Segunda Parte

Gráficos, color y sonido

CAPÍTULO SEXTO

Vista y sonido

En la primera parte vimos que el Spectrum es capaz de producir ocho colores cuyos códigos van del 0 al 7:

0 Negro
1 Azul oscuro
2 Rojo
3 Magenta
4 Verde
5 Azul claro
6 Amarillo
7 Blanco

También vimos cómo tener el borde coloreado simplemente escribiendo BORDER x, donde x es un número entre 0 y 7. También vimos que los colores pueden cambiarse globalmente (cambiando la tinta y el papel de toda la pantalla, lo cual se consigue poniendo CLS detrás de INK o de PAPER), o bien localmente (colocando los comandos dentro de una sentencia PRINT).

Para refrescar su memoria vamos a poner una frase coloreada en la pantalla:

```
10 PRINT INK 1; PAPER 5; AT 10, 15; "ZX";
   AT 12,12; "SPECTRUM"
```

La tinta de color azul oscuro sobre papel azul claro, queda muy bien. Pero hay otras combinaciones que no. Cambie la línea anterior de modo que escriba la frase en tinta verde sobre papel azul claro. Es más difícil de leer ¿no?

Rojo y magenta tampoco combinan bien, y magenta sobre papel verde es casi imposible de leer. Por lo tanto, vaya con cuidado a la hora de escojer los colores que va a mezclar.

El ejemplo anterior nos muestra la forma más común de colorear la pantalla, pero en realidad hay dos métodos más para hacerlo. El primero de ellos, es el que yo llamo **método directo.** Este método, se basa en el hecho de que con el cursor en modo E usted puede pulsar las teclas del 0 al 7 para cambiar el papel del texto que sigue. Pulsando cualquiera de estas teclas con el cursor en modo E al mismo tiempo que se pulsa CAPS SHIFT, produce el cambio instantáneo de la tinta. Este método es un poco difícil de describir porque en él intervienen instrucciones, que no están en las teclas. Escriba este programa que no hace nada:

```
10 REM FALSO
```

Ahora use EDIT para bajar la línea, una vez abajo, coloque el cursor en modo E pulsando CAPS SHIFT y SYMBOL SHIFT simultáneamente, luego pulse una de las teclas del 0 al 7. Verá cómo el papel de debajo de las palabras REM FALSO cambia instantáneamente. Ahora cambie también la tinta pulsando una de las teclas del 0 al 7 con el cursor en modo E al mismo tiempo que pulsa CAPS SHIFT.

Lo que ha hecho en realidad es poner caracteres de control en la línea, que le dicen al Spectrum que escriba el texto en esos colores. Todo el texto que se encuentre a partir del punto en que usted introdujo los códigos habrá cambiado. Para comprobar esto, escriba otra línea de programa:

```
20 REM OTRA LÍNEA
```

Toda la línea entera, tendrá el mismo color de la tinta y del papel que la línea 10. Este modo de colorear la pantalla, se puede usar para distinguir trozos de los listados de los programas en pantalla. Para cambiar el color de la línea 20 simplemente bájela con EDIT y actúe de un modo similar.

Para volverlo como estaba tiene que hacer marcha atrás

hasta llegar al punto que se incluyeron los códigos invisibles. Por ejemplo, baje la línea 10 con EDIT. Si usted intenta mover el cursor hacia la derecha, parece que hay que pulsar cuatro veces antes de que llegue a la F de FALSO. Esto ocurre porque aunque el cursor no se mueve, está pasando por los códigos que se han introducido. Hay dos para cada color (dos por la tinta, y dos por el papel). Fácilmente puede poner más colores de tinta o de papel, aunque sólo tendrá efecto el último. Para quitar los colores, tiene que situarse a la derecha del lugar en que los puso, y pulsar DELETE (CAPS SHIFT y 0). La primera vez que pulsa DELETE aparece un interrogante a la izquierda del cursor. La segunda vez el interrogante desaparece al mismo tiempo que el color (del papel o de la tinta) que acaba de borrar. Use DELETE una vez más y aparecerá otro interrogante. Haga DELETE por última vez y el otro color desaparecerá, obteniendo así el color y la tinta originales.

Buscar estos códigos escondidos, puede ser muy difícil. Aquí hay un ejemplo de cómo uno de ellos puede hacer que un programa aparente no funcionara bien.

```
300 IF INKEY$ = "c" THEN GOTO 1000
```

Esta línea pertenece a un programa que tenía preparado para este libro. La diferencia entre lo que se supone que tiene que hacer y lo que hace realmente no es evidente en absoluto. De algún modo, se ha insertado un atributo de color entre las comillas, con la 'c', que simplemente deja el papel del mismo color en que está. El efecto es totalmente invisible, pero cuando se ejecuta el programa, no se reconoce la 'c' como el carácter que hace cumplir la condición. Una vez descubierto y borrado el atributo escondido, el programa funcionó perfectamente.

UN PEQUEÑO TRUCO: LOS LISTADOS INVISIBLES

Como habrá observado, es perfectamente posible, poner el papel y la tinta del mismo color. Por ejemplo, poner tanto la tinta como el papel de color blanco. Escriba unas pocas líneas de

programa tales como estamentos REM. Ahora haga EDIT de la primera línea, y con el cursor en modo E pulse CAPS SHIFT y 7 simultáneamente (estoy suponiendo que el color general de su papel es el blanco). ¡La línea desaparece! Además una vez pulsado ENTER, no puede ver nada del listado exceptuando el primer número de línea.

LISTando a partir de la siguiente línea, el listado se hace visible (exceptuando la primera línea), pero usted puede repetir esto fácilmente con cada línea.

Ahora vamos a escribir otra vez frases en color en la pantalla:

```
10 PRINT "Hola mundo cruel"
20 PRINT "Estoy muy contento de verte"
```

Ahora baje la línea con EDIT, y mueva el cursor hasta colocarlo dentro de las comillas, antes de la H de "Hola". Ahora, con el cursor en modo E, cambie los atributos de color y tinta, tal como hizo antes. No se preocupe porque el resto del listado también tenga los colores. Simplemente ejecute el programa con RUN. Verá como sólo la primera línea aparece con los colores nuevos. Si ahora edita la línea de nuevo y justo antes de las segundas comillas vuelve a poner el papel tal como estaba, sólo aparecerá esta frase coloreada en el listado. Esto es una ventaja sobre el modo usual de definir el color localmente, con INK y PAPER, ya que de este modo, tienen las frases en el listado tal como aparecerán después en la ejecución del programa.

El segundo método, hace uso de los caracteres de control. Si usted mira el listado de códigos ASCII al final del libro, verá que los caracteres cuyos códigos van del 6 al 23 se usan para propósitos especiales. Cuando se le dice al Spectrum que los imprima, no producen caracteres sino un efecto, normalmente cambian la posición del cursor o el color del papel o la tinta. Mirando a la tabla puede ver que todas estas funciones son posibles utilizando los caracteres de control (imprimiéndolos mediante CHR$): Coma de PRINT, cursor izquierda, cursor derecha, ENTER,

INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, AT y TAB. Tanto EDIT como el cursor arriba y abajo y DELETE, no son en realidad utilizables.

La coma de PRINT, sirve para colocar los campos a imprimir en dos partes de la pantalla, la derecha y la izquierda:

```
PRINT "Hi","there","how","are","you?"
```

Esto también puede hacerse usando CHR$ 6:

```
PRINT"Hi";CHR$ 6; "there";CHR$ 6;"how";
CHR$ 6;"are";CHR$ 6 ;"you?"
```

Ya vimos cómo funcionaba CHR$ 8 al utilizar OVER. Esto hace que el cursor vuelva una posición hacia atrás, de manera que el próximo carácter se imprime encima (en inglés, OVER) del anterior. De un modo similar, CHR$ 9 mueve el cursor, una posición hacia adelante.

```
PRINT OVER 1; "O"; CHR$ 8; "/"
PRINT "G";CHR$ 9: "oof"
```

Como puede ver, CHR$ 8 es muy útil para sobreimprimir caracteres, mientras que CHR$ 9 simplemente mueve el cursor donde hubiera ido a parar de todos modos. CHR$ también es muy útil, ya que hace que el campo a imprimir lo haga en la próxima línea, es como mover el carro de la máquina de escribir.

```
PRINT "line 1";CHR$ 13;"line 2"
```

El color del papel y la tinta, también se pueden cambiar con CHR$. Aquí tiene un ejemplo (primero "normal" y luego usando los caracteres de control):

```
10 PRINT INK 2; PAPER 6; "NORMAL"
20 PRINT CHR$ 16; CHR$ 2; CHR$ 17;
   CHR$ 6; "CHR$ version"
```

Ejecute este programa y verá que las dos líneas tienen los mismos atributos de color. FLASH, INVERSE, BRIGHT

y OVER, también pueden utilizarse con CHR$ de un modo igualmente fácil:

```
10 PRINT CHR$ 18;CHR$ 1;CHR$ 16;CHR$ 2;
   CHR$ 17;CHR$ 6; "PANIC!"
```

Habrá observado que para usar este método, primero se escribe CHR$ (código) del atributo (por ejemplo CHR$ 17 para PAPER) y luego otra vez CHR$ seguido del código correspondiente. 0 a 7 para un color o bien un 1 o un 0 para la presencia o ausencia de un atributo.

Por supuesto que CHR$ 22 y CHR$ 23 hacen la función de TAB y AT respectivamente.

Usted debe estar pensando. "Todo esto está muy bien, pero ¿para qué quiero usar estos caracteres de control si parece más largo de escribir que los comandos directos como INK o PAPER?" Bien primeramente, usted no puede poner comandos como "backspace" (espacio hacia atrás). o ENTER en una línea PRINT si no es con CHR$ (al menos de un modo más fácil). Segundo: el uso de CHR$ le permite asignar a una variable de cadena todo un conjunto de textos, caracteres de control (colores, FLASH, etc.) y gráficos. Esta variable puede ser impresa en cualquier sentencia PRINT. Esto no se podría hacer sin disponer de CHR$, y es muy útil. Poner un trozo de texto, con sus colores, tabulaciones y gráficos en una cadena es una manera mucho más limpia y rápida que imprimir estas líneas en la pantalla usando las líneas normales de la sentencia PRINT. Esta diferencia de velocidad puede llegar a hacer que un juego escrito en BASIC sea aceptable o por el contrario tan lento que ya no sea interesante.

Como un ejemplo muy simple, vea cómo podría llamar a esta variable cuando fuera necesitada en el programa:

```
10 LET A$ = CHR$ 16 + CHR$ 2 + CHR$ 17 +
CHR$ 6 + "PUNTUACIÓN:"
```

Como puede ver esto produce la impresión en cualquier parte (usando PRINT AT) de la palabra "puntuación". Hubiera sido igual de fácil hacer que parpadeara (con FLASH). Observe también que cuando coloca algún CHR$ en una cadena tiene que usar '+' para separarlos en lugar del punto y coma. Note también, que la longitud de la

cadena no tiene límite, por lo tanto, es muy fácil almacenar toda una pantalla llena de texto con sus atributos de color, etc. en una variable de cadena.

Este método es sin embargo bastante lento, para agilizarlo podemos utilizar de nuevo SCREEN$, que tiene dos funciones diferentes en el Spectrum. Primero, SCREEN$ puede ser utilizado para guardar una pantalla llena de información en el cassette, y luego recuperarla de nuevo. Aquí hay un ejemplo de esto:

```
10 PAPER 3:INK 9:CLS
20 FOR N = 1 TO 64
30 PRINT " SCREEN$↑";
40 NEXT N
50 SAVE "SCREEN" SCREEN$
60 CLS
70 LOAD "SCREEN" SCREEN$
```

Ejecute este programa, y una vez esté la pantalla casi llena de "SCREEN$" aparecerá el mensaje "start tape, then press any key". Compruebe que las clavijas MIC del Spectrum y del cassette están conectadas y que dispone de una cinta libre, lista para grabar. Ponga en marcha el cassette y pulse una tecla cualquiera y cuando haya terminado se borrará la pantalla (línea 60) y el programa esperará a que recupere de la cinta lo que había en la pantalla (rebobine la cinta y conecte las clavijas EAR antes de poner en marcha el cassette). Como puede ver, su pantalla se vuelve a llenar de "SCREEN$". Primero aparecen los primeros bytes de las ocho primeras líneas, luego los segundos, etc. Después ocurre lo mismo con las ocho siguientes y por último se graban los atributos de color cambiando a magenta y blanco si es que no estaban puestos. Para observar este efecto más claramente añada la siguiente línea al programa:

```
55 PAPER 7: INK 0
```

Mientras que todo esto va muy bien para guardar una pantalla de gráficos que usted haya creado, no es muy práctico, pues para utilizarlo en un programa hay que detener éste y avisar a la persona que lo usa que recupere la pantalla de la cinta. Para solucionar esto, tenemos la segun-

da función de SCREEN$. Esta función ya la utilizamos, cuando detectábamos qué carácter había en una posición determinada de la pantalla. Para guardar una pantalla llena de información (o sólo una parte) puede usar SCREEN$ (x, y) para colocar los caracteres que hay en la pantalla en una variable de cadena (un conjunto de 24 por 32) y luego simplemente imprimir la cadena en la posición 0,0. Aquí tenemos un ejemplo que almacena e imprime las cinco primeras líneas de la pantalla:

```
10 LET X = 1
20 FOR A = 1 TO 40
30 PRINT "PCW rools "
40 NEXT A
50 DIMS$ (32*5)
60 FOR R = 1 TO 5
70 FOR C = 1 TO 32
80 LET S$(X) = SCREEN$(R,C)
85 LET X = X + 1
90 NEXT C:NEXT R
100 CLS
110 PRINT AT 0,0;S$
```

Como puede ver no es necesario calcular cuántos caracteres hay en las cinco primeras líneas de la pantalla, simplemente necesita poner 32*5. Observe que también ha creado una variable X para determinar la asignación de los caracteres a la cadena S$.

Por supuesto que este método no permite guardar de un modo fácil, los atributos de color, FLASH o BRIGHT. Para hacerlo, tiene dos posibilidades, por ejemplo puede definir una nueva variable numérica en la que se guarden los atributos de los caracteres que está almacenando y luego colocarlos (mediante POKE) directamente en el fichero de atributos (mire el mapa de memoria del apéndice para ver dónde se encuentra). Por otro lado, puede usar los caracteres de control descritos anteriormente.

Para hacer lo primero piense que necesita determinar los atributos de cada uno de los caracteres que va a almacenar. Esto puede hacerlo usando ATTR. Recuerde que ATTR, como SCREEN$, va seguido de las coordenadas de la posición en que se encuentra el carácter, encerradas entre paréntesis. ATTR devuelve un número que está com-

puesto de otros cuatro que le relacionan con FLASH, BRIGHT, INK y PAPER. El número se compone de:

128 si el cuadrado está en FLASH
64 si el cuadrado está en BRIGHT
8 multiplicado por el código del color de PAPER
el color de INK

A primera vista, la elección de estos números puede parecer un poco aleatoria. Sin embargo, esta elección, aparece mucho más clara si consideramos los números en forma binaria. Recuerde que cada columna corresponde a una potencia de 2. Es decir las columnas son:

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
qué son:	128	64	32	16	8	4	2	1

En el lenguaje de los ordenadores, estas ocho columnas se llaman **bits,** y como puede ver, el bit ocho es 128, el 7 es 64, etc. Al hablar de los atributos podemos pensar en estos ocho bits como un todo (un byte):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
FLASH	BRIGHT	PAPER			INK		

Con esto tenemos que los bits 0, 1 y 2 indican el color de la tinta, los bits 3, 4 y 5 el color del papel y los bits 6 y 7 le dicen al Spectrum si BRIGHT y FLASH están encendidos o apagados. Veamos un ejemplo concreto para ver cómo trabaja esto:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	1
FLASH	BRIGHT	PAPER			INK		

Inmediatamente puede ver qué FLASH está encendido (el bit 7 es 1), pero que no hay brillo (BRIGHT) porque el bit 6 es Ø. Los bits 5, 4 y 3 dan el número 1Ø1, que en decimal es (1*32) + (Ø*16) + (1*8) = 4Ø. Si divide 4Ø por ocho se obtiene 5 lo cual indica que el color del papel es azul claro o cyan. Ahora el color de la tinta. El número obtenido es el Ø11 que actuando del mismo modo: (Ø*4) + (1*2) + (1*1) nos da 3, por lo tanto el color de la tinta es magenta. Y el número completo es:

$$(128) + (4Ø) + (3) = 171$$

Para comprobarlo escriba:

```
PRINT INK 3; PAPER 5; FLASH 1;"anybody"
```

Entonces suponiendo que al menos parte de la frase se imprime en la esquina superior izquierda de la pantalla, escriba sin número de línea:

```
PRINT ATTR (Ø,Ø)
```

Esto debería dar 171.

Volviendo a nuestro problema conociendo ya como funciona ATTR, vamos a ver cómo podemos almacenar no sólo los caracteres de una parte de la pantalla sino también sus atributos.

Una solución es inmediata, si recordamos que se puede poner color en la pantalla usando los caracteres de control CHR$ 16 y CHR$ 17. Por ejemplo, este programa, almacenará tanto el texto como los atributos de las dos primeras líneas de la pantalla:

```
  10 REM CREACION DE DOS LINEAS DE CARAC
TERES
  20 PRINT  PAPER 4; INK 5;AT 0,0;"*****
**************************"
  30 PRINT  PAPER 6; INK 9;AT 1,0;",,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,"
  40 REM COLOCACION DE LOS CARACTERES Y
SUS ATRIBUTOS EN LA CADENA A$
  50 LET A$=""
```

```
 60 FOR X=0 TO 1
 70 FOR Y=0 TO 31
 80 LET B$=CHR$ 17+CHR$ INT (ATTR (X,Y)
/8)+CHR$ 16+CHR$ (ATTR (X,Y)-8*INT (ATTR
 (X,Y)/8))
 90 LET C$=CHR$ CODE SCREEN$ (X,Y)
100 LET A$=A$+B$+C$
110 NEXT Y: NEXT X
120 CLS
130 REM RE-PRINT 2 LINES
140 PRINT AT 0,0;A$
```

Por supuesto que no sería nada difícil, añadir a este programa lo necesario para detectar y almacenar los atributos de FLASH y BRIGHT, simplemente añadiendo más caracteres de control, junto con el cálculo conveniente en la línea 70.

Habrá observado, que en este programa uso INK 9. Esto coloca la tinta que mejor contrasta con el color del papel (y siempre es blanca o negra). INK 8 también es válido y lo que hace es imprimir en el color de tinta (o de papel si se pone PAPER 8) que se usó la última vez que se imprimió algo en ese cuadrado.

COLORES

Ya habrá observado, que mientras los puntos se pueden colocar (PLOT) en un nivel de resolución de 256 × 176, los colores sólo se pueden colocar por cuadrados con una resolución de 32 × 22. Un ejemplo rápido de esto último consiste en añadir un comando PAPER a una sentencia CIRCLE de este modo:

```
CIRCLE PAPER 4; 128,88,50
```

Observará que mientras la línea del círculo se dibuja en alta resolución, el color del papel, llena todo el cuadrado de un carácter. Sólo se puede tener un color de tinta y uno de papel en cada cuadrado, lo que significa que el Spectrum no permite la alta resolución en color.

Una vez dicho esto, de que sólo se puede tener un color

de papel y de tinta por cuadrado, diremos también que el Spectrum tiene una peculiaridad que parece que pueda haber dos colores de tinta en un mismo cuadrado:

```
 10 REM DOS TINTAS POR CARACTER
 15 REM
 20 BORDER 1
 30 FOR x=0 TO 7
 40 FOR y=0 TO 7
 50 IF y=x THEN  NEXT y
 55 PAUSE 100
 60 INK y: PAPER x: CLS
 70 PLOT 0,100: DRAW 255,0
 80 PLOT 0,98: DRAW 255,0
 90 PRINT  OVER 1;AT 9,5;"■■"
100 PLOT 0,70: DRAW 255,0
110 PLOT 0,67: DRAW 255,0
120 PRINT  OVER 1;AT 13,10;"■■"
130 NEXT y: NEXT x
```

Como ve esta peculiaridad parece contradecir el ejemplo anterior, ya que pueden aparecer líneas de distinto color en el mismo cuadrado dependiendo únicamente de la distancia en puntos que haya de la una a la otra. Por cierto a los puntos de la resolución, se les conoce con el nombre de "pixels".

128 COLORES

Hemos visto que el Spectrum puede producir 8 colores distintos, es decir, seis colores, más el blanco y el negro. ¿Pero, qué hay sobre 64 colores?, ¿y sobre 128? Pues bien, es posible generar muchos más colores mezclando los ocho ya existentes a nivel de pixels. Para hacer esto, necesita crear un gráfico definido por usted, que parece un pequeño tablero de ajedrez. Para ello vea el programa de creación de gráficos que se muestra más tarde. Haciendo el papel y la tinta de dos colores distintos aparecen mezclados, ya que los puntos están muy juntos. Este programa, le muestra toda la gama de colores que se pueden obtener haciendo esto. El número puede ser doblado si añadimos el factor brillo, creando versiones más apagadas o brillantes de cada color. Tenga en cuenta que la A de

las líneas 70 y 120 es un carácter gráfico de los definidos por el usuario.

```
  2 REM   128 colores
  5 REM juegos de colores
 10 FOR a=0 TO 6 STEP 2
 20 POKE USR "a"+a,BIN 01010101
 25 POKE USR "a"+a+1,BIN 10101010
 30 NEXT a
 40 FOR p=0 TO 7
 50 FOR i=0 TO 7
 60 FOR b=0 TO 1
 70 PRINT  PAPER p; INK i; BRIGHT b;"aa
";p;i;: NEXT b
 80 NEXT i: NEXT p
 90 REM combinaciones de color
100 FOR p=0 TO 7: FOR i=0 TO 7: FOR b=0
TO 1
110 FOR y=0 TO 31
120 PRINT  PAPER p; INK i; BRIGHT b;"aa
aaaaaaaaaaaaaaaaaaaaaaaaaaaa";
130 NEXT y
140 POKE 23692,255
150 NEXT b: NEXT i: NEXT p
```

EL FICHERO DE LOS ATRIBUTOS

Hay un modo más de cambiar los colores en el Spectrum que consiste en colocarlos directamente (mediante POKE) en un área de la memoria llamada fichero de atributos (atribute file). Este fichero, se encuentra justo detrás de la memoria de pantalla en las posiciones que van desde la 22528 hasta la 23296. Un poco de matemáticas nos revelan que esta área de la memoria consta de 768 posiciones que es por supuesto 32 × 24. En otras palabras, hay una posición de memoria en el fichero de atributos, para cada posición de la pantalla. Normalmente, sólo se usan las 22 líneas superiores, quedando las dos inferiores para mensajes de error, INPUT, etc.

Pero "colocar directamente en el fichero de atributos" ¿qué significa exactamente? Pues bien, **POKE** describe el proceso de poner un trozo de información (llamado un byte, ¿recuerda?) en una posición de memoria especificada. Su opuesto, es **PEEK,** ya que cuando al Spectrum le

damos la instrucción PRINT PEEK (número de posición) nos imprime el valor del byte que se encuentra en esa posición de memoria. Esto es lo que ocurre realmente cuando usamos ATTR para obtener los atributos de un carácter en la pantalla. Para clarificar esto un poco más, comprobaremos que ambas cosas son lo mismo:

```
10 PRINT INK2;PAPER6;AT0,0;"*"
20 PRINT ATTR(0,0)
30 PRINT PEEK 22528
```

Ejecute esto, y verá que además de obtener una estrella roja sobre fondo amarillo, tanto la línea de ATTR como la de PRINT PEEK devuelven el valor de 50. Ahora para ver cómo se puede usar el fichero de atributos, escriba lo siguiente:

```
POKE 22528,42
```

Mágicamente (e instantáneamente) la estrella está en rojo sobre fondo azul claro. Intente hacer lo mismo pero con otros números (hasta 255) para ver el efecto que esto produce sobre el recuadro en que se encuentra el asterisco (que como habrá deducido, corresponde a la posición 22528 de la memoria, claro dentro del fichero de atributos). Observe que ponga lo que ponga en esta posición, el asterisco no desaparece, aunque lo puede parecer si usted hace que el color del papel sea el mismo que el de la tinta.

Un uso inmediato de lo que acabamos de ver, es cambiar los colores del papel y la tinta (ya sea en toda la pantalla o sólo una parte), sin alterar para nada el texto ni los gráficos. Recordará que cuando esto se efectúa mediante un comando, ya sea local o global, se destruye el texto o el gráfico que había en la posición.

Aquí hay una rutina muy simple para cambiar el color y la tinta a los dos colores escogidos. Aunque lo hace para toda la pantalla, es muy fácil cambiarla para que sólo lo haga sobre una parte.

```
10 REM CAMBIO DE ATRIBUTO
20 FOR A = 22528 TO 23295
30 POKE A, N
40 NEXT A
```

(Observe que N es el número de ATTR. Por ejemplo, N = 47 produce papel azul claro y tinta blanca).

```
10 REM CAMBIO DE PAPEL
20 DIM A$ (704)
30 PRINT AT 10,0; INK 7; PAPER 7; "¡Sorpresa!"
40 PAUSE 50
50 PRINT OVER 1; AT 0,0; A$
```

Aparte del método que usa POKE, existe otra manera de cambiar los atributos que consiste en usar PRINT OVER, imprimiendo una cadena de espacios del tamaño de la pantalla para cambiar el papel. Observe que para cambiar la tinta tendrá que hacer PRINT con una cadena de cuadrados (los de la tecla 8 en modo gráfico). Este ejemplo nos muestra el uso de PRINT OVER:
En la línea 80, también se podría incluir BRIGHT 1, FLASH 1 o INVERSE 1. Este último método, es mucho más rápido si usted sólo desea cambiar el papel y una vez se ha creado A$. Más adelante en este libro, veremos una rutina en código máquina para cambiar el color del papel y la tinta casi instantáneamente sin destruir el texto.

DIBUJOS EN COLOR

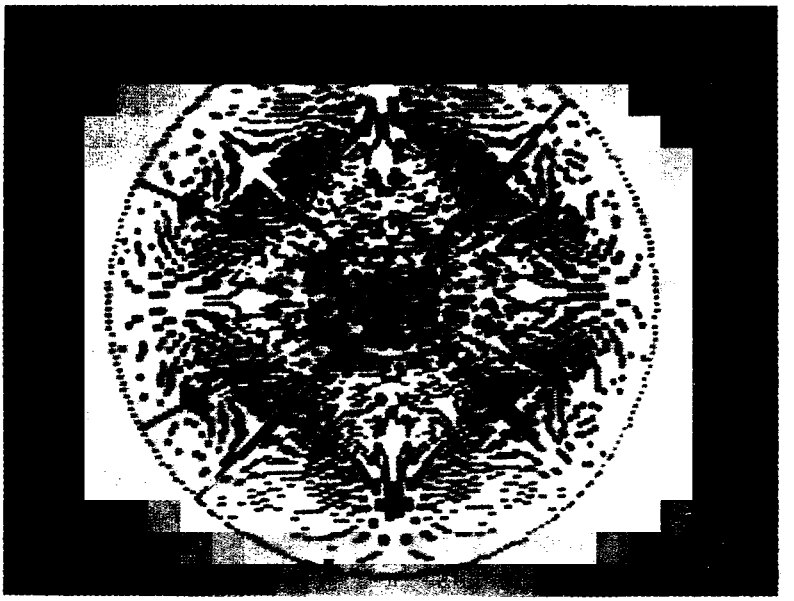
Vamos a dibujar algo en alta resolución en color. ¿Qué hay más indicado que un arco iris?

```
  5 REM  ARCO-IRIS
 10 LET X=1
 20 DIM A(6): GO SUB 120
 30 LET K=1
 40 PAPER 7: BORDER 0: CLS
 50 FOR Y=160 TO 250 STEP 15
 60 FOR J=1 TO 5
 70 INK A(K): PLOT Y+J,0: DRAW -Y,Y-80+
J,PI/2
 80 NEXT J
 90 LET K=K+1
100 IF K=7 THEN  GO TO 200
110 NEXT Y
120 LET A(1)=3: LET A(2)=1: LET A(3)=5:
LET A(4)=4: LET A(5)=6: LET A(6)=2
```

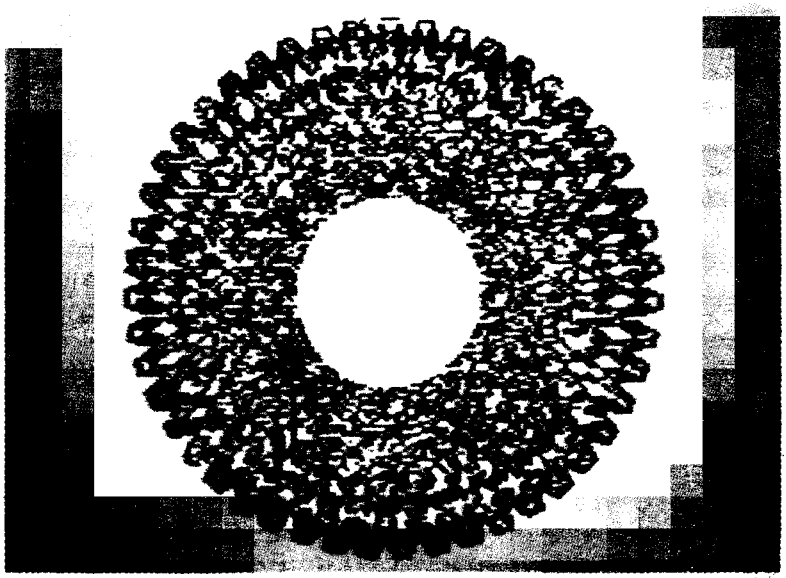
```
 130 RETURN
 200 PRINT  PAPER 7;AT 19,5; INK 3;"S";
INK 1;"P"; INK 5;"E"; INK 4;"CT"; INK 6;
"R"; INK 2;"UM"
```

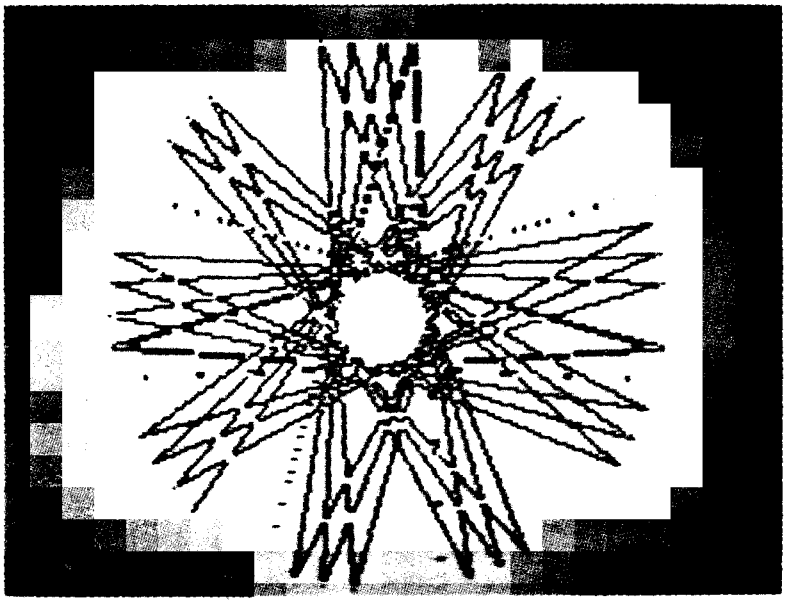
Observe cómo el hecho de que DRAW dibuja arcos, se utiliza aquí para dibujar las curvas del arco iris. Observe también que las curvas están a distancia de un cuadrado para evitar el tener más de una tinta en cada cuadrado. Dibujar atractivos cuadros en color es sorprendentemente fácil con el Spectrum. El siguiente programa sólo tiene una línea de longitud, pero usted se sorprenderá de lo que hace. Cambie el papel a amarillo (6) y la tinta a azul oscuro (1). Esta rutina tan simple fue desarrollada por Andrew Glaister, con algunas modificaciones mías:

```
1 PLOT 65,27: DRAW OVER 1; 120,120,59↑3*PI
```

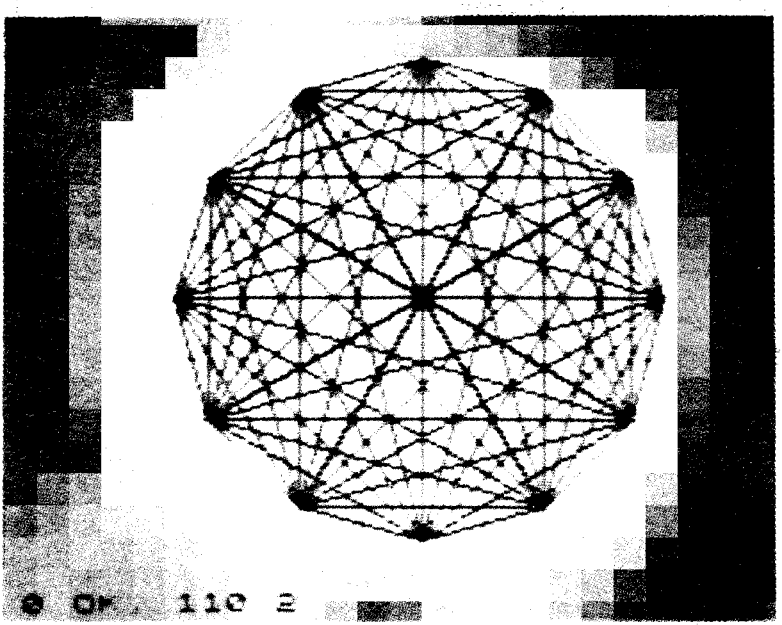


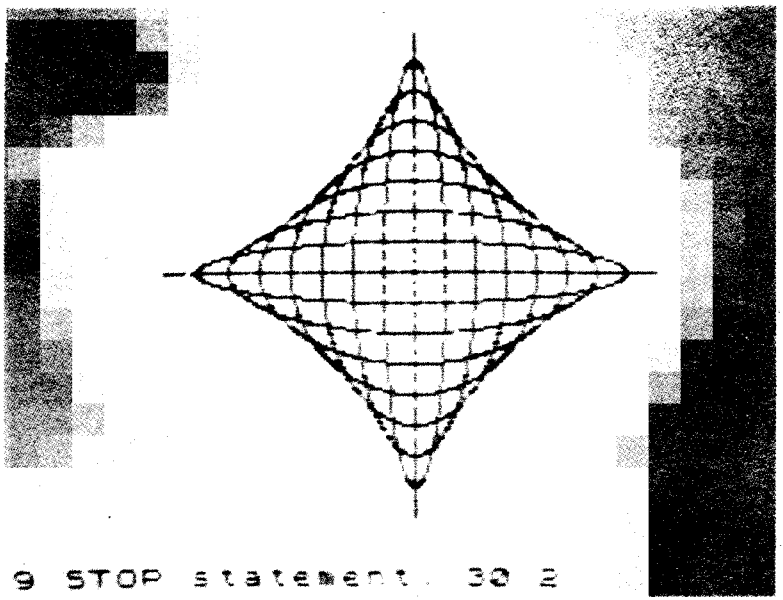
Pruebe a variar el número 59 de la línea y juegue con las potencias 2 y 3. Aquí hay dos ejemplos más que demuestran que este programa es muy versátil:

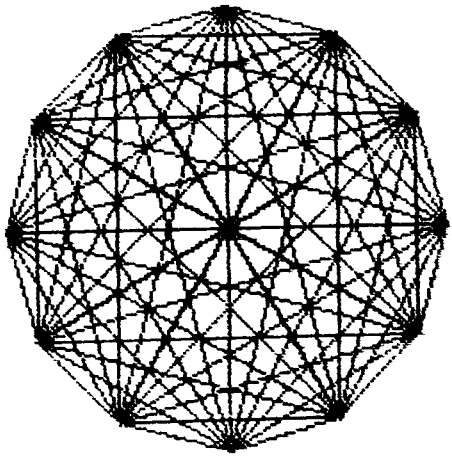




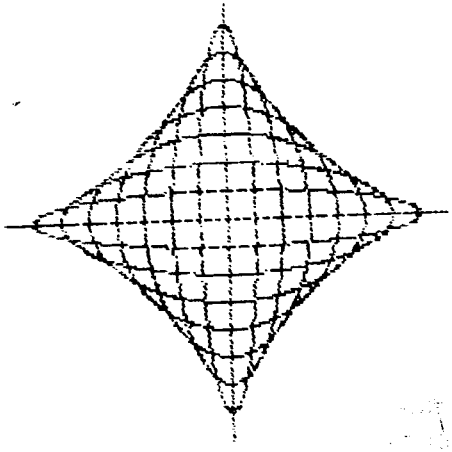
Otros dibujos extremadamente atractivos se pueden generar con sólo unas pocas líneas de BASIC. Aquí tenemos algunos ejemplos hechos por G. Scott:



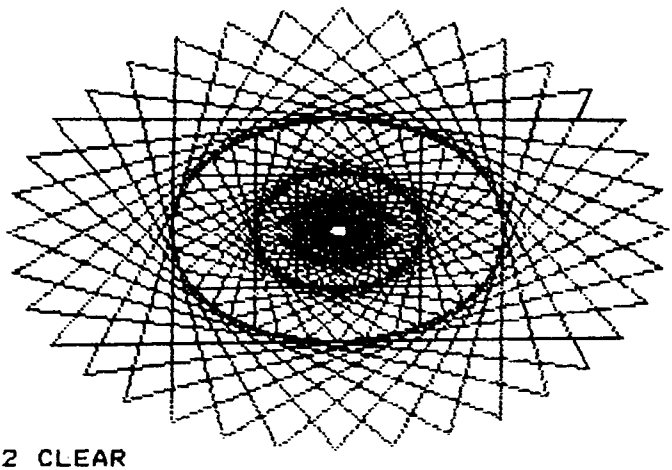




```
  2 CLEAR
  5 DIM a(12): DIM b(12)
 10 FOR n=1 TO 12
 20 LET k=n/6*PI
 30 LET a(n)=128+80*SIN k: LET b(n)=86+
80*COS k
 40 PLOT a(n),b(n)
 50 NEXT n
 60 FOR n=1 TO 12
 70 FOR m=1 TO 12
 80 LET ox=a(m)-a(n)
 90 LET oy=b(m)-b(n)
100 PLOT a(n),b(n): DRAW ox,oy
110 NEXT m: NEXT n
```



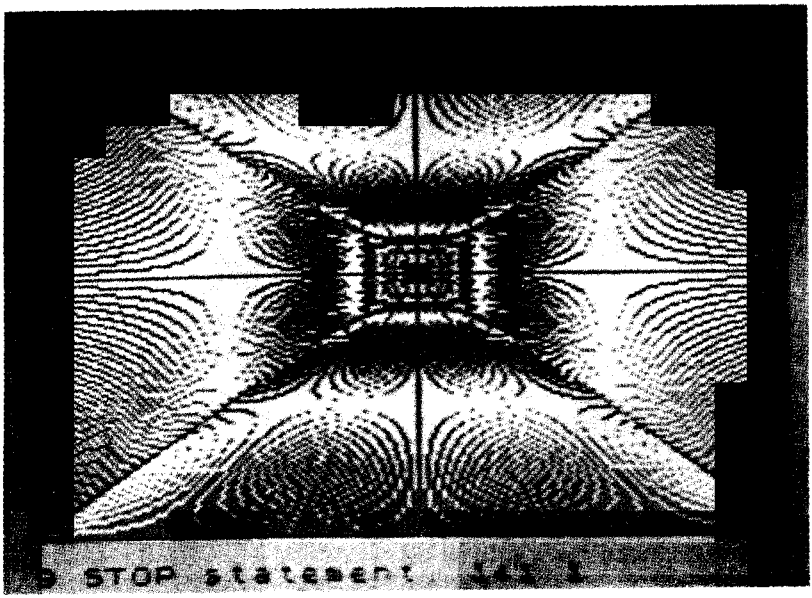
```
  2 CLEAR
  3 INK 2
  5 LET x=0: LET y=80
 10 FOR n=0 TO 2*PI STEP PI/180
 15 PLOT 128+x*SIN n,87+y*COS n
 20 NEXT n
 25 LET x=x+10: LET y=y-10
 30 IF y=-10 THEN  STOP
 40 GO TO 10
```



```
  2 CLEAR
 10 DIM a(36): DIM b(36)
 15 LET l=120: LET j=80
 17 FOR h=1 TO 5
 20 FOR n=1 TO 36
 30 LET k=n/18*PI
 40 LET a(n)=128+l*SIN k: LET b(n)=88+j
*COS k
 45 PLOT a(n),b(n)
 50 NEXT n
 60 FOR n=1 TO 36
 65 LET m=n+12
 70 IF m>36 THEN  LET m=m-36
 80 LET ox=a(m)-a(n)
 90 LET oy=b(m)-b(n)
100 PLOT a(n),b(n): DRAW ox,oy
110 NEXT n
120 LET l=l/2: LET j=j/2
125 NEXT h
```

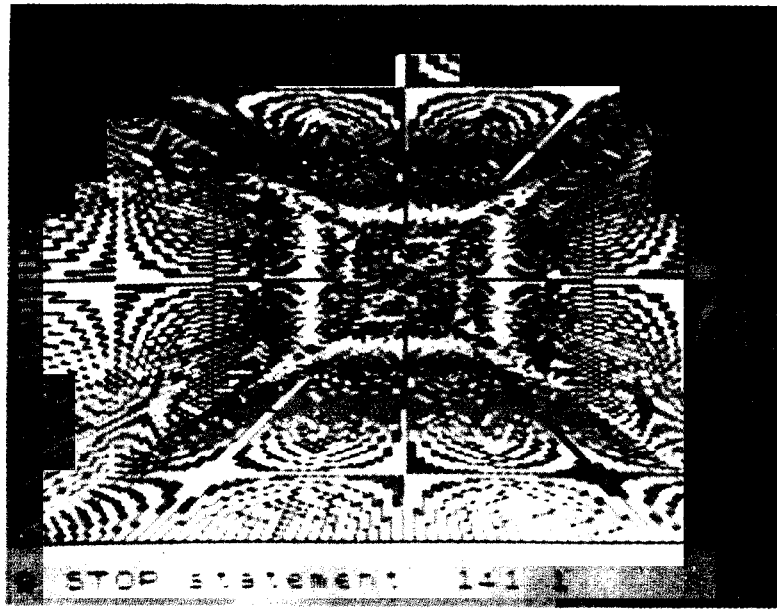
También es muy fácil llenar la pantalla de impresionantes dibujos, simplemente explotando la manera que tiene el

Spectrum de dibujar líneas. Aquí hay dos rutinas que producen resultados muy similares por métodos distintos:




```
  2 REM INTERFERENCIAS
 10 OVER 1
 20 LET z=1
 30 LET i=(RND*5)+1
 40 LET p=(RND*5)+1
 50 IF INT i=INT p THEN  GO TO 30
 60 PAPER p: INK i: CLS
 65 FOR a=1 TO 2
 70 FOR x=0 TO 254 STEP 2
 80 PLOT 128,88: DRAW (-127*z)+(x*z),z*
-87
 90 NEXT x
100 FOR y=0 TO 175 STEP 2
110 PLOT 128,88: DRAW 127*z,z*-87+(y*z)
120 NEXT y
130 LET z=-z
140 NEXT a: PAUSE 100: GO TO 20
```

Añadiendo OVER 1 con STEP 0.8 produce resultados aún más impresionantes:



Y una manera muy simple de obtener un resultado similar:

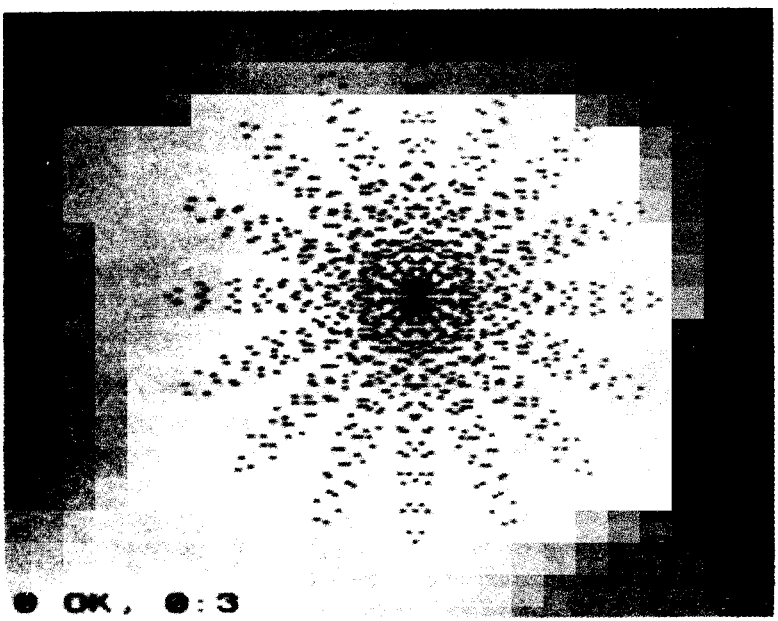
```
 5 REM  TUNEL DEL TIEMPO
10 FOR A=0 TO 255
20 PLOT A,0
30 DRAW  OVER 1;2*(127.5-A),175
40 PLOT 0,A*175/255
50 DRAW  OVER 1;255,2*(87.5-A*175/255)
60 NEXT A
```

Lo que es bastante interesante sobre estos dibujos es que la "nieve" de la pantalla puede producir un efecto muy agradable si la combinación de colores es buena.

En el próximo programa, se dibuja un copo de nieve con brazos radiales que parten del centro. Pruebe a alterar algunas líneas para cambiar el número de brazos del copo de nieve, etc.

```
10 REM Copo de nieve
20 INK 7: BORDER 3: PAPER 1: CLS
30 LET p$=" ": LET t$=" "
40 FOR a=0 TO 702
```

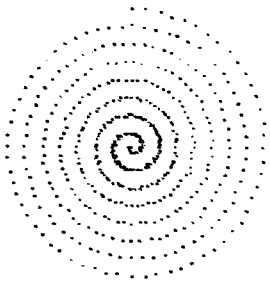
```
 50 LET p$=p$+t$
 60 PRINT ".";: NEXT a
 70 REM Crea una cadena de espacios del
tamanno de la pantalla
 80 CLS
 90 LET flag=0
100 FOR p=20 TO 80 STEP 1
110 LET y=0: LET z=PI
120 LET c=100
130 LET e=(z-y)/c
140 FOR q=y TO z STEP e
150 LET s=p*COS (q*6)
160 LET h=s*SIN q
170 LET v=s*COS q
180 PLOT 128+h,88+v
190 NEXT q
200 LET flag=flag+1
210 IF flag=101 THEN  GO TO 230
220 GO TO 150
230 LET flag=0: NEXT p
240 LET col=INT (RND*6)+1
250 PRINT AT 0,0; PAPER col; OVER 1;p$
260 PAUSE 100: GO TO 240
```



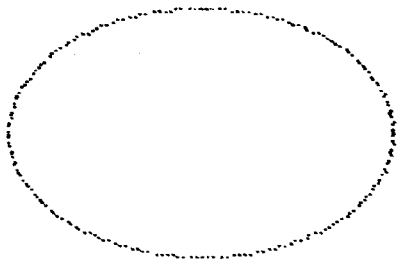
Simplemente dibujar círculos de diferentes colores puede producir un resultado muy bonito:

```
 2 REM circulos
 5 PAPER 0: CLS
10 LET x=(234*RND)+10
20 LET y=(154*RND)+10
30 LET z=(5*RND)+1
40 CIRCLE  INK z;x,y,10
50 GO TO 10
```

También se pueden dibujar fácilmente elipses y espirales:



```
 5 REM  espiral
10 BORDER 1: PAPER 0: INK 7: CLS
20 LET r=50
30 FOR f=0 TO 500
40 LET a=(RND*7)+1
50 LET r=r-0.1
60 PLOT  INK a;128+r*SIN (f/32*PI),88+
r*COS (f/32*PI)
70 NEXT f
```



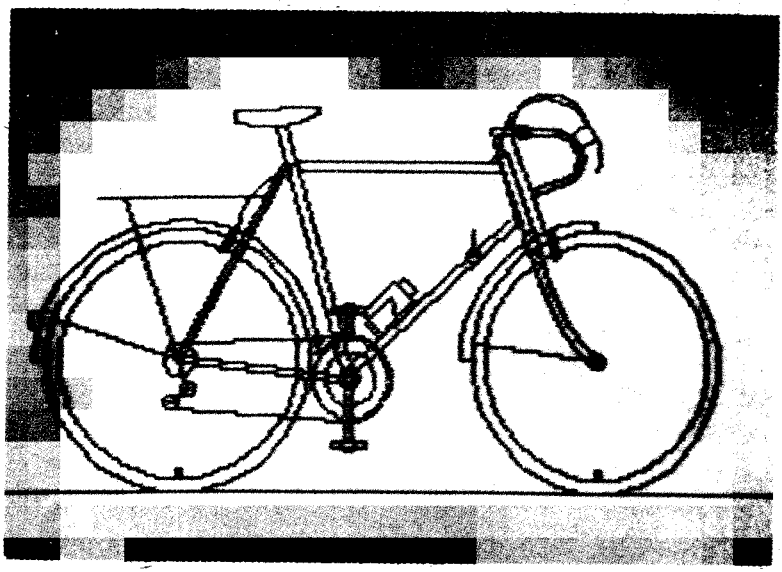
```
10 REM DIBUJA UN OVALO
20 REM LOS RADIOS DEBEN SER MENORES A
LOS LIMITES DESDE EL CENTRO
30 REM LOS LIMITES DEL CENTRO SON 128,88
```

```
  40 BORDER 1: PAPER 5: INK 1: CLS
  50 INPUT "DOS RADIOS ?";R1;R2
  60 FOR N=0 TO 200
  70 PLOT 128+R1*SIN (N/100*PI),88+R2*CO
S (N/100*PI)
  80 NEXT N
```

También es posible hacer dibujos animados en alta resolución:

```
  10 REM  pendulo
  20 BORDER 1: PAPER 1: INK 7: CLS
  30 FOR x=1 TO 3000 STEP 0.5
  40 PLOT 128,160: DRAW -120*SIN (x/10),
110,2*PI
  50 NEXT x
```

Para demostrar que el Spectrum también puede llegar muy lejos en cuanto a dibujo realista se refiere, aquí tiene una escelente imagen de una bicicleta de carreras hecha por G. Scott y R. Turner.



```
2 REM *****La bicicleta*****
3 REM  G.J.Scott&R.I.Turner
4 REM *****La bicicleta*****
```

```
  10 CIRCLE 60,70,40: CIRCLE 60,70,5
  15 PLOT 103,70: DRAW -86,0,PI: DRAW 3,
0
  16 PLOT 60,70: DRAW -38,13
  20 CIRCLE 190,70,40
  25 PLOT 150,70: DRAW -3,0: DRAW 43,43,
-PI/2: DRAW 0,-3
  26 PLOT 190,70: DRAW -38,5
  30 CIRCLE 60,70,36: CIRCLE 190,70,36
  35 PLOT 0,30: DRAW 255,0
  40 CIRCLE 112,64,12: CIRCLE 190,70,2
  45 PLOT 18,80: DRAW -2,0: DRAW 0,5: DR
AW 3,0
  50 PLOT 60,69: DRAW 50,-6
  55 PLOT 60,71: DRAW 50,-6
  70 CIRCLE 112,64,3: CIRCLE 112,64,8
  80 PLOT 59,70: DRAW 32,60
  90 PLOT 61,70: DRAW 32,60
 100 DRAW -1,0: DRAW 19,-67
 110 DRAW 3,1: DRAW -19,67
 120 DRAW 64,0
 130 PLOT 97,128: DRAW 63,0
 140 DRAW -1,5: DRAW 8,-24
 150 DRAW 3,1: DRAW -8,24
 160 DRAW -3,-1: DRAW 6,-21
 170 DRAW -52,-47: DRAW -3,1
 180 DRAW 52,47
 190 CIRCLE 63,61,2: CIRCLE 58,57,2
 200 PLOT 60,70: DRAW 3,-9: DRAW -5,-4
 210 PLOT 58,55: DRAW 53,-4
 220 PLOT 112,51: DRAW 0,26,PI
 230 DRAW -47,-2
 240 CIRCLE 151,102,2: PLOT 151,102: DRA
W 0,8
 250 CIRCLE 170,105,2: PLOT 169,103: DRA
W 4,-12: DRAW 16,-22,PI/5
 260 PLOT 172,103: DRAW 4,-12: DRAW 15,-
20,PI/5
 270 PLOT 168,104: DRAW -1,4: PLOT 172,1
04: DRAW -1,5
 280 PLOT 102,72: DRAW 7,4,-PI/6
 290 PLOT 102,74: DRAW 7,4,-PI/5
 300 PLOT 111,63: DRAW 0,-17
 310 PLOT 113,63: DRAW 0,-17
```

```
 320 CIRCLE 112,44,2
 330 PLOT 107,45: DRAW 10,0
 340 DRAW 0,-2: DRAW -10,0: DRAW 0,2
 350 PLOT 111,78: DRAW 0,8
 360 PLOT 113,78: DRAW 0,8
 370 CIRCLE 112,85,2: PLOT 107,84: DRAW
10,0: DRAW 0,2: DRAW -10,0: DRAW 0,-2
 380 PLOT 159,134: DRAW -2,8
 390 DRAW 21,0: DRAW 0,-2,-PI: DRAW -19,
0: DRAW 2,-6
 400 PLOT 167,141: CIRCLE 167,141,2
 410 PLOT 178,142: DRAW 0,-18,-PI
 420 DRAW -8,-3: DRAW 0,2: DRAW 8,3: DRA
W 0,14,PI
 430 PLOT 184,141: DRAW 4,3: DRAW 3,-4:
DRAW -4,-3
 440 PLOT 191,140: DRAW 0,-11,-PI/4
 450 PLOT 188,144: DRAW -28,-7,PI/1.2: D
RAW -8,-6,-PI/2
 455 PLOT 188,144: DRAW -24,-6,PI: DRAW
13,-37: DRAW 1,1: DRAW -3,8: DRAW 1,0: D
RAW 2,-7
 460 PLOT 92,130: DRAW -3,12
 470 DRAW 3,0: DRAW 3,-12: DRAW -3,0
 480 PLOT 76,143: DRAW 26,3,PI/4: DRAW 0
,2,-PI: DRAW -26,-1: DRAW 0,-4
 490 PLOT 95,130: DRAW -18,-20,PI/3
 500 DRAW -3,-7: DRAW -1,0: DRAW 3,7: DR
AW -1,0: DRAW -3,-7
 510 PLOT 85,120: DRAW -50,0: PLOT 60,70
: DRAW -17,50
 520 PLOT 122,78: DRAW -6,6: DRAW 10,10:
 DRAW 6,-6: DRAW -8,-8: DRAW 8,8: DRAW -
1,1: DRAW 2,2: DRAW -4,4: DRAW -1,-1
 530 PLOT 121,77: DRAW -4,4: DRAW 8,8: D
RAW 2,-7
 540 PLOT 60,34: DRAW 0,3: DRAW 1,0: DRA
W 0,-3
 550 PLOT 190,34: DRAW 0,3: DRAW 1,0: DR
AW 0,-3
```

Para finalizar esta colección de dibujos en alta resolución, un par de cosas que aunque el programa es muy corto, tardan un poco en ejecutarse:

```
  10 REM el sombrero de Merlin
  20 BORDER 2: PAPER 6: INK 2: CLS
  30 LET x=128: LET y=88
  40 LET m=0
  50 LET r1=70: LET r2=20
  60 FOR f=m TO 200-m
  70 PLOT x+r1*SIN (f/100*PI),y+r2*COS (
f/100*PI)
  80 NEXT f
  90 LET y=y-2
 100 LET r1=r1-1: LET r2=r2-1
 110 LET m=50: GO TO 60
```

```
  10 REM la isla magica
  20 REM necesita varias horas
 para ejecutarse
  30 LET x=126: LET y=40
  50 BORDER 1: PAPER 0: INK 7: CLS
  60 LET r1=70: LET r2=20
  70 FOR f=50 TO 150
  80 PLOT  OVER 1;x+r1*SIN (f/100*PI),y+
r2*COS (f/100*PI)
  90 NEXT f
 100 LET r1=r1-1: LET r2=r2-1
 120 GO TO 70
```

LOS LÍMITES DE DRAW

En los dibujos en alta resolución es necesario controlar que los puntos colocados en la pantalla no rebasen el área de PAPER. Es decir, que las coordenadas horizontales estén entre 0 y 255 y las verticales entre 0 y 175.

Dado que la línea que dibuja DRAW parte de un punto determinado (el último punto PLOTeado) y termina en otro también determinado (en la propia instrucción) usted debe asegurarse de que la coordenada horizontal del PLOT inicial más la primera coordenada de DRAW no exceda de 255. De un modo similar, las verticales, no deben sumar más de 175. Habiendo puesto el primer punto en el centro de la pantalla, sólo puede dibujar hasta los puntos ±127 horizontalmente y ±87 verticalmente. Estas reglas son la base de la mayoría de gráficos que imprimen puntos en posiciones aleatorias.

En algunos de estos gráficos en alta resolución los mejores colores consisten en tinta clara sobre fondo oscuro. Sin embargo, cuando se usa COPY para imprimir el dibujo en papel, nos encontramos con que éste pierde detalle y parece todo negro. El remedio es cambiar la tinta y el papel en toda la pantalla. Esto se puede hacer intercambiándolos mediante INVERSE, ya sea en cada línea o globalmente. También se puede intentar hacer uso de POKE para colocar los colores directamente en el fichero de atributos.

Para cambiar la tinta y el papel, puede usar la siguiente rutina (veremos más adelante que se puede hacer lo mismo en código máquina y que va mucho más rápido):

```
10 FOR A = 16384 TO 22528
20 POKE A,255 - PEEK A
30 NEXT A
```

Con todo lo que hemos visto sobre dibujo, se puede observar que el Spectrum tiene la suficiente potencia para hacer complicados diseños gráficos. Por ejemplo, considere este programa que crea triángulos sólidos de dimensiones dadas:

```
 1 REM TRIANGULO SOLIDO
 5 BORDER 1: PAPER 7: CLS
10 INPUT "COORDENADA"; X
20 INPUT "COORDENADA"; Y
30 INPUT "¿ALTURA?"; H
35 IF Y + H > 175 THEN GO TO 100
40 INPUT "¿LONGITUD?"; L
45 IF X +L < 255 THEN GO TO 100
50 INPUT "¿COLOR?"; C
55 CLS
60 FOR P = 0 TO H
70 PLOT X, Y: DRAW INK C; L, P
80 NEXT P
90 STOP
100 CLS: PRINT "DEMASIADO" '' "ENTER AGAIN"
110 GO TO 10
```

Esta rutina que ahora sigue, le permitirá dibujar lo que quiera en la pantalla de un modo similar a un "telesketch", y luego colorearlo. Observe que dado que los cuadrados de los caracteres determinan los distintos colores puede haber partes de su dibujo en que los colores se

salgan de la línea que deseamos que los envuelva (esto ocurre cuando la línea no coincide con un cuadrado). Por lo tanto es aconsejable hacer coincidir en lo posible, el dibujo con los cuadrados. Para ayudarle a hacer esto añada al siguiente programa otra rutina que convierta la pantalla en una parrilla que marque las líneas izquierdas de cada cuadrado verticalmente y las de abajo horizontalmente. Para ello sólo necesitará crear un gráfico, ya que el segundo ya existe (use SYMBOL SHIFT y Ø). El gráfico que necesita crear deberá tener todos los bytes iguales a 128, lo que lo hace muy fácil de crear (para ello puede ver el próximo apartado).

```
   2 BORDER 1: PAPER 7: INK 0: CLS : LET
  q=0
   3 PRINT AT 8,0;"pulsa 'S' para pintar
 ": PRINT AT 1,1;"Dibuja una figura usand
 o          las flechas";AT 4,0;" pulsa '0
 ' para borrar";AT 5,0 ;"'W' para volver
 a entrar en modo dibujo"
   4 PRINT ''"pulsa una tecla para empez
 ar": PAUSE 0: CLS : LET x=128: LET y=88
   5 DIM a$(32): PLOT  INK q;x,y
   6 IF INKEY$="5" THEN  LET x=x-1: PAUS
 E 25
   7 IF INKEY$="6" THEN  LET y=y-1: PAUS
 E 25
   8 IF INKEY$="7" THEN  LET y=y+1: PAUS
 E 25
   9 IF INKEY$="8" THEN  LET x=x+1: PAUS
 E 25
  10 IF INKEY$="s" THEN  GO TO 21
  11 IF INKEY$="0" THEN  LET q=6
  12 IF INKEY$="w" THEN  LET q=0
  13 GO TO 5
  15 PLOT 100,50: DRAW 50,50: DRAW -50,3
 0: DRAW 0,-80
  20 PRINT AT 0,0;"

                                  "
  30 INPUT "x coord?";x: INPUT "y coord?
 ";y: PLOT x,y: INPUT "OK?";q$: IF q$="n"
  THEN  GO TO 900
  35 GO TO 2000
```

```
 36 REM ****PINTANDO****
 40 INPUT "que tinta?";i
 50 PRINT AT 0,0;"Pintando..."
 60 LET g=y
 90 PRINT AT 1,0;a$
100 LET z=x
105 PLOT  INK i;z,y
110 LET z=z+1
115 IF POINT (z,y)=1 THEN  GO TO 200
120 IF POINT (z,y)=0 THEN  PLOT  INK i;
z,y
130 GO TO 105
200 LET z=x
210 LET z=z-1
220 IF POINT (z,y)=1 THEN  GO TO 300
230 IF POINT (z,y)=0 THEN  PLOT  INK i;
z,y
240 GO TO 210
300 LET y=y+1
310 IF POINT (x,y)=1 THEN  GO TO 400
320 GO TO 100
400 LET y=g
405 LET z=x
410 LET y=y-1
420 PLOT  INK i;x,y: LET x=x+1
430 IF POINT (x,y)=1 THEN  GO TO 500
440 IF POINT (x,y)=0 THEN  PLOT  INK i;x,y
450 GO TO 420
500 LET x=z
510 LET x=x-1
520 IF POINT (x,y)=1 THEN  GO TO 600
530 IF POINT (x,y)=0 THEN  PLOT  INK i;
x,y
540 GO TO 510
600 LET x=z
610 LET y=y-1
620 IF POINT (x,y)=1 THEN  GO TO 700
630 GO TO 420
700 PRINT AT 0,0;"

710 PRINT AT 0,0;"FIN"
800 INPUT "Otro dibujo ?";q$
810 IF q$="y" THEN  GO TO 25
820 STOP
```

```
 900 PLOT  OVER 1;x,y: GO TO 30
1997 REM ****POSICIONARSE****
2000 PRINT AT 0,0;"MUEVE 5,6,7 O 8 PARA
POSICIONARSE 's' PARA SALIR.
  'p' para pintar"
2100 PLOT x,y: PAUSE 10: PLOT  OVER 1;x,
y
2200 IF INKEY$="5" THEN  LET x=x-1
2300 IF INKEY$="8" THEN  LET x=x+1
2400 IF INKEY$="6" THEN  LET y=y-1
2500 IF INKEY$="7" THEN  LET y=y+1
2600 IF INKEY$="s" THEN  GO TO 30
2650 IF INKEY$="p" THEN  GO TO 3000
2700 GO TO 2100
3000 PRINT AT 0,0;"
          "

3010 GO TO 40
```

CONSTRUCCIÓN DE CARACTERES GRÁFICOS

Lo que aquí sigue, es un programa que se hace indispensable para todos aquellos que deseen crear sus propios caracteres gráficos (marcianitos, figuras técnicas, alfabeto griego, etc.). En la pantalla aparece un tablero de 8 × 8 y los posibles caracteres definidos por el usuario debajo de la letra a la que corresponden en modo gráfico. Al lado, a la izquierda de cada hilera se encuentra el valor del byte correspondiente. A la derecha hay un resumen de los comandos que usted puede utilizar en este programa. Usted puede mover el cursor (asterisco) a través del tablero usando las flechas (teclas 5 a 8) sin SHIFT. El asterisco se moverá en la dirección de la flecha que hay sobre cada tecla. En cualquier momento puede cambiar el estado del recuadro en que se encuentra el asterisco pulsando la tecla C. Si el recuadro está vacío, se llenará, y si está lleno se volverá blanco. Al mismo tiempo que hace esto, los bytes de cada hilera bajo la columna DATA cambiarán según las modificaciones que haga.

Una vez ha obtenido el caracter que desea, puede conservarlo pulsando la tecla K. Entonces el Spectrum le preguntará en qué tecla lo quiere (de la A a la U). Pulsando

ENTER el carácter reemplazará instantáneamente a la letra elegida en la lista que hay en la parte alta de la pantalla. Aquí es donde usted comprobará en pequeño, si queda bien lo que ha construido. Puede insertar en el tablero cualquiera de los gráficos definidos pulsando la tecla I. Esto necesitará hacerlo tanto si desea modificarlo como imprimirlo.

Para tener una copia del carácter en la impresora simplemente pulse P. Esto coloca una copia del carácter en la impresora así como los bytes que forman cada línea, tanto en decimal como en binario. El listado en binario es útil porque permite ver la apariencia del carácter ocho veces mayor.

Finalmente, es posible también grabar en cassette los caracteres construidos, así como recuperarlos más tarde. Esto se hace saliendo del programa (pulsando la tecla Q) y utilizando el comando SAVE "xxx" CODE. Las direcciones de memoria que usted quiere grabar tienen que ir detrás de la palabra CODE, y usted ya sabe que son USR "a" para el gráfico de la tecla A, USR "b" para el de la B, etc. Luego hay que poner el número de posiciones de memoria (o para ser exactos, el número de bytes) que va a grabar. Son siempre ocho por cada carácter. Por lo tanto, para grabar todo el juego tendrá que escribir:

```
SAVE "caracteres" CODE USR "a", (21*8)
```

Esto es, mientras que el primer número después de CODE nos dice la primera posición que se graba, el número de después de la coma indica el número de bytes que quiere grabar. En este caso, usted sabe que son 21 caracteres de 8 bytes cada uno. Observe que no hay que hacer el cálculo, pues el Spectrum lo hace por usted directamente.

Un carácter dado puede ser grabado con un comando del tipo:

```
SAVE "carácter A" CODE USR "a",8
```

(usted escoge el nombre del programa mientras tenga menos de 10 caracteres)

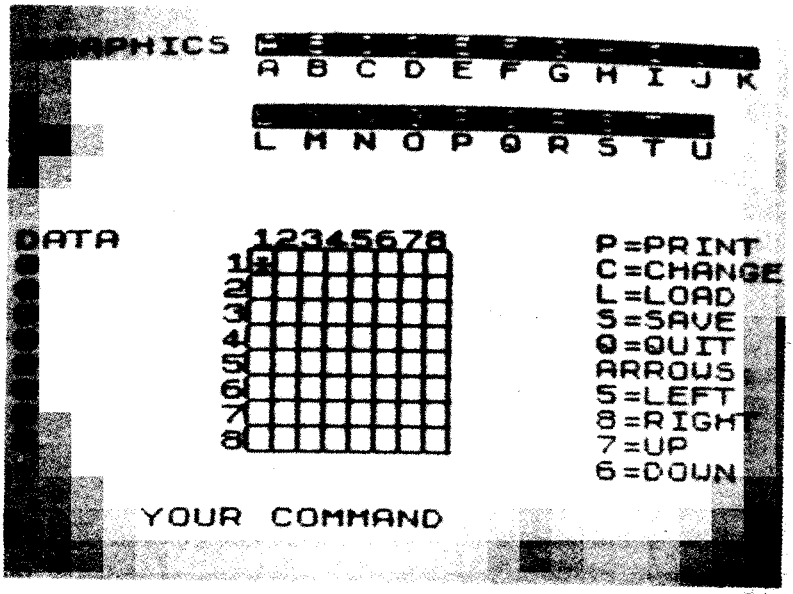
Recuperarlo es igual de fácil. Simplemente escriba:

LOAD "nombre" CODE USR "a"

o para el juego entero:

LOAD "caracteres" CODE USR "a"

El número de bytes después de la dirección en LOAD no es necesario porque en la cinta hay un número fijo de bytes. Es aconsejable que el nombre del programa sea indicativo del contenido de los caracteres.




```
   2 REM GENERADOR DE CARACTERES
  40 BORDER 1: PAPER 5: INK 1: CLS
  50 POKE 23609,100
  60 PRINT AT 9,23;"P=IMPRIME";AT 10,23;
"C=CAMBIA";AT 11,23;"L=CARGA";AT 12,23;"
S=GUARDA"
  70 PRINT AT 13,23;"Q=FUERA";AT 14,23;"
FLECHAS";AT 15,23;"5=IZQ.";AT 16,23;"8=D
ER.";AT 17,23;"7=ARRIBA";AT 18,23;"6=ABA
JO"
  80 DIM G$(8,8)
  90 PRINT AT 1,0;"GRAFICOS"; INVERSE 1;
AT 1,9;"A B C D E F G H I J K"
```

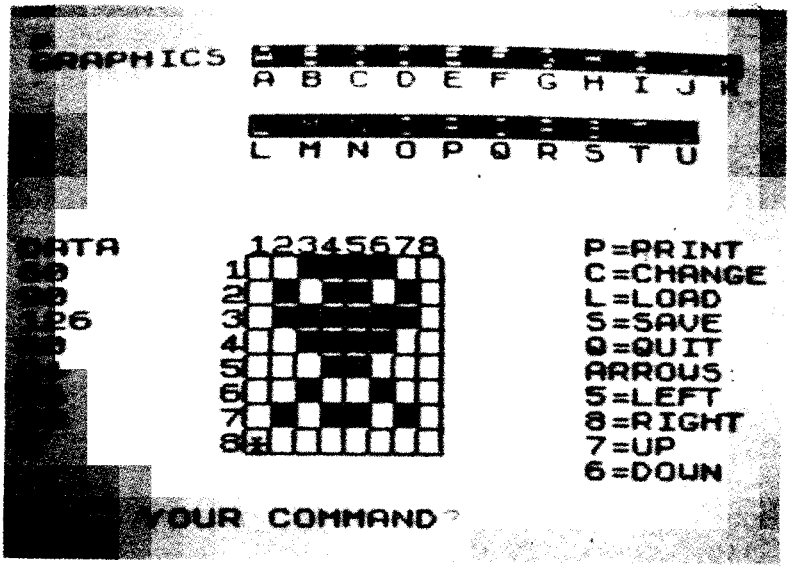
```
  95 REM LA LINEA 100 ES EN MODO
 GRAFICOS
 100 PRINT AT 2,9;"a b c d e f g h i j k
 "
 110 PRINT  INVERSE 1;AT 4,9;"L M N O P
Q R S T U"
 115 REM LA LINEA 120 ES EN MODO
GRAFICOS
 120 PRINT AT 5,9;"l m n o p q r s t U "
 130 GO SUB 540: REM PARRILLA DE CHR$
 140 FOR A=10 TO 17
 150 FOR B=9 TO 17
 160 PRINT  OVER 1;AT B,A-1;"_";AT A,B;"
u"
 170 LET S=A-9: PRINT  OVER 1;AT 9,A-1;S
;AT A,8;S
 180 NEXT B: NEXT A: OVER 0
 190 LET X=10: LET Y=9
 200 PRINT  OVER 1;AT X,Y;"*"
 210 PAUSE 1
 220 PRINT AT 9,0;"DATA"
 230 FOR A=10 TO 17: PRINT AT A,0;VAL ("
BIN "+G$(A-9));"    ": NEXT A
 240 PRINT AT 20,7;"SU COMANDO"
 250 PRINT  OVER 1;AT 20,17;"?": OVER 0
 260 IF INKEY$="6" THEN  GO TO 360
 270 IF INKEY$="7" THEN  GO TO 400
 280 IF INKEY$="c" THEN  GO TO 620
 290 IF INKEY$="5" THEN  GO TO 440
 300 IF INKEY$="s" THEN  GO TO 680
 310 IF INKEY$="p" THEN  GO TO 990
 320 IF INKEY$="8" THEN  GO TO 480
 330 IF INKEY$="1" THEN  GO TO 760
 340 IF INKEY$="q" THEN  STOP
 350 GO TO 210
 360 PRINT  OVER 1;AT X,Y;"*": LET X=X+1
: IF X=18 THEN  LET X=17
 370 PAUSE 10
 380 PRINT  OVER 1;AT X,Y;"*"
 390 GO TO 210
 400 PRINT  OVER 1;AT X,Y;"*": LET X=X-1
: IF X=9 THEN  LET X=10
 410 PAUSE 10
 420 PRINT  OVER 1;AT X,Y;"*"
```

```
 430 GO TO 210
 440 PRINT  OVER 1;AT X,Y;"*": LET Y=Y-1
: IF Y=19 THEN  LET Y=9
 450 PAUSE 10
 460 PRINT  OVER 1;AT X,Y;"*"
 470 GO TO 210
 480 PRINT  OVER 1;AT X,Y;"*": LET Y=Y+1
: IF Y>16 THEN  LET Y=16
 490 PAUSE 10
 500 PRINT  OVER 1;AT X,Y;"*"
 510 GO TO 210
 520 STOP
 530 REM
 540 REM PARRILLA DE CARACTERES
 550 REM
 560 FOR A=0 TO 7: POKE USR "U"+A,128: N
EXT A
 570 FOR A=1 TO 8
 580 LET G$(A)="00000000"
 590 NEXT A
 600 RETURN
 610 REM
 620 REM CAMBIO DEL BIT CHR$
 630 PRINT AT 0,0;"p": REM
 640 LET G$(X-9,Y-8)=("1" AND G$(X-9,Y-8
)="0")+("0" AND G$(X-9,Y-8)="1")
 650 PRINT  INK 0; OVER 1;AT X,Y;" "
 660 GO TO 210
 670 REM
 680 REM GUARDA CHR$
 690 REM
 700 INPUT "QUE CHR$?  "; LINE C$
 710 FOR A=0 TO 7: POKE VAL ("USR "+"C$"
)+A,VAL ("BIN "+G$(A+1)): NEXT A
 715 REM LA LINEA 720 ES EN
 GRAFICOS
 720 PRINT AT 2,9;"a b c d e f g h i j k
"
 725 REM LA LINEA 730 ES EN
 GRAFICOS
 730 PRINT AT 5,9;"l m n o p q r s t u"
 740 GO TO 210
 750 REM
 760 REM CARGAR CHR$
```

```
 770 REM
 780 INPUT "QUE CRH$?  "; LINE C$
 790 FOR W=1 TO 8
 800 LET A=PEEK ((USR C$)+W-1)
 810 LET T=128
 820 FOR S=1 TO 8
 830 LET G$(W,S)="0"""
 840 IF A>=T THEN  GO SUB 880
 850 LET T=T/2
 860 NEXT S
 870 NEXT W: GO TO 910
 880 LET G$(W,S)="1"
 890 LET A=A-T
 900 RETURN
 910 FOR A=1 TO 8
 920 NEXT A
 930 FOR W=1 TO 8
 940 FOR S=1 TO 8
 950 IF ATTR (W+9,S+8)=40 THEN  PRINT  O
VER 1;AT W+9,S+8;" "
 960 IF G$(W,S)="1" THEN  PRINT  INK 0;
OVER 1;AT W+9,S+8;" "
 970 NEXT S: NEXT W: GO TO 210
 980 REM
 990 REM CHR$ PARA PINTAR
1000 REM
1010 LPRINT CHR$ (CODE C$+47)
1020 LPRINT "DATA:"
1030 FOR A=1 TO 8
1040 LPRINT G$(A),VAL ("BIN "+G$(A))
1050 NEXT A
1060 GO TO 210
```

IDEAS PARA MODIFICAR EL PROGRAMA

Hay cantidad de modificaciones que usted puede hacer para conseguir que este programa sea aún más versátil. Lo primero de todo es que contiene muy pocas rutinas que comprueben si las respuestas dadas por el usuario son correctas o no. Por ejemplo, puede añadir una rutina que compruebe que el carácter que responde a la pregunta "Qué carácter?" sea una letra entre la A y la U (simplemente ponga INPUT A$, IF A$ > "U" THEN PRINT A$; "no es un gráfico definible").



También puede construir rutinas que hagan girar el tablero, lo que le será muy útil para juegos de naves espaciales. Por último puede crear un formato de pantalla diferente, en donde haya cuatro tableros de 8 × 8 formando una grande de 16 × 16. Entonces puede tener la opción de crear un carácter en cualquiera de los tableros. De este modo puede fabricar caracteres más grandes y ver cómo quedan sin necesidad de correr el programa que los use.

OVER

El Spectrum dispone de un comando muy útil llamado **OVER,** que ejecuta lo que se llama un OR exclusivo en la impresión. Esto significa que cuando un carácter se imprime en la misma posición que otro, si OVER está puesto (OVER 1), entonces en vez de borrarse el primero, se produce la fusión de los dos. Para entender mejor cómo funciona OVER necesitamos volver de nuevo a la numeración binaria. Los dos bytes de cualquier posición de la pantalla, son comparados bit a bit, de manera que dos unos o dos ceros dan un cero, y si son distintos dan un uno.

		A 0	A 1
B	0	0	1
	1	1	0

Obviamente, OVER es otra de las ampliaciones que le podría hacer al programa de construcción de caracteres. Necesitaría añadir una subrutina que comparara dos caracteres bit a bit y pintara un cuadrado blanco si son iguales, y uno negro si son distintos.

El OR exclusivo de estos dos números binarios da:

```
10011010
01010001
11001011
```

Como puede ver es posible crear un carácter, entonces crear otro, y obtener la simulación de OVER en un tablero mayor.

MÁS DE 21 GRÁFICOS DEFINIBLES

Hasta ahora, sólo me he referido a los 21 caracteres para los que el Spectrum tiene reservada un área de memoria que va desde la dirección USR "A" hasta USR "U" + 8; sin embargo, si usted lo desea puede redefinir la totalidad del juego de caracteres (con excepción de los gráficos que se encuentran en las teclas numéricas).

Para ver esto instantáneamente haga (pero cuidado: todos los caracteres se convertirán en un grupo aleatorio de puntos; asegúrese de que no tiene ningún programa vital en memoria):

POKE 23607,0 (ENTER)

Si tiene algún programa en memoria verá que el listado se ha convertido en un barullo de puntos. Para volver a la situación normal pulse ENTER (para asegurarse de que tiene el cursor K, aunque no pueda reconocerlo) y pulse la tecla 0 (POKE) cuidadosamente seguida de 23607,60. ¡Es mejor que no mire la pantalla mientras lo hace! Ahora pulse ENTER y la situación volverá a su cauce normal, si esto no ocurre, no desespere, simplemente desenchufe y vuelva a enchufar.

Hay una variable llamada CHARS, que se encuentra en

las posiciones de memoria 23606 y 23607. En estas posiciones, se encuentran los bytes que le dicen al Spectrum la dirección (menos 256) de la tabla de caracteres en la ROM. Puede ver cuál es este número escribiendo:

```
PRINT PEEK 23606 + 256* PEEK 23607
```

Obtendrá la respuesta 15360. Lo que le he hecho hacer hace un momento es cambiar el byte de la posición 23607 por un 0 y como el otro también es 0 (compruébelo con PEEK) su Spectrum ha creído que la tabla de caracteres empezaba en la posición 256. Habrá adivinado que cambiando, esta dirección en múltiplos de 8 se pueden obtener efectos sorprendentes. Escriba esto (teniendo el juego normal):

```
POKE 23606,8
```

Es interesante, ¿verdad?

Todo esto se puede usar para hacerle más difícil a la otra gente, el leer los listados de su programa. Sin embargo, su programa, también usará los caracteres, entonces tendrá que definir un nuevo juego, como máximo de 21 que no serán afectados al hacer POKEs en las variables del sistema.

Una aplicación más seria consiste en reemplazar el juego de caracteres del Spectrum, por el suyo propio. La manera más fácil de hacerlo es reservar un área al final de la memoria usando CLEAR. Como el mismo nombre sugiere (CLEAR en inglés significa "limpiar") este comando borra toda la memoria a partir de la dirección que se indica tras él. Así por ejemplo, CLEAR 30000 reserva espacio desde la posición 30000 en adelante. Dependiendo del tamaño de la memoria de su Spectrum (16K o 48K) usted necesitará hacer CLEAR seguido de 32767 — x o bien de 65535— x donde x es el número de caracteres que va a definir multiplicado por 8. Después de esto, necesita colocar los datos (bytes) de sus caracteres en esta área de memoria, usando POKE. Si usted usa un programa anterior (el constructor de caracteres), para obtener los bytes de los nuevos, simplemente requerirá una rutina como ésta para hacer todo el trabajo:

```
10 FOR a = 32000 TO 32000 + 8*95    (esto es para el
20 INPUT n                          juego entero y con
30 POKE a,n                         Spectrum de 16K)
40 NEXT a
```

Por supuesto que como alternativa para cometer menos errores, puede colocar todos los bytes (que aquí se cargan a la variable n) en una línea DATA y utilizar READ en la línea 20.

SONIDO EN SU SPECTRUM

Vimos en la primera parte que en el Spectrum, el sonido se produce con el comando BEEP seguido de dos números, el primero de los cuales indica la longitud y el segundo el tono. Por ejemplo:

```
BEEP 0.5, 10
```

Todo esto no parece muy excitante, sin embargo, es posible crear sonidos mucho más interesantes y útiles, con ayuda del BASIC, y si usted quiere intentar algo de código máquina (véase la última parte), podrá obtener sonidos asombrosos.

Con ayuda del BASIC vemos todas las notas que se pueden obtener:

```
1 FOR N = -60 TO 69
20 BEEP 0.2 , N
30 NEXT N
```

O de un modo más aleatorio:

```
10 REM MUSICA ALEATORIA
20 LET nota = (RND* 100) — 50
30 LET nota = RND
40 BEEP dur, nota
50 GOTO 20
```

Observará que el sonido del Spectrum es bastante bajo, pero usted puede amplificarlo usando el cassette o un amplificador. La señal sale tanto de MIC como de EAR, quizás un poco más fuerte por este último, pruebe usted

mismo cuál es la salida que le va mejor para su amplificador. Cuando el Spectrum no envía sonido al amplificador, éste emite un zumbido bastante molesto. El único modo de evitarlo es mediante filtros o bien encontrar un punto crítico de volumen en el que se oiga suficiente el sonido pero no el zumbido.

Otro problema al amplificar el sonido es que el "clic" del teclado también queda amplificado. Pero el volumen del "clic" se puede alterar gracias a otra variable del sistema. Escriba:

POKE 23609, 100

Esto producirá un sonido más alto. Se puede "pokear" cualquier número entre 0 y 255. Pero si este número es muy grande, el "clic" se enlentece mucho. Yo encuentro que los valores alrededor de 100 son los más ideales.

Obviamente, mientras los enfuches MIC y EAR están conectados al amplificador, no pueden usarse para LOAD y SAVE. Un interruptor-conmutador sería una buena solución, si no quiere estar enchufando y desenchufando.

Volvamos a la música. Detrás de esta numeración de notas, hay por supuesto una regla. Así la nota 0 corresponde al DO central, el 1 es el DO sostenido, el 4 el MI, etc. Si sabe algo de música verá que hay doce números en cada octava, por lo tanto el 12 es el DO una octava mayor que el central. Usando esto junto con lo que ya sabemos de manipulación de cadenas, podemos construir un programa que permita introducir las notas directamente en lugar de estar escribiendo largos comandos BEEP. Aquí está, se llama COMPOSER:

```
  10 REM COMPOSITOR
  20 DIM X(50): DIM Y(50)
  30 LET K=0: LET L=1
  40 BORDER 0: PAPER 6: INK 9: CLS
  50 PRINT AT 0,10; INVERSE 1;"EL COMPOS
ITOR"''
  60 PRINT "ESCRIBE LA MUSICA EN FORMA D
E LINEA CON NUMEROS Y LETRAS"
  70 PRINT ''"PON LAS NOTAS EN LETRAS SE
GUIDAS POR LA DURACION EN TIEMPOS"
  80 PRINT ''"TIENES DOS OCTAVAS, LA BAJ
```

```
A DESDE LA A A LA G Y LA ALTA DESDE LA a
 A LA g"
  90 INPUT N$
 100 FOR A=1 TO LEN N$ STEP 2
 110 IF CODE N$(A)<97 THEN  GO TO 280
 120 IF N$(A)="a" THEN  LET K=-0.5
 130 IF N$(A)="d" THEN  LET K=0.5
 140 IF N$(A)="e" THEN  LET K=1
 150 IF N$(A)="f" THEN  LET K=1
 160 IF N$(A)="g" THEN  LET K=1.5
 170 LET Y(L)=(CODE N$(A)-87)+(2*K)
 180 LET L=L+1
 190 LET K=0
 200 NEXT A
 210 LET L=1
 220 FOR T=2 TO LEN N$ STEP 2
 230 LET X(L)=VAL N$(T)/2
 240 LET L=L+1
 250 NEXT T
 260 GO TO 360
 270 STOP
 280 IF N$(A)="A" THEN  LET K=-0.5
 290 IF N$(A)="B" THEN  LET K=-0.5
 300 IF N$(A)="F" THEN  LET K=0.5
 310 IF N$(A)="G" THEN  LET K=0.5
 320 LET Y(L)=(CODE N$(A)-67-K)*2
 330 LET K=0
 340 LET L=L+1
 350 GO TO 200
 360 FOR Z=1 TO LEN N$/2
 370 BEEP X(Z)/2,Y(Z)
 380 NEXT Z
 390 STOP
```

Como podrá observar al ejecutar este programa, le permite escribir las letras de la A a la G para una octava baja en la que la C representa al Do central, las letras de la 'a' a la 'g' son una octava más alta. He escogido el número 1 para representar un tiempo, así una melodía podría ser "A1 B1 C2 A1 C2". Un tiempo tiene una duración de

---

N.T. Las notas se escriben en nomenclatura anglosajona, es decir son C, D, E, F, G, A, B en lugar de Do, Re, Mi, Fa, Sol, La, Si; lo que por otro lado facilita mucho la manipulación de las cadenas.

0.25; si usted quiere ir más rápido o más lento, entonces esta constante puede ser establecida por usted. Además, ¿por qué no añadir una línea que pregunte la velocidad de ejecución?

```
5 INPUT "Qué compás? 1 lento, 5 rápido"; d
```

Hacer música con este programa es mucho más fácil que usar los interminables BEEPs y también más fácil que escribir primero la partitura en un papel. Aquí, la melodía, se guarda en una cadena, así que si usted quiere guardar su melodía en el cassette puede grabar todo el programa entero, pero luego al hacer LOAD ejecútelo con GOTO 1 en lugar de RUN, pues este último comando borra las variables.

¿Por qué no convertir el Spectrum en un órgano? Este programa utiliza el comando INKEY$ para conseguirlo. INKEY$ da como resultado una cadena conteniendo la tecla que se está pulsando en aquel momento y según su resultado, se ejecuta una nota u otra. He puesto el Do central en la tecla T y con tres octavas a elegir. Las teclas numéricas son los sostenidos y bemoles, las teclas de la Q a la P corresponden a las teclas blancas del órgano. También puede accionar un vibrator.

```
   2 REM ****ORGANO****
   3 BORDER 0: PAPER 2: INK 7: CLS
   4 PRINT AT 8,0;"PULSE 'Z' PARA OCTAVA
 ALTA,            'X' PARA LA BAJA"
   5 PRINT AT 16,0; INVERSE 1;"PULSE 'V'
 PARA VIBRATOR,                'B' PARA SA
LIR DE VIBRATOR"
   6 PRINT AT 19,0; INVERSE 1;"PULSE 'C'
 PARA LA OCTAVA CENTRAL    LA TECLA 'T'
ES EL DO        "
   7 LET k=0: LET x=0.3
   8 PAUSE 500
   9 REM TECLADO VISUALIZADO
  10 CLS
  11 PRINT  INVERSE 1;AT 10,4;"Q";AT 10,
6;"W";AT 10,8;"E";AT 10,10;"R";AT 10,12;
"T";AT 10,14;"Y";AT 10,16;"U";AT 10,18;"
I";AT 10,20;"O";AT 10,22;"P"
```

```
  15 PRINT  PAPER 0;AT 8,5;"2";AT 8,7;"3
";AT 8,9;"4";AT 8,13;"6";AT 8,15;"7";AT
8,19;"9";AT 8,21;"0"
  18 REM debe tocarse con
minusculas
  20 IF INKEY$="z" THEN  LET K=12
  21 IF INKEY$="t" THEN  BEEP x,0+k
  22 IF INKEY$="6" THEN  BEEP x,1+k
  23 IF INKEY$="y" THEN  BEEP x,2+k
  24 IF INKEY$="7" THEN  BEEP x,3+k
  25 IF INKEY$="u" THEN  BEEP x,4+k
  26 IF INKEY$="i" THEN  BEEP x,5+k
  27 IF INKEY$="9" THEN  BEEP x,6+k
  28 IF INKEY$="o" THEN  BEEP x,7+k
  29 IF INKEY$="0" THEN  BEEP x,8+k
  30 IF INKEY$="p" THEN  BEEP x,9+k
  31 IF INKEY$="r" THEN  BEEP x,-1+k
  32 IF INKEY$="4" THEN  BEEP x,-2+k
  33 IF INKEY$="e" THEN  BEEP x,-3+k
  34 IF INKEY$="3" THEN  BEEP x,-4+k
  35 IF INKEY$="w" THEN  BEEP x,-5+k
  36 IF INKEY$="2" THEN  BEEP x,-6+k
  37 IF INKEY$="q" THEN  BEEP x,-7+k
  38 IF INKEY$="x" THEN  LET k=-12
  39 IF INKEY$="c" THEN  LET k=0
  40 IF INKEY$="v" THEN  LET x=0.03
  41 IF INKEY$="b" THEN  LET x=0.3
  42 GO TO 20
```

Usted probablemente pensará en muchos refinamientos para este programa. ¿Por qué no tener al menos dos teclados, como los órganos grandes? Puede tener las hileras de arriba para las octavas altas y las de abajo para las bajas. ¿Por qué no añadir también un tempo? Puede añadir una rutina que con sólo pulsar una tecla produzca un "BUM CHA CHA" en lugar de un solo sonido. El Spectrum también permite variaciones fraccionarias de tono. Este hecho puede ser usado para afinar su órgano con otro instrumento. Quizá pueda añadir una rutina de afinación como ésta:

```
9990 IF INKEY$ = "N" THEN LET d = d + 0.05
9991 IF INKEY$ = "M" THEN LET d = d - 0.05
```

(Donde d representa el tono de las notas.)

Por supuesto que también puede apartarse de la práctica usual y crear un instrumento con octavas de 16 notas o incluso más, tal como ocurre en la música oriental.

Por último, le sugiero que para tocar este órgano, conecte su Spectrum a un amplificador. El sonido es mucho mejor cuando se amplifica. Por supuesto que con un poco de imaginación puede convertir este programa para que él mismo componga sus melodías. También puede usar los caracteres definibles por el usuario para crear las partituras de las notas y conseguir que la nota correcta, con la longitud correcta, aparezca en la pantalla sobre un pentagrama mientras usted toca. Quizás es un proyecto muy ambicioso, pero aquí tiene a PARTITURA un programa que es efectivamente una extensión del anterior:

```
 10 REM PARTITURA
 20 PAPER 5: INK 0: BORDER 1: CLS
 30 LET A=151
 40 REM DIBUJA PENTAGRAMA
 50 FOR B=1 TO 5
 60 PLOT 0,A: DRAW 255,0
 70 LET A=A-8
 80 NEXT B
 90 FOR A=0 TO 7
100 READ T
105 REM LO DE ENTRE COMILLAS ES
'A' EN GRAFICOS
110 POKE USR "a"+A,T
120 NEXT A
130 FOR B=0 TO 7
140 READ T
145 REM LO DE ENTRE COMILLAS ES
'B' EN GRAFICOS
150 POKE USR "b"+B,T
160 NEXT B
170 PRINT  OVER 1;AT 3,3;"a";AT 4,3;"b"
:
175 REM A I B SON GRAFICOS
180 BEEP 0.25,0
190 REM PARTE SUPERIOR DE LA
NOTA
200 DATA 0,4,7,5,4,4,4,4
```

```
210 REM PARTE INFERIOR DE LA
NOTA
220 DATA 0,60,124,124,124,56,0,0
230 STOP
```

Usando el comando BEEP es difícil crear en BASIC sonidos raros para juegos, dado que sólo se permite variar la duración y el tono. Sin embargo, algunas rutinas muy útiles permiten crear sonidos sin estas limitaciones. Usted puede crear sonidos del tipo de un paso. Por ejemplo, aquí tenemos a un hombre subiendo una escalera:

```
  2 REM SUBIENDO UNA ESCALERA
  3 REM creando un hombre CHR$
  4 BORDER 4: PAPER 6: CLS
 10 POKE USR "p",  BIN 00011000
 20 POKE USR "p"+1,BIN 00100100
 30 POKE USR "p"+2,BIN 10011001
 40 POKE USR "p"+3,BIN 01111110
 50 POKE USR "p"+4,BIN 00011000
 60 POKE USR "p"+5,BIN 01100100
 70 POKE USR "p"+6,BIN 10000100
 80 POKE USR "p"+7,BIN 00000100
 90 REM Segundo hombre para movimiento
100 POKE USR "u",  BIN 00011000
110 POKE USR "u"+1,BIN 00100100
120 POKE USR "u"+2,BIN 10011001
130 POKE USR "u"+3,BIN 01111110
140 POKE USR "u"+4,BIN 00011000
150 POKE USR "u"+5,BIN 00100110
160 POKE USR "u"+6,BIN 00100001
170 POKE USR "u"+7,BIN 00100000
200 REM Dibuja la escalera
210 LET x=BIN 01000010
220 POKE USR "a",x
230 POKE USR "a"+1,x
240 POKE USR "a"+2,BIN 01111110
250 POKE USR "a"+3,x
260 POKE USR "a"+4,x
270 POKE USR "a"+5,x
280 POKE USR "a"+6,BIN 01111110
290 POKE USR "a"+7,x
300 FOR t=0 TO 21
```

```
 310 PRINT AT t,10;"a": REM Este es el g
rafico CHR$ en la tecla 'A'
 320 NEXT t
 330 LET x=0: LET y=21: LET z=-1
 340 FOR n=y TO x STEP z
 350 PRINT AT n,10;"p": PAUSE 2-z: PRINT
 AT n,10;"u"
 355 PAUSE 2-z
 360 REM Recuerda que son graficos CHR$!
 370 PRINT AT n,10;"a"
 380 IF ABS z=z THEN  GO TO 500
 390 BEEP 0.02,30: BEEP 0.02,40
 400 NEXT n
 410 LET k=x: LET x=y: LET y=k: LET z=-z
 420 GO TO 340
 500 BEEP 0.01,(40-(n/2))
 510 NEXT n: GO TO 330
```

También hay sonidos muy variados para programas de invasores:

```
10 REM sonidos especiales
20 FOR x=-10 TO 0
30 BEEP 0.0125,x
40 NEXT x
50 FOR y=0 TO -5 STEP -1
60 BEEP 0.0125,y
70 NEXT y

10 REM sonidos especiales
20 LET d=0.0125
30 FOR x=1 TO 2
40 FOR y=4 TO 16 STEP 2
50 BEEP d,y
60 NEXT y
70 NEXT x
```

También es posible tener sonidos de nave espacial:

```
 2 REM  nave espacial
 5 LET n=0
10 FOR a=0 TO 3
20 BEEP .01,n
```

```
30 LET n=n+3
40 IF n>60 THEN  GO TO 5
50 GO TO 10
```

Combinar gráficos y sonido, puede ser divertido:

```
 10 REM  collage musical
 20 INK 7: BORDER 1: PAPER 1: CLS
 30 FOR x=0 TO 31
 40 LET n=RND*20
 50 LET d=RND*0.4
 60 LET i=(RND*6)+1
 70 PRINT  INK i;AT 20-n,x;"■"
 80 BEEP d,n
 90 NEXT x
100 CLS : GO TO 30
```

En ordenadores más sofisticados, usted puede tener otros tipos de control, además de estos, sobre el sonido producido. No sólo puede definir la duración y el tono, sino también cómo varían estos factores en el tiempo. A esta variación se le llama envolvente. Podemos simular algo de esto con el Spectrum, teniendo en cuenta que tanto la duración como el tono pueden tomar valores fraccionarios:

```
 10 REM frecuencia/duracion
 20 INK 7: BORDER 1: PAPER 1: CLS
 30 LET d=0.05
 40 FOR f=0 TO 2 STEP 0.5
 50 BEEP d,f
 60 DRAW 300*d,20*f
 70 LET d=d-0.005
 80 NEXT f
 90 DRAW 50,0: BEEP 0.5,2
100 GO TO 10
```

Para añadir aún algo más a estas rutinas, puede hacer que los cambios que se hacen en la duración y el tono varíen asimismo según el tiempo. Así al principio, la frecuencia empieza a crecer según una razón dada, y enseguida crece el tono a una razón más rápida. De todas maneras, las restricciones del comando BEEP no permiten crear sonidos

aceptables en este sentido. También es posible crear zumbidos de "clics" pero que deben de tener muy poco uso.

Ahora bien, si usted va a intentar hacer algo de código máquina podrá crear sonidos que impresionan mucho más, pero de esto ya hablaremos más tarde.