOS
20 REM D1=DADO 1
30 REM D2=DADO 2
40 LET D1=1+INT (RND*6)
50 LET D2=1+INT (RND*6)
60 PRINT "DADO 1= ";D1,"DADO 2= ";D2
```

3. Moneda arrojada al aire (2 caras).

```
10 REM CARA O CRUZ
20 IF RND<.5 THEN  PRINT "CARA"
30 PRINT "CRUZ"
```

4. Extracción de una carta de la baraja francesa (54 cartas).

```
10 REM BARAJA FRANCESA
20 REM R=NUMERO ALEATORIO
30 REM C=CARTA
40 REM P=PALO
50 LET R=INT (RND*54)
60 IF R>51 THEN  PRINT "COMODIN": GO TO 10000
70 LET P=INT (R/13)
80 LET C=R-13*P
90 IF C=10 THEN  LET C$="JACK": GO TO 140
100 IF C=11 THEN  LET C$="REINA": GO TO 140
110 IF C=12 THEN  LET C$="REY": GO TO 140
120 IF C=0 THEN  LET C$="AS": GO TO 140
130 LET C$=STR$ (C+1)
140 IF P=0 THEN  LET P$="CORAZONES": GO TO 180
150 IF P=1 THEN  LET P$="DIAMANTES": GO TO 180
160 IF P=2 THEN  LET P$="PICAS": GO TO 180
170 IF P=3 THEN  LET P$="TREBOLES"
180 PRINT C$;" DE ";P$
```

5. Estadística de las notas asignadas al azar (de 0 a 10).

```
10 REM ESTADISTICA DE NOTAS AL AZAR
20 REM N=NOTA
30 LET N=INT (RND*11)
40 PRINT "NOTA= ";N
50 GO TO 30
```

6. Extracción de bolas de un bombo de bingo (90 bolas).

```
10 REM BINGO
20 REM B=BOLA EXTRAIDA
30 LET B=1+INT (RND*90)
40 PRINT "BOLA: ";B
```

Al tipo de sucesos que dependen del azar, se les conoce como SUCESOS ALEATORIOS o ESTOCASTICOS.

*

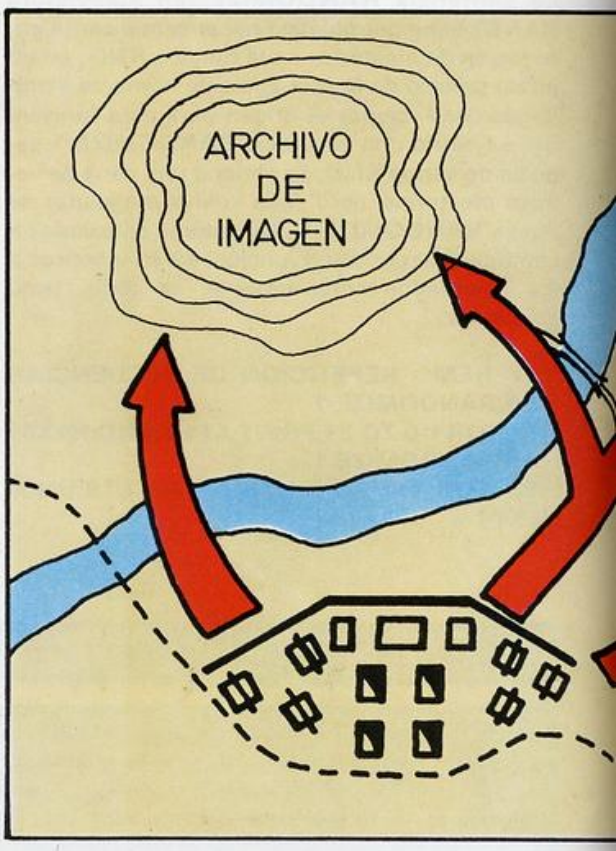
Dos son las sentencias que controlan el proceso de generación de números aleatorios: **RANDOMIZE** (abreviado en el teclado a **RAND**) y **RND**.

A TODA PANTALLA

EN el capítulo anterior, adquirimos los conocimientos necesarios sobre la pantalla del Spectrum, como para permitirnos en las páginas siguientes discutir una serie de técnicas encaminadas al dinamismo de pantallas. Fundamentalmente, nuestra acción se concentrará sobre dos puntos:

* El archivo de imagen.
* El área de atributos.

En una primera fase, nos dedicaremos al tratamiento conjunto de ambas zonas, para más adelante pasar a técnicas que afectan a cada una de ellas separadamente. Nuestro primer movimiento se refiere concretamente a la carga de pantallas previamente almacenadas, y fundamentalmente trata el aspecto de la dinámica inherente a la misma, proponiendo varios sistemas para alterar el proceso normal de carga en el Spectrum. En segundo lugar, concentraremos todas nuestras fuerzas sobre el área de atributos, estudiando una interesante subrutina en código máquina, con la cual poder obtener espectaculares efectos. Por último, nos adentraremos en el archivo de imagen, donde estudiaremos la más pura animación de pantallas, aunque lamentablemente las características de ésta reservan su disfrute a aquellos que poseamos el modelo de 48 K o el Plus, que en lo referente a la memoria, tanto monta, monta tanto.



BITS

Una alternativa al comando **NEW**, la proporciona **RANDOMIZE USR 4572**. Esta llamada a la R.O.M., tendrá el mismo efecto de borrado de la memoria, hasta el límite fijado por RAMTOP. Pero a diferencia de **NEW**, la pantalla no cambiará al negro durante unos instantes.

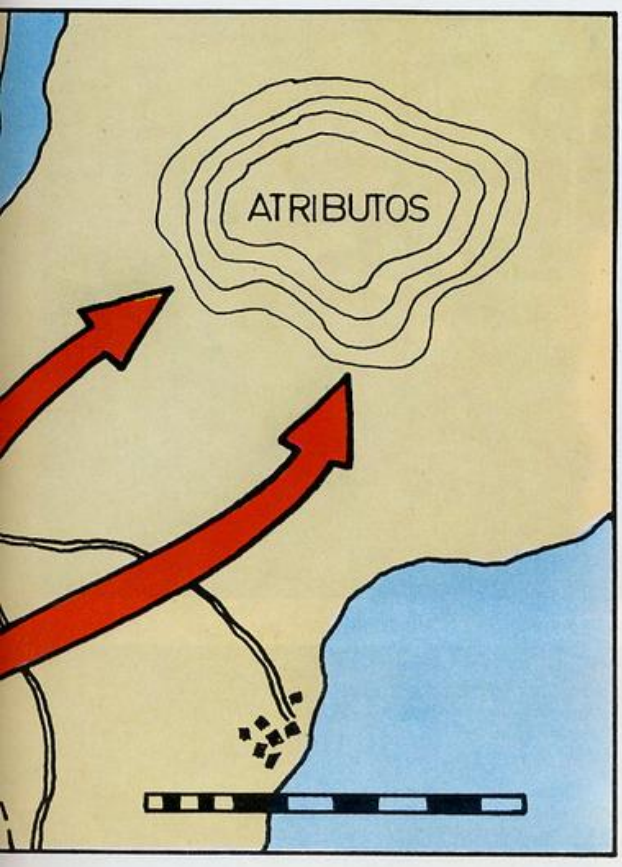
CARGA DE PANTALLAS

Una de las cosas que más llaman la atención al ver cargar un programa en el Spectrum, es la peculiar manera de construir las pantallas que, como ya vimos anteriormente, se debe a la organización un tanto caótica de su archivo de imagen.

El gran problema que se aprecia inmediatamente es que, lógicamente, se carga antes el archivo de imagen que su área de atributos; a causa de ello, el primer aspecto que presenta la pantalla al cargarse, antes de su finalización, no suele ser muy estético. Sería bueno disponer de un sistema que no permitiera ver la pantalla hasta que no se hubieran asignado los colores, evitando así primeras impresiones, generalmente bastante feas. Efectivamente, disponemos de este sistema, y además es aplicable de una manera bien sencilla: sólo hemos de asignar un mismo color de **INK** (tinta) y **PAPER** (papel) antes de la carga de la pantalla. Gracias a este pequeño truco, la pantalla no aparecerá hasta que se inicie la carga del área de atributos. Podemos probar la carga de una pantalla mediante el siguiente programa:

```
10 REM PANTALLA INVISIBLE - C. DE LA OSSA & F. LOPEZ MARTINEZ
20 BORDER 7: PAPER 7: INK 7: BRIGHT 0: FLASH 0: CLS
30 LOAD ""SCREEN$
```

Pero quizás los amantes de la rapidez prefieran algo más efectivo, más rápido aún; para este se-



Para dinamizar pantallas en conjunto, concentraremos nuestra acción en dos puntos: el archivo de imagen y el área de atributos.

gundo truco, necesitaremos el auxilio del código máquina. Nos basaremos en una instrucción LDIR (instrucción de ensamblador, pero no de BASIC), que permite la transferencia de bloques de memoria; de esta forma, podremos cargar la pantalla en una zona de memoria, que no sea la suya propia, y luego transferirla a la zona de memoria de pantalla de manera casi instantánea. El siguiente programa realiza esta función:

```
10 REM PANTALLA INSTANTANEA - C. DE
LA OSSA & F. LOPEZ MARTINEZ
20 CLEAR 25687
30 FOR I=0 TO 11
40 READ A: POKE USR "A"+I,A
50 NEXT I
60 DATA 33,88,100,17,0,64,1,0,27,237,
176,201
70 LOAD ""CODE 25688
80 RANDOMIZE USR USR "A"
```

Para aquellos que hagamos nuestros pinitos en el código máquina, añadiremos el ensamblador de esta rutina, que podríamos denominar de TRANSFERENCIA DE PANTALLAS:

SUBRUTINA DE TRANSFERENCIA

ETIQUETA	OBJETO	FUENTE	COMENTARIO
TRSF	33 88 100	LD HL,25688	Carga HL con la dirección de origen.
	17 0 64	LD DE,16384	Carga DE con la dirección de destino.
	1 0 27	LD BC,6912	Carga BC con la longitud del bloque a transferir.
	237 176	LDIR	Transfiere el bloque de memoria especificado.
	201	RET	Retorna al BASIC.

Por otra parte, la utilidad de esta rutina es doble, y no sólo se limita a la representación rápida de pantallas; veamos el porqué de esta afirmación. El tiempo que el Spectrum emplea para la carga de una pantalla no es precisamente poco (más de 36 segundos), y resulta un tanto frustrante que tanto esfuerzo desaparezca tras un brevísimo esplendor. Si observamos cuidadosamente el sistema de carga que hemos estudiado unas líneas más arriba, comprobaremos que la pantalla sigue aún situada en el lugar en que fue cargada, puesto que la rutina en código máquina simplemente la copia, pero no la deteriora ni cambia de lugar; por tanto, esta subrutina puede ser también empleada para restituir la imagen inicial en la pantalla siempre que lo consideremos oportuno, por ejemplo, en el cambio de jugadores en un juego de acción *(arcade)* o tras el mensaje de fin de juego.

Así pues, esta rutina que hemos visto nos puede ser de gran utilidad, ya que más que realizar una transferencia de pantallas, efectúa una copia o reproducción. Otra de sus cualidades es la de ser reubicable, lo cual significa que cualquiera que

El sistema de carga de pantallas en el Spectrum, permite ver el archivo de imagen antes de la asignación de color.

BITS

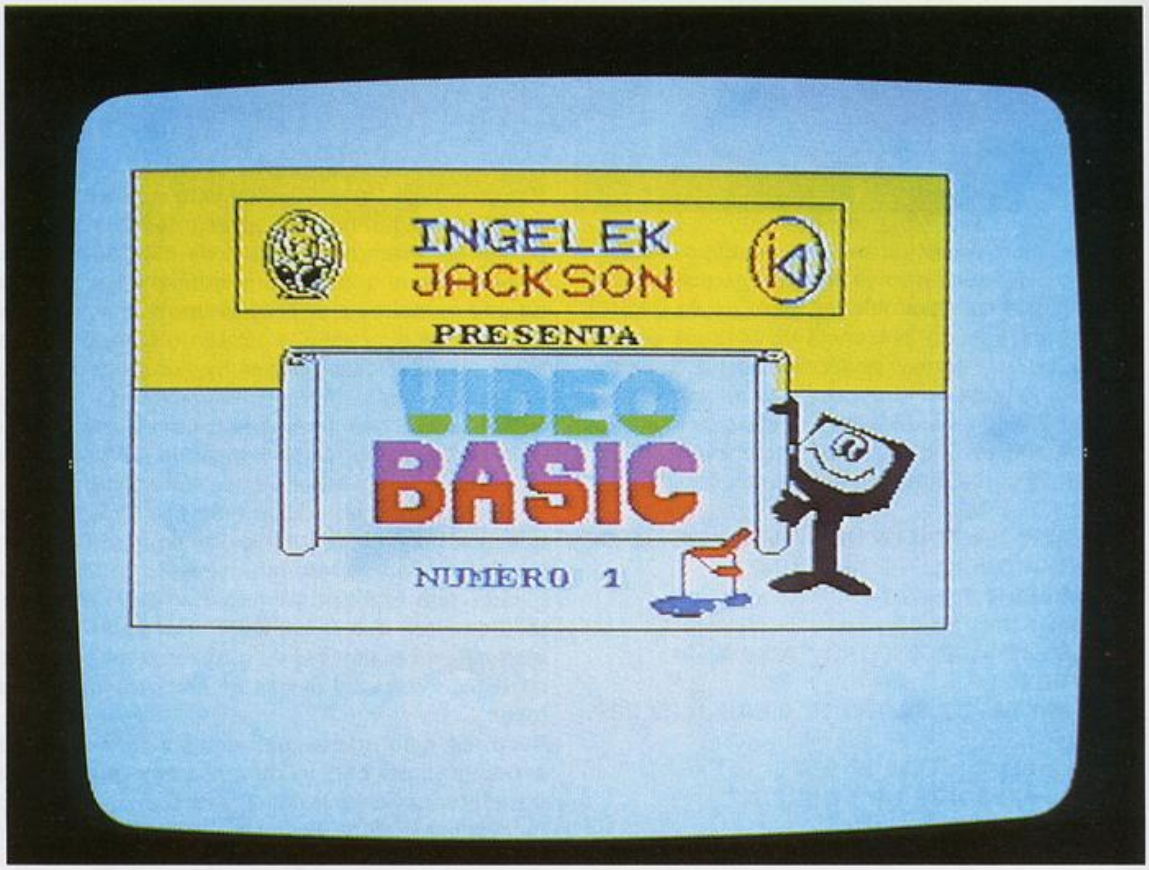
Las últimas hojas del manual de nuestro Spectrum, incluyen una lista de los posibles informes que el ordenador puede visualizar en la parte inferior de la pantalla, durante la ejecución de un programa. Probemos el efecto de las siguientes líneas:

```
10 POKE 23296,207
20 FOR N=0 TO 26
30 POKE 23297,N
40 PRINT USR 23296
50 NEXT N
```

Cada vez que obtengas un mensaje de error, con **GO TO 50** y **ENTER**, aparecerá uno nuevo en la parte inferior de la pantalla, hasta completar el programa con el conocido **0 OK**. Después de esto, ya no podremos decir que existe algún error que nunca hemos visto.

*

Cuando una estructura de bucle incluye a otro o varios se denomina «anidamiento».

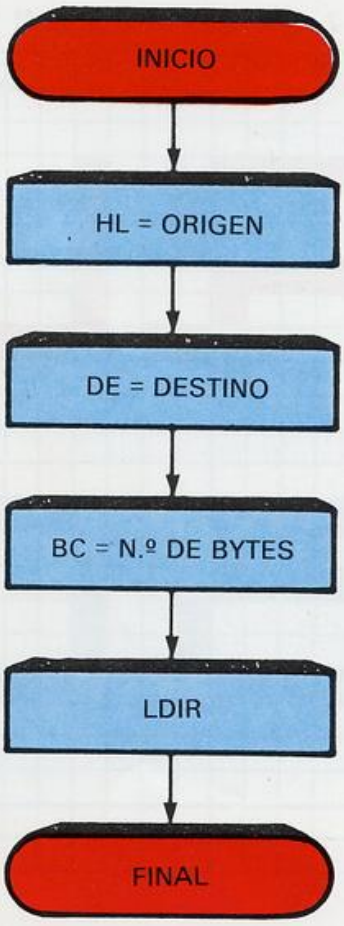


Mediante un truco bien simple, podemos conseguir que la pantalla no se reconozca hasta que no se haya cargado el área de atributos.

sea el lugar de la memoria donde se encuentre funcionará a la perfección. En nuestro caso, pensando en que el programa de ejemplo pudiera ser ejecutado en un modelo de 16 K o 48 K indistintamente, se ha empleado el área de gráficos definidos, concretamente a partir de la A, para su almacenamiento; sin embargo, el mismo efecto habríamos obtenido cargándola, por ejemplo, a partir de 25000 (decimal), y alterando por supuesto la llamada correspondiente de la línea 80 (**80 RANDOMIZE USR 25000**).

Cuando hablemos de la animación del archivo de imagen, tendremos ocasión de comentar más ampliamente, y aprender a utilizar en nuestros propios programas, esta interesante subrutina; sin embargo, sobre este último sistema de carga de pantallas que se apoya en ella, hemos de añadir que ocupa una gran cantidad de memoria: principalmente los casi 7 K (6912 bytes) de la pantalla almacenada (los 12 bytes de subrutina son insignificantes), por tanto, es muy probable que los usuarios del modelo de 16 K no podamos permitirnos este despilfarro de memoria.

```
  10 REM RUN-SOFT - C.DE LA OSSA - F.LOPEZ MARTINEZ
  20 BORDER 0: PAPER 0: INK 2: CLS
  30 PRINT AT 3,1; FLASH 1;"88888"; FLASH 0;"    "; FL
ASH 1;"8"; FLASH 0;"    "; FLASH 1;"8"; FLASH 0;"  ";
 FLASH 1;"8"; FLASH 0;"    "; FLASH 1;"8"
  40 PRINT AT 4,1; FLASH 1;"8"; FLASH 0;"    "; FLASH
 1;"8"; FLASH 0;"  "; FLASH 1;"8"; FLASH 0;"    "; FL
ASH 1;"8"; FLASH 0;"  "; FLASH 1;"88"; FLASH 0;"   ";
 FLASH 1;"8"
  50 PRINT AT 5,1; FLASH 1;"8"; FLASH 0;"    "; FLASH
 1;"8"; FLASH 0;"  "; FLASH 1;"8"; FLASH 0;"    "; FL
ASH 1;"8"; FLASH 0;"  "; FLASH 1;"8"; FLASH 0;" "; FL
ASH 1;"8"; FLASH 0;"  "; FLASH 1;"8"
  60 PRINT AT 6,1; FLASH 1;"88888"; FLASH 0;"   "; FL
ASH 1;"8"; FLASH 0;"    "; FLASH 1;"8"; FLASH 0;"  ";
 FLASH 1;"8"; FLASH 0;"  "; FLASH 1;"8"; FLASH 0;" ";
 FLASH 1;"8"; FLASH 0;"  "; FLASH 1;"888888"
  70 PRINT AT 7,1; FLASH 1;"8"; FLASH 0;"    "; FLASH
 1;"8"; FLASH 0;"    "; FLASH 1;"8"; FLASH 0;"    "; FL
ASH 1;"8"; FLASH 0;"  "; FLASH 1;"8"; FLASH 0;"   ";
FLASH 1;"88"
  80 PRINT AT 8,1; FLASH 1;"8"; FLASH 0;"    "; FLASH
 1;"8"; FLASH 0;"    "; FLASH 1;"8888"; FLASH 0;"   ";
 FLASH 1;"8"; FLASH 0;"    "; FLASH 1;"8"
  90 PAPER 0: INK 5
 100 PRINT AT 13,2; FLASH 1;"    "; FLASH 0;"    "; F
LASH 1;"    "; FLASH 0;"  "; FLASH 1;"      "; FLASH
 0;" "; FLASH 1;"  "
 110 PRINT AT 14,1; FLASH 1;" "; FLASH 0;"       "; F
LASH 1;" "; FLASH 0;"    "; FLASH 1;" "; FLASH 0;"  "
; FLASH 1;" "; FLASH 0;"          "; FLASH 1;" "
 120 PRINT AT 15,2; FLASH 1;"    "; FLASH 0;"   "; FL
ASH 1;" "; FLASH 0;"    "; FLASH 1;" "; FLASH 0;" ";
 FLASH 1;"     "; FLASH 0;"     "; FLASH 1;" "
 130 PRINT AT 16,6; FLASH 1;" "; FLASH 0;" "; FLASH
1;" "; FLASH 0;"    "; FLASH 1;" "; FLASH 0;"  "; FLA
SH 1;" "; FLASH 0;"          "; FLASH 1;" "
 140 PRINT AT 17,6; FLASH 1;" "; FLASH 0;" "; FLASH
1;" "; FLASH 0;"    "; FLASH 1;" "; FLASH 0;"  "; FLA
SH 1;" "; FLASH 0;"          "; FLASH 1;" "
 150 PRINT AT 18,2; FLASH 1;"    "; FLASH 0;"    "; F
LASH 1;"    "; FLASH 0;"    "; FLASH 1;" "; FLASH 0;"
      "; FLASH 1;" "
 160 FOR I=0 TO 56
 170 READ A,B
 180 BEEP A,B
 190 NEXT I
 200 DATA .5,5,.5,4,.5,2,.5,0,.2,9,.08,9,.08,9,.2,9
 210 DATA .25,0,.7,9,.3,-59,.5,5,.5,4,.5,5,.5,2,.2,10
,.08,10
 220 DATA .08,10,.2,10,.25,2,.7,10,.3,-59,.35,7,.1,6,
.35,7,.1,6
 230 DATA .35,7,.1,6,.20,7,.2,10,.2,9,.08,9,.08,7,.1,
9,.1,-59
 240 DATA .1,10,.1,-59,.2,12,.2,-1,.2,0,.2,1,.2,2,.08
,2,.08,1
 250 DATA .2,2,.25,5,.2,4,.1,4,.1,3,.1,4,.1,-59,.2,7,
.3,5
 260 DATA .2,-59,.3,0,.2,-59,.3,5
 270 PAUSE 250
 280 RUN
```



Organigrama de la subrutina de transferencia.

INTERMITENCIA

Para terminar con esta primera parte de la animación a toda pantalla, entraremos de lleno en el tema que nos ocupa: el dinamismo. El ordenador no es capaz de ejecutar dos programas a un mismo tiempo, sin embargo, asistiremos con asombro a la interpretación de una musiquilla mientras la pantalla no deja de «bailar». ¿Magia? No, un simple truco basado en el modo **FLASH**; como sabemos, el **FLASH** o intermitencia es una tarea llevada a cabo de forma automática por el *hardware* del ordenador, y por tanto siempre puede actuar, aunque se estén realizando otras funciones, como por ejemplo, la carga de programas. Pasemos a la práctica.

El truco consiste en la creación de dos mensajes gigantes en la pantalla, que escribiremos me-

¿?

RUNEAR: Ejecutar un programa mediante el comando **RUN**.

*

POKEAR: Ejecutar una sentencia **POKE**.

*

FORMATEAR: Someter un cartucho a una operación **FORMAT**.

*

LEER: Cargar un programa o datos desde la cinta. Obtener algún valor mediante una función.

*

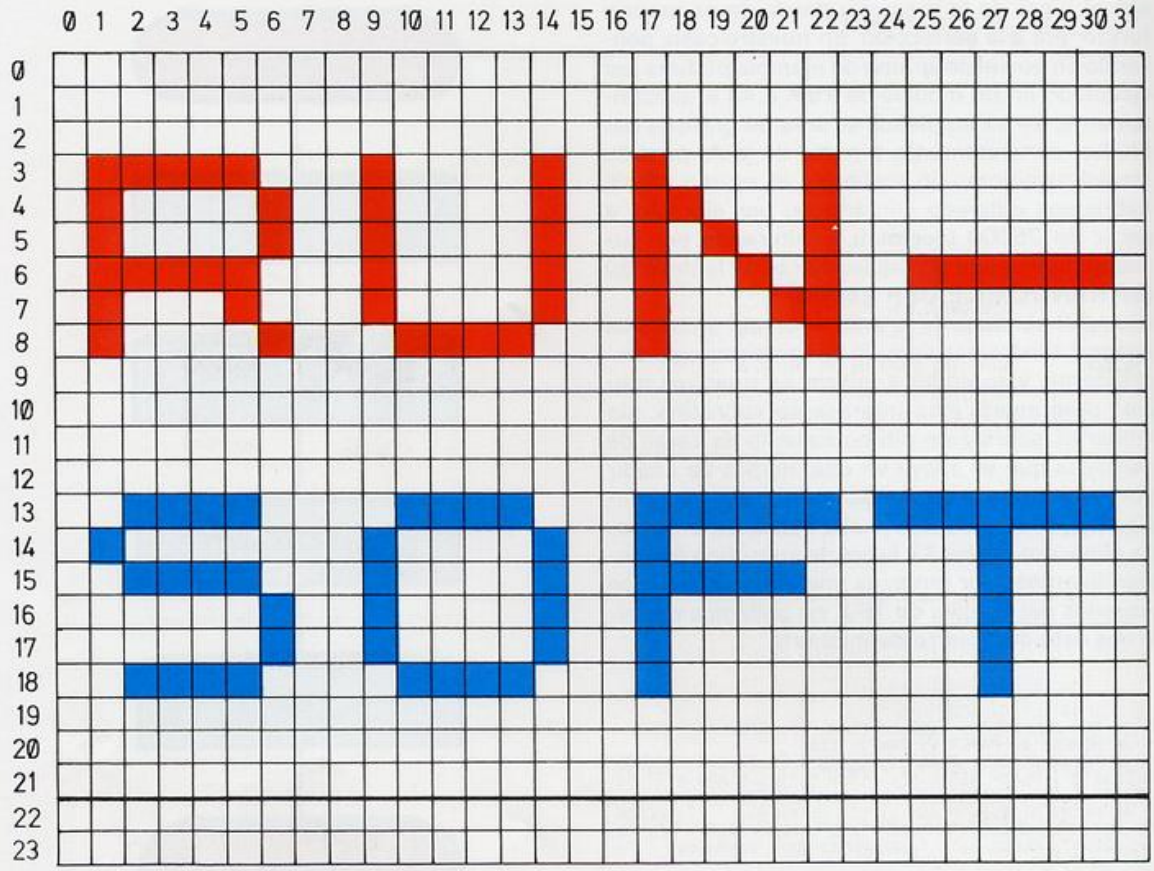
MERGEAR: Mezclar un programa con el de la memoria mediante el comando **MERGE**.

*

PLOTEAR: Ejecutar una sentencia **PLOT**.

*

PRINTEAR: Escribir en la pantalla mediante la sentencia **PRINT**.



Disposición de los caracteres en la pantalla, para el programa de demostración de dinamismo por intermitencia.

¿?

diante cuadrados macizos de la dimensión de un carácter, los cuales podremos conseguirlos a partir del gráfico cambiado del 8; es decir, **CAPS SHIFT + 8**, con el cursor en modo **GRAPHICS (CAPS SHIFT + 9)**. La posición en que se escriban los caracteres, no es en este caso lo más importante, sino la sincronización del modo **FLASH** y los colores correspondientes, para que se produzca un efecto de movimiento por intermitencia. Para acabar de perfilar la impresión óptica, haremos que el sistema de «puesta en escena» de los mensajes, sea contrario en cuanto a códigos de tinta y fondo, gracias a lo cual, cuando un mensaje se presente apagado, el otro aparecerá encendido y viceversa. Lógicamente, para conseguir el efecto de desaparición del mensaje, deberemos utilizar el mismo color de fondo para éste que para el resto de la pantalla.

Llevando aún más lejos este efecto óptico, podemos entremezclar dos mensajes, haciéndolos aparecer en una misma zona de la pantalla, aunque en este caso la coordinación de colores será más complicada, puesto que el color de «apagado» ya no corresponderá siempre con el del fondo de la pantalla, sino con el de «encendido» del otro mensaje. Veamos un ejemplo práctico con dos únicos caracteres, que sirven de nombre de pila a nuestro aparato: ZX.

```
  10 REM Z-X - C.DE LA OSSA - F.LOPEZ MARTINEZ
  20 BORDER 0: PAPER 0: FLASH 0: BRIGHT 0: INK 0: CLS

  30 INK 2
  40 PRINT AT 8,13; FLASH 1; PAPER 6;"8"; PAPER 0;"88
88"; PAPER 6;"8"
  50 PRINT AT 9,14; FLASH 1; PAPER 6; INK 0;"8"; PAPE
R 0;"  "; PAPER 6; INK 2;"8"
  60 PRINT AT 10,15; FLASH 1; PAPER 6; INK 0;"8"; PAP
ER 6; INK 2;"8"
  70 PRINT AT 11,15; FLASH 1; PAPER 6; INK 2;"8"; INK
 0;"8"
  80 PRINT AT 12,14; FLASH 1; PAPER 6;"8"; PAPER 0; I
NK 0;"  "; PAPER 6; INK 0;"8"
  90 PRINT AT 13,13; FLASH 1; PAPER 6;"8"; PAPER 0;"8
888"; PAPER 6;"8"
```

Ha llegado el momento de la despedida, pero eso sí, temporal. En nuestro próximo capítulo estudiaremos el resto de técnicas de animación a pantalla completa. Aprovechamos la ocasión para advertir, que para uno de los programas de demostración, va a ser necesario el programa de gráficos en tres dimensiones, aparecido en el segundo capítulo de nuestra sección dedicada a la programación.
En todo caso, sería conveniente que practicásemos con el sistema de animación por **FLASH** que hemos aprendido, puesto que, aunque aparentemente es fácil conseguir el efecto deseado, en la mayoría de los casos llegamos a resultados frustrantes; máxime si se utiliza el segundo tipo de animación, en el cual las letras se entrecruzan, haciendo de la pantalla un misterio difícil de desentrañar.

SALVAR: Grabar un programa en cinta, disco o cualquier otro soporte, mediante el comando **SAVE**.

*

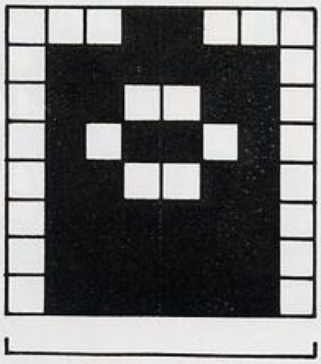
BIT: Unidad mínima de información. Se representa por un uno o un cero binario.

*

BYTE: Agrupación de ocho bits. La información que puede contener un byte va desde 0 hasta 255.

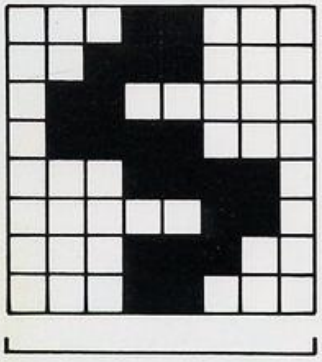
¡ARAÑAS!

ESTE programa nos brinda la oportunidad de convertirnos en un spray super eficaz con el que defendernos del despiadado ataque de unos arácnidos.
El objetivo del juego es evitar que alguno de los artrópodos alcance la barra azul en la cual estamos situados. Para ello, nos desplazaremos a izquierda y derecha, pulsando las teclas Z y X respectivamente, y lanzaremos nuestra mortífera carga de DDT por medio de la tecla SPACE. Cuanto mayor sea el número de puntos conseguido, más rápidamente se descolgarán las arañas por la pantalla. Si al final del juego hemos logrado batir el récord, nuestro Spectrum nos lo hará saber mediante una serie de notas y palpitaciones en el marcador (sin duda debidas a la emoción del momento).
Aunque nuestro SPRAY tiene la suficiente carga como para eliminar miles y miles de arañas, debido a la gran rapidez de los arácnidos estamos irremisiblemente condenados a la desaparición; sin embargo, la competencia se puede establecer entre nosotros y los otros jugadores, o bien con nosotros mismos, intentando batir los récords de permanencia establecidos.



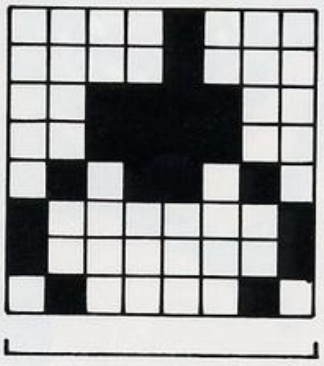
SPRAY

El spray se consigue gracias al gráfico definido de la A.



MATA-ARAÑAS

El mortífero chorro de insecticida, se representa mediante el gráfico definido de la letra B.



ARAÑA

Los arácnidos corresponden a los gráficos definidos de la C.

VARIOS NIVELES

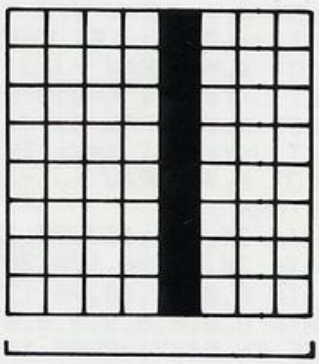
Este programa nos da pie a introducir un concepto importante en la confección de juegos de acción: no hay nada más desesperante que un juego en el que por muy experimentado que se sea, nunca se consiga alcanzar un mínimo número de puntos; para evitar esto, hemos de escoger un nivel intermedio para la dificultad del juego, que no sea ni tediosamente sencillo, ni increiblemente difícil.
En el programa ARACNIDOS, el control de esta dificultad es gestionado por la línea 210, mediante el siguiente algoritmo: **RND*10000 >P+7000;** cuando esta condición se cumple, no

se ejecuta el grupo de instrucciones que efectúa el avance de las arañas.
Observando el algoritmo, se aprecia que en el mejor de los casos, al comenzar el juego, la probabilidad de avance del arácnido es de aproximadamente el 30 por 100, y conforme vamos adentrándonos en el juego, esta probabilidad aumenta hasta alcanzar el 100 por 100, lo cual sucede al rebasar los 3000 puntos.

CARACTERES SUBRAYADOS

Como es habitual, los caracteres que aparecen subrayados en el listado, deben sustituirse por los gráficos definidos de las teclas correspondientes; así por ejemplo, una A subrayada, corresponde a un carácter gráfico de la A, que debe ser introducido con el ordenador en modo **GRAPHICS (CAPS SHIFT + 9)**, y pulsando a continuación la tecla A, sin olvidar desconectar a continuación el modo GRAFICO con **CAPS SHIFT + 9** o, por tratarse de la desconexión, simplemente con **9**.
La adopción del programa se efectuará por el método habitual, puesto que este no incluye código máquina: **SAVE "ARACNIDOS"**, o bien, **SAVE "ARACNIDOS" LINE 10**, para que al finalizar la carga, el programa se ejecute automáticamente desde la línea 10.

Con el gráfico definido de la D, se representan los hilos por los que se descuelgan las arañas.



HILO

¡!

Los caracteres subrayados que aparecen en el listado del programa, corresponden a los gráficos definidos, por lo que a la hora de introducir el programa, pondremos el cursor en modo gráficos, pulsando posteriormente la letra subrayada.

*

El algoritmo que controla la dificultad de juego es: **RND*10000 >P+ 7000.**

*

El movimiento del spray se efectúa mediante la pulsación de las teclas Z (izquierda), X (derecha) y SPACE (fuego).

*

Para grabar el programa ejecutaremos el comando: **SAVE ''ARACNIDOS'' LINE 10.**

```
 10 BORDER 5: PAPER 5: INK 6: BRIGHT 0: CLS
 20 LET R=0: POKE 23658,8
 30 FOR I=0 TO 31
 40 READ J
 50 POKE USR "A"+I,J
 60 NEXT I
 70 BORDER 5: PAPER 5: INK 6: BRIGHT 0: CLS
 80 FOR I=0 TO 30: GO SUB 300: NEXT I
 90 INK 2: PAPER 5: PRINT AT 4,7; BRIGHT 1;"
          ": FOR I=5 TO 16: PRINT AT I,7; BRIGHT 1;"
";AT I,24; BRIGHT 1;" ": NEXT I: PRINT AT 17,7; BRIGH
T 1;"
100 BRIGHT 0: PRINT AT 1,1; PAPER 1; INK 9;"PUNTOS:0
000";AT 1,20;"RECORD:";FN P$(R);AT 20,1; PAPER 1; INK
 7;"Z=IZDA. X=DECHA. 'SPACE'=SPRAY": PAPER 6: BRIGHT
1
110 PRINT AT 5,8; PAPER 2;TAB 24
120 FOR I=6 TO 15: PRINT AT I,8;TAB 24: NEXT I
130 PRINT AT 16,8; PAPER 1;TAB 24
140 LET X=16: DIM A(16): LET P=0: PRINT AT 16,16; PA
PER 1; INK 7;"A"
150 LET X$=INKEY$
160 IF X$="Z" AND X>8 THEN  LET X1=X-1: GO TO 200
170 IF X$="X" AND X<23 THEN  LET X1=X+1: GO TO 200
180 IF X$=" " THEN  FOR I=14 TO 6 STEP -1: PRINT AT
I,X; INK 4;"B";AT I+1,X;" ": BEEP .1,0: NEXT I: PRINT
 AT 6,X;" ": LET P=P+(25 AND A(X-7)): LET A(X-7)=0: P
RINT AT 1,8;FN P$(P)
190 GO TO 210
200 PRINT AT 16,X; PAPER 1;" ";AT 16,X1; INK 7;"A":
LET X=X1
210 IF RND*10000>P+7000 THEN  GO TO 150
220 LET A=INT (RND*16)+1
230 PRINT AT A(A)+6,A+7;"D";AT A(A)+7,A+7;"C": LET A
(A)=A(A)+1: IF A(A)=9 THEN  GO TO 250
240 GO TO 150
250 IF P>R THEN  FLASH 1: LET R=P: PRINT AT 1,27;FN
P$(R): FOR I=30 TO 0 STEP -1: BEEP .1,I: NEXT I: FLAS
H 0
260 PRINT AT 11,3;"DESEA INTENTARLO OTRA VEZ?"
270 IF INKEY$="S" THEN  GO TO 70
280 IF INKEY$="N" THEN  GO TO 10000
290 GO SUB 270
300 LET X=INT (RND*253)+1: LET Y=INT (RND*174)+1
310 PLOT X,Y: DRAW 0,-Y: PLOT X,Y: DRAW 0,175-Y: PLO
T X,Y: DRAW -X,0: PLOT X,Y: DRAW 255-X,0
320 LET J=175-Y: LET K=255-X: IF K<J THEN  LET J=K
330 PLOT X,Y: DRAW J,J
340 LET J=175-Y: LET K=X: IF K<J THEN  LET J=K
350 PLOT X,Y: DRAW -J,J
360 LET J=Y: LET K=255-X: IF K<J THEN  LET J=K
370 PLOT X,Y: DRAW J,-J
380 LET J=Y: LET K=X: IF K<J THEN  LET J=K
390 PLOT X,Y: DRAW -J,-J
400 RETURN
410 DATA 24,126,102,90,102,126,126,126
420 DATA 24,56,96,124,30,6,28,24
430 DATA 8,8,60,60,90,129,129,66
440 DATA 8,8,8,8,8,8,8,8
450 DEF FN P$(Y)=CHR$ 19+CHR$ 0+CHR$ 17+CHR$ 1+CHR$
16+CHR$ 7+("000"+STR$ Y)(LEN STR$ Y TO )
```