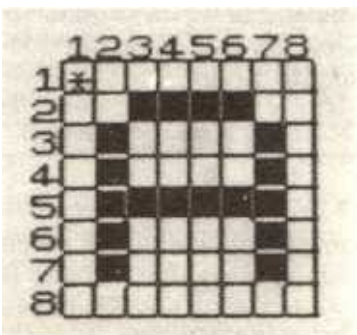
CAPÍTULO CUARTO

Los gráficos

CARACTERES DEFINIDOS POR EL USUARIO

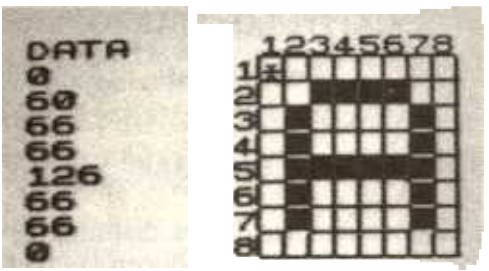
Ahora ya tiene una idea de los distintos caracteres, letras, números y símbolos que producen las teclas del Spectrum. Pero también es posible crear sus propios caracteres. Cosas como alfabetos extranjeros, invasores del espacio, símbolos técnicos, etc.

El Spectrum da gran facilidad para la creación de estos caracteres, y tiene un área de la memoria suficiente para el almacenamiento de hasta 21 caracteres. Cuando se conecta el Spectrum estos caracteres son las letras mayúsculas de la A a la U, y se obtienen pulsando dichas teclas, estando el cursor en modo G (CAPS SHIFT y 9). Para entender cómo se crea uno de estos nuevos caracteres, mire este dibujo que representa un carácter de los normales contenidos en el Spectrum.



Aquí está la letra mayúscula A tal como la hace el Spectrum. La he dibujado a propósito en un cuadrado de

8 × 8, porque cada letra, o símbolo del Spectrum está diseñado de este modo, con cada uno de los 64 cuadritos lleno o vacío. Para la creación de caracteres, usted debe pensar en este cuadrado. En el hay ocho filas de ocho puntos cada una, cada fila tiene un número asociado llamado "byte" (¿recuerda esta palabra del capítulo 1?). Aquí aparece el cuadrado con el carácter y el byte asociado a cada fila. Veamos cómo actúan estos números.



Para entender de dónde vienen estos números es necesario saber un poco de **aritmética binaria.** La aritmética binaria, consiste simplemente en contar de dos en dos en lugar de en decenas como hacemos normalmente. El problema está en que estamos acostumbrados a contar en decenas y entonces es fácil olvidar que puede haber otras maneras de hacerlo. Observemos este número escrito tal como lo hacemos normalmente:

1 0 6 4

Ahora pensemos en él como si fuera un 1 en la columna de los millares, un 0 en la de las centenas, 6 en las decenas, y 4 en las unidades. De esta manera podemos marcar cada columna del siguiente modo:

10^3	10^2	10^1	10^0
1	0	6	4

Si no está acostumbrado a contar en potencias, no se preocupe, esto, lo único que significa es que la primera columna es 1000, la segunda 100, la tercera 10 y la última 1. Ahora bien, también es posible contar en pares, y es de este modo como opera la aritmética binaria. Veamos las columnas en binario:

2^3 2^2 2^1 2^0

Las columnas son 2 al cubo (8), 2 al cuadrado (4), dos elevado a 1 (2) y 2 elevado a cero (1).

Al contar en decenas, cada columna puede tener uno de los diez dígitos (Ø a 9), y de manera similar, contando en binario, cada columna puede contener uno de los dígitos (cero o uno). Ahora recuerde del primer capítulo que el Spectrum trataba los dos niveles de voltaje como Ø o 1. De esto se deduce que la aritmética binaria es fundamental en los ordenadores. Vamos a escribir un número en binario, los números 1, 2, 4 y 8 son muy fáciles:

Decimal	Binario
1	ØØØ1
2	ØØ1Ø
4	Ø1ØØ
8	1ØØØ

En el primero no hay ni ochos ni cuatro ni doses, pero sí un uno y así ocurre con los demás. ¿Cómo se escribirá el cero?, ¿y el 12? Intente convertir algunos números de binario a decimal, pero no se preocupe porque en seguida veremos una manera muy fácil de hacerlo.

Antes, sólo hemos marcado la cabecera de las cuatro columnas de la derecha, pero en binario; las cuatro de la izquierda se encabezarían 128, 64, 32 y 16 respectivamente de derecha a izquierda.

Volvamos a las filas de la letra A que estábamos viendo antes. La primera está vacía, y no es difícil adivinar que el número que le corresponde es el cero (ØØØØØØ en binario). Pero la segunda tiene llenos la mitad de sus puntos.

Para calcular a qué número corresponde esta fila necesitamos escribirla como un número binario, de manera que los puntos vacíos sean ceros, y los llenos unos. Así esta fila corresponde al número ØØ111ØØ. Para convertir este número a decimal, hay que hacer los siguientes cálculos:

128*Ø + 64*Ø + 32*1 + 16*1 + 8*1 + 4*1 + 2*Ø + 1*Ø

que dan como resultado

$$32 + 16 + 8 + 4 = 60$$

Mire a ver si puede averiguar cómo se obtienen los números correspondientes a las otras filas.

Ahora ya ha visto los principios de la aritmética binaria, y de cómo se relaciona con los caracteres. Voy a explicarle una manera muy sencilla de convertir números de binario a decimal:

CONSEJO

En el Spectrum hay un comando especial llamado **BIN.** Ha sido incluido para permitir la entrada de números binarios en el Spectrum. Pero también permite su conversión a la forma decimal simplemente escribiendo "PRINT BIN...".

Escriba esto para ver lo que quiero decir

```
PRINT BIN 00111100    (ENTER
PRINT BIN 01010101
PRINT BIN 11111111
```

Como recordará, el primer ejemplo, corresponde a la segunda línea de la letra A. Y el Spectrum ha imprimido el número 60 en la pantalla.

CREACIÓN DE UN CARÁCTER

Para hacerlo, el Spectrum dispone de unos comandos especiales, que le permiten introducir los ocho bytes necesarios para cada nuevo carácter. Con estos comandos, los bytes se sitúan automáticamente en la posición de memoria correspondiente a cada carácter, para que usted no tenga que efectuar ningún cálculo. Aquí viene un ejemplo en el que se crea la A mayúscula en el lugar en que se encontraba la U en modo gráfico (cursor G). Esto no es muy útil en principio, puesto que la letra A ya existe en el Spectrum, pero viendo como se construye comprenderá la forma de crear cualquier otro carácter.

```
10 POKE USR "U",0
20 POKE USR "U" + 1,60
30 POKE USR "U" + 2,66
40 POKE USR "U" + 3,66
50 POKE USR "U" + 4,126
60 POKE USR "U" + 5,66
70 POKE USR "U" + 6,66
80 POKE USR "U" + 7,0
```

Este programa puede parecerle un poco extraño, pero ahora le explicaré el funcionamiento. El comando **USR** se obtiene pulsando la tecla L con el cursor en modo E. USR "U" es la dirección de memoria en la que se almacena el primer byte del carácter definible por el USuaRio que corresponde a la letra U en modo G. Puede ver a qué dirección corresponde simplemente escribiendo:

```
PRINT USR "U"
```

Observe cómo la dirección que se obtiene concuerda con los límites del mapa de memoria que se encuentra en la página 165 del manual.

POKE simplemente significa "coloca el número que sigue en la dirección que se indica" (primero se escribe la dirección, luego una coma y por último el valor decimal del byte que se desea colocar en esa dirección). Puede imaginarse que la memoria del ordenador es como una larga fila de cajas, cada una de las cuales corresponde a una posición (byte) de memoria. La posición (o dirección) más baja es la cero, y la más alta la 65536. En cada una de estas cajas hay un número (de cero a 255, ya que es lo máximo que se puede conseguir con un byte), que de hecho es un byte de información o de datos. Así, en las direcciones que van desde USR "U" hasta USR "U" + 7 hay ocho bytes que corresponden al carácter definido por el usuario de la tecla U (que al enchufar el Spectrum corresponden, a los de la letra U, pero después de correr el programa anterior, serán los de la letra A). Ahora escriba el programa colocando las comas en su lugar y ejecútelo. Luego, con el cursor en modo G pulse la letra U y en lugar de obtener la U mayúscula como ocurría antes, aparecerá una A mayúscula. Una vez visto esto ya es muy fácil colocar cualquier otro carácter en esta tecla.

HINT

Probablemente, la mejor manera de definir nuevos caracteres, es utilizar READ DATA y RESTORE, colocando los bytes de los caracteres en líneas DATA. De todas maneras, el uso de BIN puede hacer mucho más fácil la comprensión del programa listado, ya que colocando los bytes en binario, se entrevé el carácter a definir. Por ejemplo, hagámoslo de nuevo con nuestra A mayúscula:

```
10 POKE USR"U"          BIN 00000000
20 POKE USR"U" + 1,     BIN 00111100
30 POKE USR"U" + 2,     BIN 01000010
40 POKE USR"U" + 3,     BIN 01000010
50 POKE USR"U" + 4,     BIN 01111110
60 POKE USR"U" + 5,     BIN 01000010
70 POKE USR"U" + 6,     BIN 01000010
80 POKE USR"U" + 7,     BIN 00000000
```

Esta habilidad que tiene el Spectrum de introducir los números directamente en binario, da mucha facilidad a la hora de crear un carácter gráfico.

READ Y DATA PARA LOS NUEVOS CARACTERES

Quizá la manera más elegante de crear nuevos caracteres (una vez conocidos los bytes que lo componen) es usar READ y DATA. Colocando estamentos REM que clarifiquen lo que contiene cada línea DATA, se consigue una mayor claridad. Aquí hay un ejemplo que podría muy bien ser una parte de un programa de invasores:

```
10 REM INVASORES
20 REM CREACION DEL GRAFICO
30 GOSUB 1000
40 STOP
1000 FOR A = 1 TO 7: READ X: POKE
USR"A" + A,X:
NEXT A
```

```
1100 DATA 24,60,90,126,24,36,90
1200 RETURN
```

MOVIMIENTO

Una vez creado el carácter del invasor del espacio, ¿cómo podemos hacer que se mueva por la pantalla? Hay varias maneras de conseguirlo, pero la más simple consiste en utilizar PRINT AT. Es decir, se imprime un carácter en una posición de la pantalla, luego se borra y se imprime de nuevo una posición más adelante. Esto se puede repetir hasta que se termine la pantalla. Aquí tenemos una manera de hacerlo:

```
10 REM MOVIMIENTO
20 FOR A = 0 TO 31
30 PRINT AT 5,a;"*";AT 5,a;"^"
40 NEXT A
```

Si lo prueba, probablemente se sorprenderá de lo rápido que se mueve el asterisco por la pantalla. Para aminorar la velocidad, podemos utilizar la instrucción **PAUSE.** PAUSE se usa seguida de un número positivo, para su orientación PAUSE 50 detiene el programa durante un segundo. Intentemos añadir PAUSE 10 a nuestra rutina. Escriba esta línea:

```
30 PRINT AT 5,a; "*": PAUSE 10: PRINT AT 5,a; "^"
```

Esto provoca que el movimiento del asterisco, sea más lento que antes. Por supuesto, que usted puede aumentar la velocidad, disminuyendo el número que sigue a PAUSE. También puede cambiar el asterisco por el carácter de un invasor (¿por qué no prueba con el que hemos definido antes?).

Otra manera de hacer lo mismo sin necesidad de escribir el espacio encima del carácter cada vez, es imprimiéndolos los dos a la vez, como hace este programa:

```
10 FOR A = 0 TO 30
20 PRINT AT 5,A;"^*"
30 PAUSE 2
```

```
40 NEXT A
50 PRINT AT 5,31;"
```

De esta manera, sólo se necesita la última sentencia PRINT que imprime un blanco sobre el último asterisco de la derecha.

Si lo que queremos es mover un carácter que represente a una pelota, ¿cómo podemos detectar cuando choca contra una pared? Hay dos maneras de detectar si un objeto que se mueve por la pantalla, ha chocado con algo. Uno usa el comando **SCREEN,** y el otro usa **ATTR.** Veamos cómo actúan.

SCREEN es un comando que va seguido de dos números como PRINT AT, y representan coordenadas en la pantalla de la misma manera (SCREEN significa pantalla en inglés). Su función, es localizar un lugar de la pantalla y ver qué carácter hay allí. Pruebe este simple programa para verlo en acción:

```
10 PRINT AT 10,10;"A"
20 PRINT "SCREEN$ (10, 10) ES";
" "" ";SCREEN$(10,10);" "" "
```

Si lo ejecuta obtendrá:

SCREEN$(10,10) IS "A"

Observe que se necesitan cuatro comillas para producir las que envuelven a la letra A.

Esto podemos usarlo para ver cuándo un carácter en movimiento va a chocar contra otro. Introduzca este programa tal como está aquí:

```
  10 REM PELOTA SALTARINA
  20 LET T=1
  30 LET R=1
  40 FOR A=0 TO 21
  50 PRINT AT A,31;"H";AT A,0;"H"
  60 NEXT A
  70 FOR A=0 TO 30: PRINT AT 0,A;"X";AT
21,A;"X"
```

```
 80 NEXT A
 90 LET X=RND*10: LET Y=RND*15
100 PRINT AT X,Y;"o"
110 PAUSE 1
120 PRINT AT X,Y;" "
130 IF SCREEN$ (X+T,Y+R)="H" THEN  LET
R=R*-1
140 IF SCREEN$ (X+T,Y+R)="X" THEN  LET
T=T*-1
150 LET X=X+T: LET Y=Y+R
160 GO TO 100
```

Esta "bola loca" es localizada del siguiente modo: El programa dibuja las paredes de arriba y de abajo con la letra X, y las de la derecha e izquierda mediante la letra H. Entonces el programa mueve la pelota, hasta que detecta el impacto con una X o una H. Cuando esto ocurre (líneas 130 y 140) entonces se cambia la dirección de la pelota multiplicando R o T por menos uno. Usted puede mantener la trayectoria de la pelota en la pantalla simplemente suprimiendo las líneas 110 y 120.

El uso de SCREEN$ tiene un inconveniente, y es que sólo reconoce los caracteres almacenados en la ROM del Spectrum, y no lee los caracteres definidos por el usuario. Esto es una pena porque se ve claramente que SCREEN puede tener una gran utilidad para juegos en los que intervengan caracteres definidos por el usuario. De todos modos, existe una manera de solucionar el problema. Esencialmente, lo que hay que hacer es primero determinar en qué lugar se va a leer un carácter definido por el usuario, luego cambiar una *variable del sistema* de manera que SCREEN crea que la tabla de caracteres empieza 256 bytes antes de la tabla de gráficos definidos por el usuario, en vez de 256 bytes antes de la tabla de caracteres en la ROM. En el siguiente ejemplo, A y B, son las coordenadas del gráfico que se quiere detectar. En esta rutina, en lugar de almacenar el carácter, se almacena su código en la variable X, pero usted puede almacenarlo como una cadena simplemente poniendo: LET A$ = SCREEN$ (A, B).

```
 10 REM subrutina para usar SCREEN$ con
los graficos definibles
 15 LET x=CODE SCREEN$ (a,b)
```

```
  20 IF x THEN  RETURN
  25 REM 23606&23607 punto para la tabla
de CHR$
  30 POKE 23606,PEEK 23675
  35 REM 23675&23676 punteros para usar
el area de graficos definidos
  40 POKE 23607,(PEEK 23676)-256
  45 LET x=CODE SCREEN$ (a,b)+112
  50 REM cambio para CHR$
  55 POKE 23606,0
  60 POKE 23607,60
  70 RETURN
```

ATTR es una abreviación de la palabra inglesa ATTRIBUTE, y usándolo, devuelve los atributos (características) de la posición que se le indica (otra vez, por tanto, necesita ir seguido de dos coordenadas como PRINT AT). ATTR es más difícil de usar que SCREEN ya que su resultado es un número que es la suma de otros distintos, cada uno de los cuales representa una característica del carácter que se encuentra en la posición indicada; por ejemplo, nos dice de qué color es la tinta, de qué color es el papel, en el lugar en que está el carácter, etc.

El número calculado por ATTR es la suma de los siguientes:

128 si la posición está en FLASH (o si no lo está)
64 si la posición está en BRIGHT
8 multiplicado por el código de color del papel
el código de la tinta

Por ejemplo:

```
10 PRINT AT 10,10; INK 2; PAPER 6;"*"
20 PRINT ATTR (10,10)
```

Ejecute esto y obtendrá la respuesta 50. Que está compuesta de 0, porque no está en FLASH, 0 porque no está en BRIGHT, 8*6 por el color del papel (PAPER), y 2 por la tinta (INK). Esto da 0 + 0 + 48 + 2 = 50. Si usted desea que el cuadrado esté en FLASH (use EDIT para añadir FLASH 1 a la línea 10) entonces el resultado será 178 (50 más 128 por el FLASH).

Realmente el uso de ATTR requiere un poco de atención. Aun así el programa de la bola loca, podría ser reescrito usando ATTR. Lo único que habría que hacer es

cambiar el color de las paredes, es decir, poner uno diferente arriba y abajo y en los lados (entonces en lugar de Xs y Hs podrían ser cualquier gráfico definido por usted). Una vez determinado el ATTR para los dos tipos de paredes (horizontal y vertical) sólo hay que cambiar las líneas 130 y 140 por otra sentencia condicional que compruebe el ATTR en lugar de SCREEN.

Por ejemplo:

130 IF ATTR(X + t,Y + r) = 50 THEN LET r = r* – 1

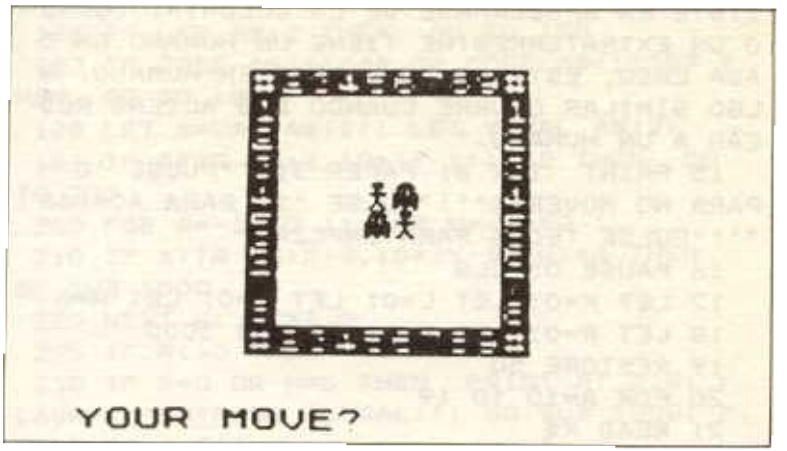
Aquí hay un programa de un juego mucho más excitante que hace uso de ATTR. Trata de unas estrellas que aparecen y desaparecen viajando por la pantalla. Las teclas 5 y 8 le mueven a derecha y a izquierda, las teclas 1 y 0 le permiten saltar diez puestos.

```
  10 REM tragon
  20 BORDER 1: PAPER 0: INK 6: CLS
  30 LET chr=480
  40 GO SUB chr
  50 LET x=5
  60 LET hit=400
  70 LET cont=1
  80 LET tantos=0
  90 LET b=0
 100 LET w=0
 110 LET s=10
 120 LET y=0
 130 LET a=INT (RND*10)
 140 FLASH 0
 150 LET b=b+2*(a>=5 AND b<28)-2*(a<4 AN
D b>0)
 160 REM a tiene que ser 'a' con modalid
ad de graficos
 170 PRINT  INK 4;AT x,y+5;"a"
 180 IF INKEY$="1" THEN  LET w=(-10 AND
s>10)
 190 IF INKEY$="0" THEN  LET w=(10 AND s
<21)
 200 IF INKEY$="5" THEN  LET w=-1
 210 IF INKEY$="8" THEN  LET w=1
```

```
 220 IF SCREEN$ (x+1,y+s)="*" THEN  GO S
UB hit
 230 LET s=s+w
 240 IF s>31 THEN  LET s=31
 250 IF s<0 THEN  LET s=0
 260 IF cont/2=INT (cont/2) THEN  GO TO
280
 270 GO TO 290
 280 PRINT AT 20,0;TAB 8+b;"*"
 290 PRINT AT 20,0;
 300 POKE 23692,-1
 310 PRINT
 320 PRINT
 330 PRINT
 340 LET cont=cont+1
 350 PRINT AT x-1,y+s-w;" "
 360 PRINT AT x-2,y+s-w;" "
 370 IF cont=200 THEN  GO TO 460
 380 LET w=0
 390 GO TO 130
 400
 410 REM tantos
 420
 430 LET tantos=tantos+1
 440 PRINT  FLASH 1; INK 2;AT 2,12;"HIT"
 450 RETURN
 460 PRINT AT 10,5; FLASH 1;"TUS PUNTOS;
";tantos;" PUNTOS"
 470 STOP
 480 REM ***CHR***
 490 RESTORE 540
 500 FOR k=0 TO 7
 510 READ n
 520 POKE USR "a"+k,n
 525 NEXT k
 530 RETURN
 540 DATA 60,90,126,60,24,36,90,0
```

Aunque parezca que el uso de ATTR es menos problemático que el de SCREEN, a veces es agradable saber qué carácter se encuentra en una posición determinada. En el siguiente juego llamado COLONY SCREEN, se puede hacer que el ordenador juegue sus movimientos y que compruebe los suyos, pero esto sólo si se usan caracteres

normales, no definidos por el usuario. El juego es una versión del popular Othello.




```
    2 REM LA COLONIA
    3 BORDER 5: PAPER 0: INK 6: CLS
    4 PRINT AT 0,0;" .... .... .... ....
.... .. ..";AT 1,0;" .... .... .... ....
 .... .. .."
    5 PRINT AT 2,0;" ..   .... ..    ..
..   ...."
    6 PRINT AT 3,0;" .... .  . ...   ..
.... ..."
    7 PRINT AT 4,0;" .... .... ...   ..
.... ..."
    8 PRINT AT 5,0;"   .. .... ..    ..
..   .... "
    9 PRINT AT 6,0;" .... .... ..    ..
.... .. .."
   10 PRINT AT 7,0;" .... .... ..    ..
.... .. ..";AT 8,30;"TM"
   11 PRINT AT 10,12; FLASH 1; PAPER 4; I
NK 6;"PRESENTA": PLOT 0,50: DRAW 15,0: D
RAW 60,120: PAUSE 500: INK 5: CLS
   12 FOR A=1 TO 15: FOR B=1 TO 6: PRINT
 FLASH 1; PAPER B; INK 9;AT 10,12;"COLON
IA": BEEP .01,RND*20: PRINT AT RND*20,RN
D*30; INK RND*6;"HC": NEXT B: NEXT A
   13 PRINT  FLASH 0; PAPER 3; INK 6;AT 4
,0;"PARA JUGAR "; FLASH 0;"ENTRE LAS COO
```

```
RDENADAS DE POSICION PONIENDO PRIMERO LA
 FILA Y LUEGO LA COLUMNA"
  14 PRINT  INK 6; PAPER 3;"EL JUEGO CON
SISTE EN APODERARSE DE LA COLONIA. CUAND
O UN EXTRATERRESTRE TIENE UN HUMANO EN C
ADA LADO, ESTE SE CONVIERTE EN HUMANO. A
LGO SIMILAR OCURRE CUANDO DOS ALIENS ROD
EAN A UN HUMANO."
  15 PRINT  INK 6; PAPER 3;''"PULSE '0'
PARA NO MOVERSE"''"PULSE 'S' PARA ACABAR
"''"PULSE TECLA PARA EMPEZAR"
  16 PAUSE 0: CLS
  17 LET K=0: LET L=0: LET P=0: LET H=0
  18 LET R=0: LET V=0: GO SUB 3000
  19 RESTORE 50
  20 FOR A=10 TO 19
  21 READ X$
  30 PRINT  INK 3; INVERSE 1;AT 5,A;X$
  40 NEXT A
  50 DATA "1","2","3","4","5","6","7","8
","9","1"
  60 FOR Z=1 TO 8
  65 FOR F=1 TO 2
  70 READ X
  80 PRINT  INK 3; INVERSE 1;AT 5+Z,10+(
9*V);X;
  90 LET V=V+1
 100 IF V=2 THEN  LET V=0
 105 NEXT F
 110 NEXT Z
 120 DATA 1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,
8
 125 RESTORE 50
 130 FOR A=10 TO 19
 140 READ X$
 150 PRINT  INK 3; INVERSE 1;AT 14,A;X$
 160 NEXT A
 170 PRINT AT 9,14; INK 6;"H";AT 9,15; I
NK 4;"C";AT 10,14;"C";AT 10,15; INK 6;"H
"
 175 POKE 23609,100
 180 INPUT "MUEVES?"; LINE A$
 181 IF A$="0" THEN  GO TO 8000
 182 IF A$="S" OR A$="s" THEN  GO TO 990
0
```

```
 183 LET R=0: LET P=0: LET H=1
 184 BEEP .1,1: BEEP .1,3: BEEP .1,2: BE
EP .2,4
 186 IF LEN A$<2 THEN  GO TO 180
 187 IF CODE A$(1)<49 OR CODE A$(1)>57 T
HEN  GO TO 180
 188 LET X=VAL A$(1): LET Y=VAL A$(2)
 191 IF ATTR (5+X,10+(Y-1))<>5 THEN  GO
TO 230
 200 FOR Q=-1 TO 1: FOR W=-1 TO 1
 210 IF ATTR (5+X+Q,10+(Y-1)+W)=4 THEN
GO SUB 1000
 220 NEXT W: NEXT Q
 225 IF R<>0 THEN  GO TO 240
 230 IF R=0 OR H=0 THEN  PRINT AT 3,8; F
LASH 1;"ENTRADA ILEGAL!": GO SUB 2000: P
RINT AT 3,8;"                    "
 235 IF H=0 THEN  PRINT AT X+5,B;" "
 240 IF R=0 THEN  GO TO 180
 245 IF H=0 THEN  GO TO 180
 250 PRINT AT 5+X,10+(Y-1); INK 6;"H"
 260 GO TO 5000
1000 LET R=1
1010 RETURN
2000 FOR E=0 TO 5: FOR A=-10 TO -20 STEP
 -1: BEEP .01,A: NEXT A: NEXT E
2010 RETURN
3000 RESTORE 3010
3005 FOR A=0 TO 7: READ X: POKE USR "C"+
A,X: NEXT A
3010 DATA BIN 00011100,BIN 00011100,BIN
00101010,BIN 01101011,BIN 01111111,BIN 0
1111111,BIN 01101101,BIN 01001001
3020 RESTORE 3040
3030 FOR A=0 TO 7: READ X: POKE USR "H"+
A,X: NEXT A
3040 DATA BIN 00011100,BIN 00011100,BIN
00001000,BIN 00111110,BIN 00001000,BIN 0
0001000,BIN 00010100,BIN 00100010
3050 RETURN
5000 LET A=5+X: LET B=10+(Y-1)
5100 LET H=0
6000 FOR Q=-1 TO 1 STEP 1
6010 FOR W=-1 TO 1 STEP 1
```

```
6020 IF ATTR (A+Q,B+W)=4 THEN  GO SUB 70
00
6030 NEXT W: NEXT Q
6036 IF H=0 THEN  GO TO 230
6040 GO TO 8000
7000 IF ATTR (A+(2*Q),B+(2*W))=6 THEN  P
RINT  INK 6;AT (A+Q),(B+W);"H"
7010 IF ATTR (A+Q;B+W)=6 THEN  LET H=1
7020 RETURN
8000 LET C=INT (14*RND)
8005 FOR A=C TO 14
8010 FOR B=11 TO 19
8020 IF ATTR (A,B)=4 THEN  GO SUB 8500
8030 NEXT B: NEXT A
8040 LET C=6: LET P=P+1
8050 IF P=2 AND A$="O" THEN  GO TO 9900
8060 GO TO 8005
8500 FOR Q=-1 TO 1
8510 FOR W=-1 TO 1
8520 IF ATTR (A+Q,B+W)=6 THEN  GO TO 855
0
8530 NEXT W: NEXT Q
8540 RETURN
8550 IF ATTR (A+(2*Q),B+(2*W))=5 THEN  G
O TO 9000
8560 GO TO 8530
9000 BEEP .1,4: BEEP .1,2: BEEP .1,3: BE
EP .2,1
9002 BEEP .1,4: BEEP .1,2: BEEP .1,3: BE
EP .2,1
9005 PRINT  INK 4;AT A+(2*Q),B+(2*W),"C"
9010 PRINT  INK 4;AT A+Q,B+W;"C"
9020 LET A=A+(2*Q): LET B=B+(2*W)
9030 FOR Q=-1 TO 1: FOR W=-1 TO 1
9040 IF ATTR (A+Q,B+W)=6 THEN  GO TO 906
0
9050 NEXT W: NEXT Q
9054 LET T=1
9055 GO TO 180
9060 IF ATTR (A+(Q*2),B+(W*2))=4 THEN  P
RINT  INK 4;AT A+Q,B+W;"C"
9070 GO TO 9050
9900 LET H=0: LET C=0
9910 FOR A=6 TO 14: FOR B=11 TO 19
```

```
9920 IF ATTR (A,B)=6 THEN  LET H=H+1
9922 IF ATTR (A,B)=4 THEN  LET C=C+1
9925 IF A$="S" THEN  GO TO 9940
9930 IF ATTR (A,B)=0 THEN  GO TO 180
9940 NEXT B: NEXT A
9941 IF C=H THEN  PRINT AT 2,12; FLASH 1
;"DIBUJA": STOP
9950 IF C>H THEN  GO TO 9990
9960 PRINT  PAPER 6; INK 2; FLASH 1;AT 2
,4;"GANASTE CON ";H;" A MI";C
9965 FOR A=20 TO 40: BEEP .2,A: NEXT A:
GO TO 9965
9970 STOP
9990 PRINT  INK 2; PAPER 6; BRIGHT 1; FL
ASH 1;AT 2,0;"HE GANADO CON ";C;" A TUS
";H
```

Como puede observar, en este programa se usan muchas líneas del tipo:

```
10 IF CODE SCREEN$ (10,10) = 42 THEN GOTO 1000
```

El uso de **CODE** permite obtener lo que se llama el **código ASCII** de cualquier carácter producido por el Spectrum. En el apéndice se proporciona un listado de todos los códigos ASCII. Todos los caracteres que se usan tienen un número asociado a ellos. Si usted está interesado en esto, el siguiente programa le listará todos los caracteres y sus códigos.

```
10 FOR A = 32 TO 255
20 PRINT "CODE ";A "ES ";CHR$A
30 NEXT A
```

CODE y **CHR** hacen lo inverso el uno del otro, así, CHR 122 es 'z' mientras que CODE 'z' es 122. En el programa anterior pulse ENTER como respuesta a la pregunta "scroll" para visualizar una nueva pantalla de caracteres y sus códigos. No he incluido los primeros 31 caracteres porque son códigos de control y al intentar imprimir los códigos de control, pueden producirse efectos muy peculiares.

Los códigos del 0 al 5 y del 24 al 31, no se usan en el Spectrum, los otros son:

CÓDIGO	FUNCIÓN
6	coma de PRINT
7	EDIT
8	cursor izquierda
9	cursor derecha
10	cursor abajo
11	cursor arriba
12	DELETE
13	ENTER
14	número
15	sin uso
16	INK (tinta)
17	PAPER (papel)
18	FLASH (parpadeo)
19	BRIGHT (brillo)
20	INVERSE
21	OVER
22	AT
23	TAB

Estos caracteres de control, pueden usarse como comandos en líneas de programa, o también pueden ser muy útiles para usar desde un programa, aspectos que sólo es posible utilizarlos directamente desde el teclado. De hecho, si usted quiere mover el cursor por la pantalla durante un programa, sólo puede hacerlo usando PRINT AT o bien los caracteres de control. Por ejemplo, supongamos que usted quiere subrayar la letra 'a'. Aquí tiene cómo puede hacerse esto usando el carácter de control 8 (cursor izquierda):

```
PRIN  OVER     a'';CHR$ 8;
```

OVER significa que se van a fusionar los dos caracteres en lugar de que el segundo borre al primero.

La producción de color mediante el uso de los caracteres de control, será tratada más tarde, pero ahora, nos será útil observar, que los caracteres de control pueden unirse

(concatenarse) como cadenas, para formar un solo cuerpo gráfico, etiquetado con un mismo nombre:

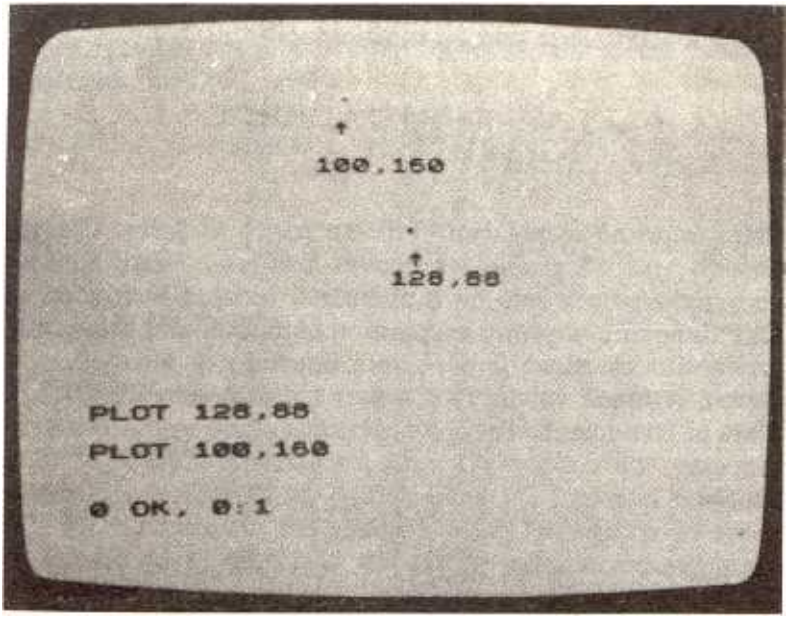
```
10 LET A$ = CHR$17 + CHR$6 + "Hi"
20 PRINT A$
```

Esto imprime la palabra "Hi" en papel amarillo. Por supuesto que A$ puede ser mucho más larga. Este método es equivalente a que la variable A$ recurde por usted lo que tiene que imprimir en un momento dado del programa, pero sólo es eficaz si tiene que imprimir lo mismo varias veces. Aunque volveremos a usar los caracteres de control para el tratamiento del color, observe cómo se ha conseguido esto. La cadena A$ está formada de CHR$ 17 para cambiar el papel (17 es el código de PAPER), y luego se pone el código del color después del próximo CHR$. De una manera similar CHR$ 18 + CHR$ 1 provocará el parpadeo, y así sucesivamente.

DETALLES GRÁFICOS

Algunos ordenadores, tienen lo que se llama **modos de resolución.** Esto permite al usuario escoger el nivel de resolución gráfica que desee en cada momento; es decir, cómo es de grande el punto más pequeño que se puede imprimir, o lo que es lo mismo en cuántas unidades se divide la pantalla. El Spectrum sólo posee un nivel de resolución que es razonablemente alto. Se puede pensar que la pantalla está compuesta de una red de puntos de 256 de ancho por 176 de alto. Para ennegrecer uno de estos puntos, se usa el comando PLOT. El punto queda pintado con el color de la tinta. Pruebe este ejemplo:

```
10 PLOT 128,88
```



Una vez ejecutado el programa, fíjese bien y podrá observar un tenue punto en el centro de la pantalla. Lo que hace esta línea es pintar un punto en la posición 128 (puntos) empezando por la izquierda y 88 contando de abajo para arriba (así por ejemplo, la posición 0,0 corresponde a la esquina superior izquierda de la pantalla). Para ver actuar a la instrucción PLOT más activamente vamos a pintar más puntos en la pantalla:

```
 5 PAPER 0:CLS
10 FOR X = 1 TO 200
20 PLOT INK 7*RND; 255*RND,
175*RND
30 NEXT X
```

Ejecute este programa y verá un "cielo de noche" lleno de estrellas (puntos) coloreadas. Observe cómo se usa RND en la línea 20 para escoger de forma aleatoria tanto el color de la tinta como la posición de los puntos, que están entre 0 y 255 y entre 0 y 175.

Es fácil modificar un poco este programa para que nos muestre realmente las habilidades del Spectrum:

```
10 INK 0: BORDER 1: CLS
20 LET A = RND*127
30 LET B = RND *87
40 PLOT INK RND*7;128 + A,88 + b: PLOT INK RND*7;
   128 - A,87 - B: PLOT INK RND*7; 128 - A,87 + B:
   PLOT INK RND *7; 128 + A,87 - B
50 GOTO 20
```

Observe atentamente la línea 40 y trate de averiguar por qué los puntos se sitúan simétricamente en la pantalla.

Al contrario de lo que dije anteriormente, el Spectrum posee otro nivel de resolución más bajo, que se consigue usando los gráficos ya construidos que se encuentran en las teclas de los números del 1 al 8. Hay un gráfico diferente en cada una de esas teclas, pero no se obtienen pulsando simplemente CAPS SHIFT. Para ver cómo son estos gráficos, coloque el cursor en modo G (CAPS SHIFT y 9). Ahora pulse las teclas del 1 al 8 para ver los primeros ocho gráficos, y luego manteniendo apretado CAPS SHIFT pulse de nuevo estas teclas para obtener otros ocho. Como puede observar los segundos son los inversos de los primeros. Mire el gráfico que se obtiene pulsando la tecla 1 tal cual, verá que es un pequeño cuadrado que ocupa exactamente una cuarta parte del cuadrado que le corresponde a un carácter. Si pensamos que este pequeño cuadrado es un punto, podemos escribir una nueva versión del programa anterior en una resolución más baja:

```
10 BORDER 1: PAPER 0: CLS
20 LET A = RND *10
30 IF A>.5 THEN GOTO 60
40 INK 1 + RND*6
50 GOTO 70
60 INK 0
70 LET X = RND *10
80 LET Y = RND *15
90 PRINT AT 10+ X, 15 + Y; "▗"
100 PRINT AT 10 + X,15 - Y; "▗"
110 PRINT AT 10 - X,15 + Y; "▗"
120 PRINT AT 10 - X,15 - Y; "▗"
130 GOTO 20
```

Al contrario que el programa anterior, éste hace que más o menos en la mitad de ocasiones el color de la tinta sea

el mismo que el del papel (línea 30). Debido a esto, los puntos se crean y se destruyen constantemente. Por supuesto que si lo desea puede añadir esto muy fácilmente en el programa anterior.

RAPIDEZ EN LOS DIBUJOS

Su Spectrum no sólo coloca puntos en la pantalla, sino que también dibuja líneas utilizando el comando DRAW (que significa dibujo). Aquí tiene un ejemplo:

```
10 PLOT 0, 128: DRAW 255,0
```

Esto colocará una línea horizontal en la parte alta de la pantalla. Si hay otro color que no sea negro, la línea aparecerá en este color; pero también se puede definir el color de la tinta localmente. Pruebe:

```
10 PLOT 128,88
20 DRAW INK RND*7;(RND*127)*((RND*2) -
   (RND*87)*((RND*2) - 1)
30 GOTO 10
```

Por lo simple que es el programa, produce efectos muy sorprendentes. La línea 20 hace todo el trabajo. Primero asigna un valor aleatorio para la tinta, y lo mismo hace con las coordenadas de la línea que hay que dibujar (esto se hace desde el punto "PLOTeado" en la línea 10). (RND*127) y (RND*87) definen puntos a una distancia máxima de 127 horizontales y 87 verticales. Al añadir [(RND*2)-1] hacen que las coordenadas sean positivas o negativas aproximadamente en la mitad de los casos, lo que provoca que la línea dibujada llegue a derecha, izquierda, arriba o abajo del punto "PLOTeado". Si aún no lo ve claro modifique el programa para que le escriba los números que se obtienen aleatoriamente, esto le ayudará a ver cómo actúa DRAW.

CÍRCULOS

Del mismo modo que puede dibujar líneas, el Spectrum también puede dibujar círculos con un solo comando. El

comando **CIRCLE** se obtiene con el cursor en modo E pulsando simultáneamente SYMBOL SHIFT y la tecla H.

Este es un ejemplo en el que se dibuja un pequeño círculo en el centro de la pantalla, y otro concéntrico a su alrededor:

```
10 CIRCLE 128,88,10
20 CIRCLE 128,88,87
```

De nuevo la tinta con que desea dibujar el círculo se puede definir fácilmente sin ningún problema:

```
CIRCLE INK 2; 128,88,87
```

También se puede indicar localmente el papel, pero como éste sólo se define en un cuadrado entero, los efectos son un poco raros:

```
CIRCLE INK 3; PAPER 6;100,30,25
```

Los arcos se pueden obtener fácilmente añadiendo un nuevo elemento a los ya familiares PLOT y DRAW:

```
10 PLOT 128,88: DRAW 50,50,PI
```

Este programa dibuja un semicírculo empezando en el punto 128,88 y terminando en el punto situado a 50 posiciones a la derecha y 50 hacia arriba.

Si lo desea puede hacer una nueva versión del programa que dibujaba líneas, usando arcos en vez de líneas rectas, para ello simplemente tiene que reemplazar la línea 50 por ésta última que hemos visto añadiéndole los RNDs correspondientes para que todo se produzca aleatoriamente.