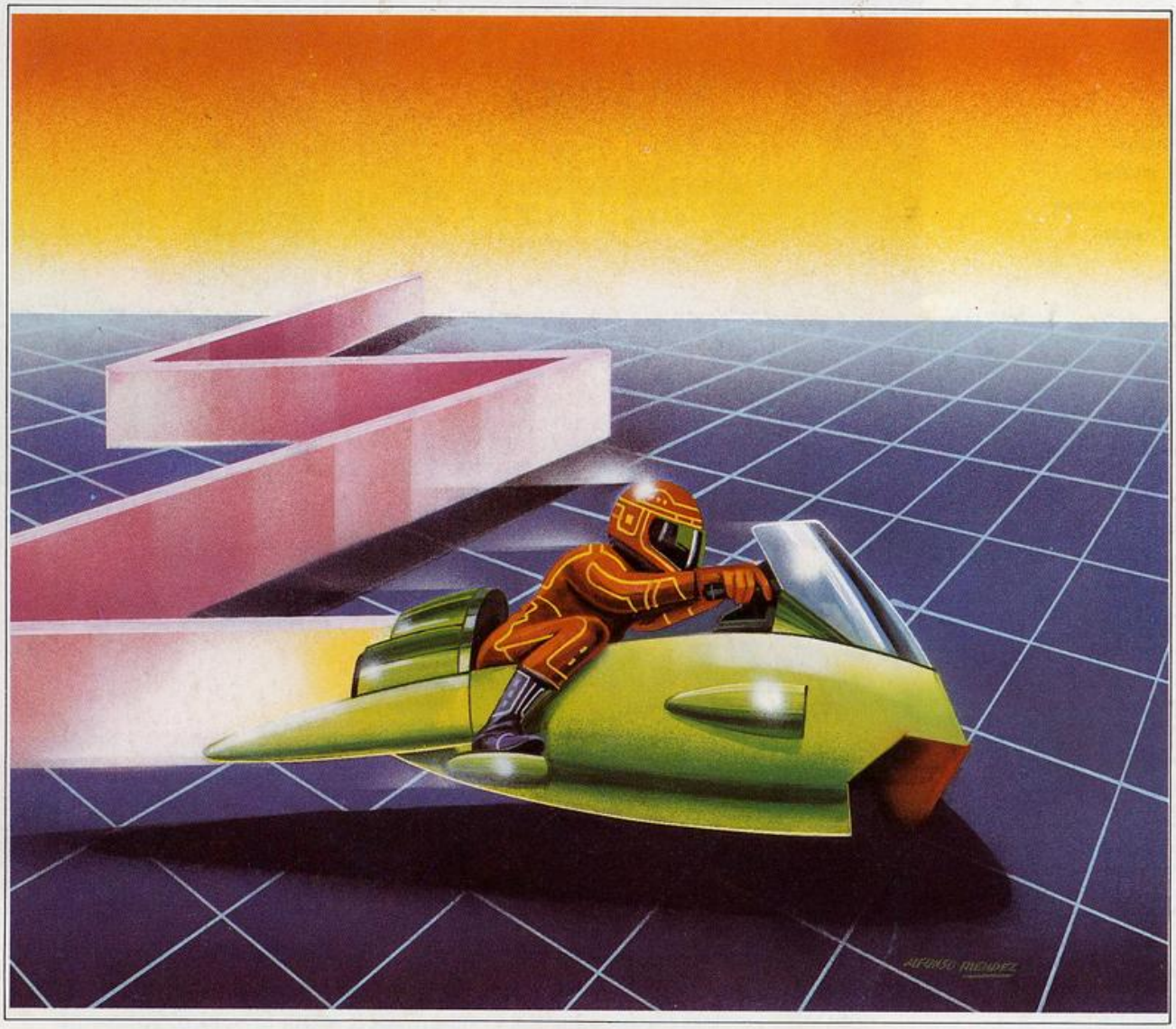
26
150 pts.

RUN

Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

PUNTO A PUNTO

Ya hemos dicho, al hablar de la configuración general de la pantalla del Spectrum, que ésta consta de 24 filas de 32 columnas. A su vez, la pantalla está dividida en dos zonas que podemos denominar como de «impresión» y de «Sistema». La zona reservada a uso del Sistema está compuesta por las dos líneas inferiores de la pantalla, la 22 y la 23, y se emplea para las entradas de datos por medio de **INPUT**, mostrar los mensajes de aviso y error, etc...

La zona de impresión es la accesible directamente por el usuario, y permite la escritura por medio de **PRINT** con sus diferentes calificativos, **TAB**, **AT**, apóstrofe ('), coma (,) y punto y coma (;). Cada una de las posiciones de impresión de las 22 líneas accesibles tiene una configuración de ocho filas de ocho puntos, en total 64, que componen la configuración de cada carácter, de forma similar a como vimos en la definición de caracteres por el usuario.

Cada uno de esos puntos elementales de que se compone la imagen en pantalla, recibe el nombre de *pixel*, abreviatura de las palabras inglesas *picture element*, cuyo significado es «elemento de imagen».

Sabido esto, estamos en condiciones de pensar en la pantalla como una malla de 256 × 176 (45056 puntos en total). El Spectrum cuenta con un modo de alta resolución, que permite encender o apagar cada uno de esos puntos con sólo suministrar sus coordenadas.

Para ello, consideramos como origen de coordenadas el ángulo inferior izquierdo de la pantalla, es decir, el punto situado más abajo y a la izquierda de la columna 0 de la línea 21. Las coordenadas de este punto son, pues (0,0).

Disponemos así de un plano cartesiano con ejes horizontal y vertical. Por seguir la notación convencional, nos referiremos al eje horizontal como EJE X y al vertical como EJE Y.

La acción de encender y apagar puntos concretos de la pantalla, se lleva a cabo a través de la sentencia **PLOT**. Además, el Spectrum dispone de la

En el Spectrum disponemos de unas sentencias de gran precisión que nos permiten manejar la pantalla punto a punto.

La zona reservada al Sistema está compuesta por las dos líneas inferiores de la pantalla (líneas 22 y 23).

¡!

La pantalla está dividida en 24 filas de 32 columnas, numeradas a partir de cero, de las cuales las dos últimas filas (22 y 23) están reservadas al Sistema.

*

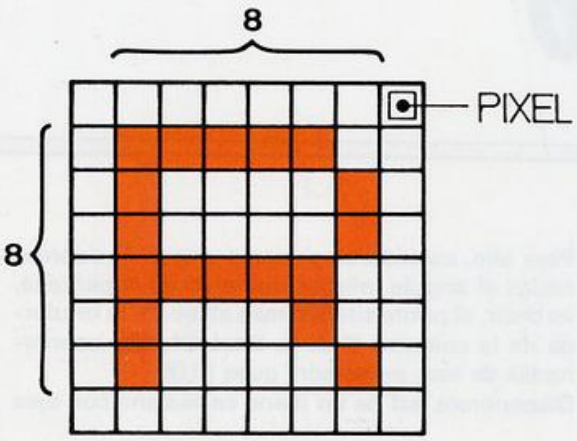
Cada uno de los puntos elementales que componen la pantalla se denomina *pixel*; abreviatura de las palabras inglesas *picture element*, cuyo significado es «elemento de imagen».

¡!

El puntero de alta resolución se encuentra físicamente localizado en la Variable del Sistema COORDS, situada en las direcciones decimales 23677 y 23678; la primera de ellas contiene la coordenada X, y la segunda la coordenada Y del último punto afectado por una instrucción de alta resolución.

*

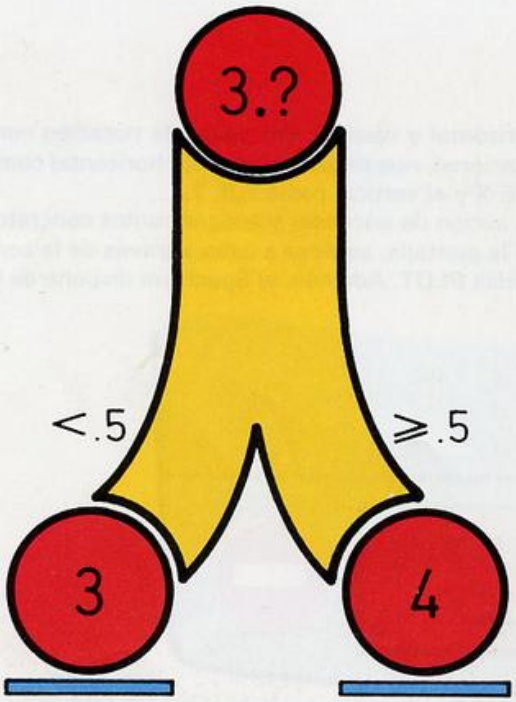
Se dice que una instrucción de alta resolución utiliza parámetros relativos, cuando éstos toman como origen el último punto afectado por una instrucción de este género, es decir, el puntero de alta resolución. Las sentencias de alta resolución del tipo relativo son **DRAW**, en su formato de línea, y en el de arco circular. Por tanto, sus parámetros pueden adoptar tanto valores positivos como negativos.



Cada una de las posiciones de impresión de la pantalla está constituida por 64 pixels.

sentencia **DRAW**, que permite dibujar líneas, y de la sentencia **CIRCLE** que dibuja circunferencias.
El conjunto de estas tres sentencias es suficientemente potente como para emprender la confección de diseños gráficos muy variados, sobre todo si pensamos en que puede hacerse uso de funciones ya conocidas como **INVERSE** y **OVER**, como complemento de las nuevas de manejo de la alta resolución.

La sentencia **PLOT** *redondea sus parámetros al entero más próximo, sumando 0.5, y calculando su parte entera.*

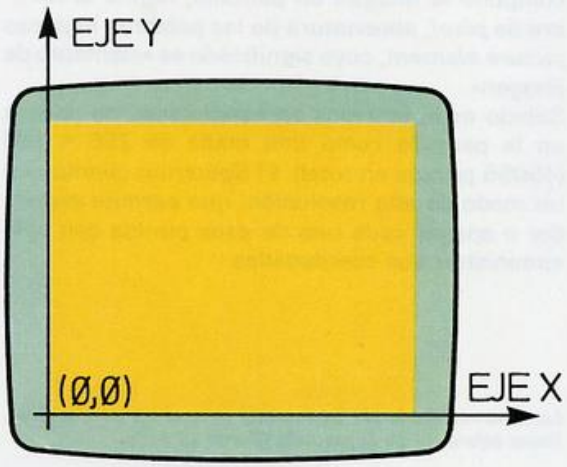


LA SENTENCIA PLOT

Por medio de **PLOT** puede encenderse un *pixel* cualquiera de la pantalla. El formato general de la sentencia es:

PLOT X,Y

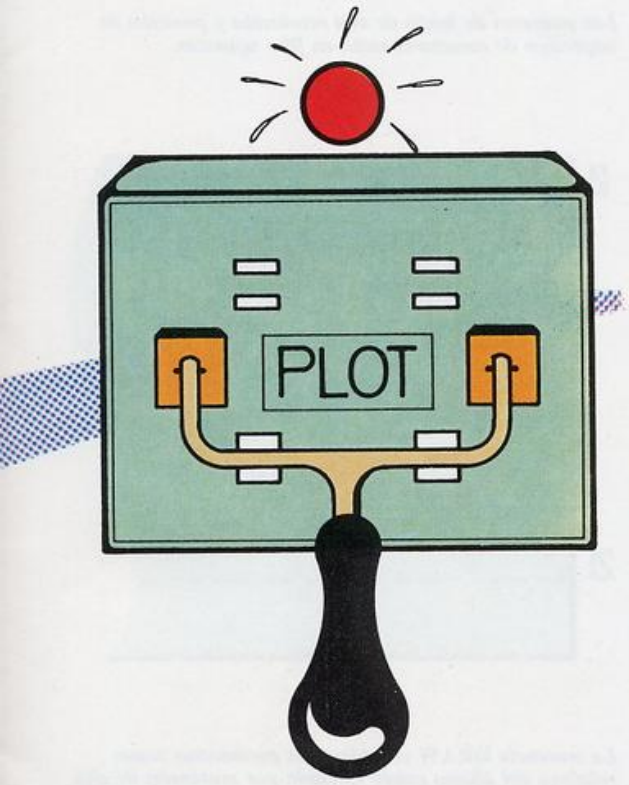
El origen de coordenadas para la alta resolución se sitúa en el punto inferior izquierdo de la pantalla útil (descontadas las líneas del Sistema).



Donde **X** representa el desplazamiento en el eje horizontal, e **Y** en el vertical. Dado que el origen está situado en el extremo inferior izquierdo de la pantalla, es evidente que no pueden admitirse valores negativos. Asimismo, los máximos valores admisibles son 255 para la abscisa (EJE X) y 175 para la ordenada (EJE Y).
Uno de los cometidos más importantes de **PLOT** es la representación de funciones matemáticas en la pantalla de nuestro ordenador. El siguiente programa nos muestra la representación de las funciones **SIN** (seno) **COS** (coseno):

PLOT L, L

El primer parámetro de **PLOT** *indica la coordenada X del punto a activar, y el segundo la coordenada Y.*



La sentencia **PLOT** *nos permite encender independientemente cualquier punto de la pantalla útil.*

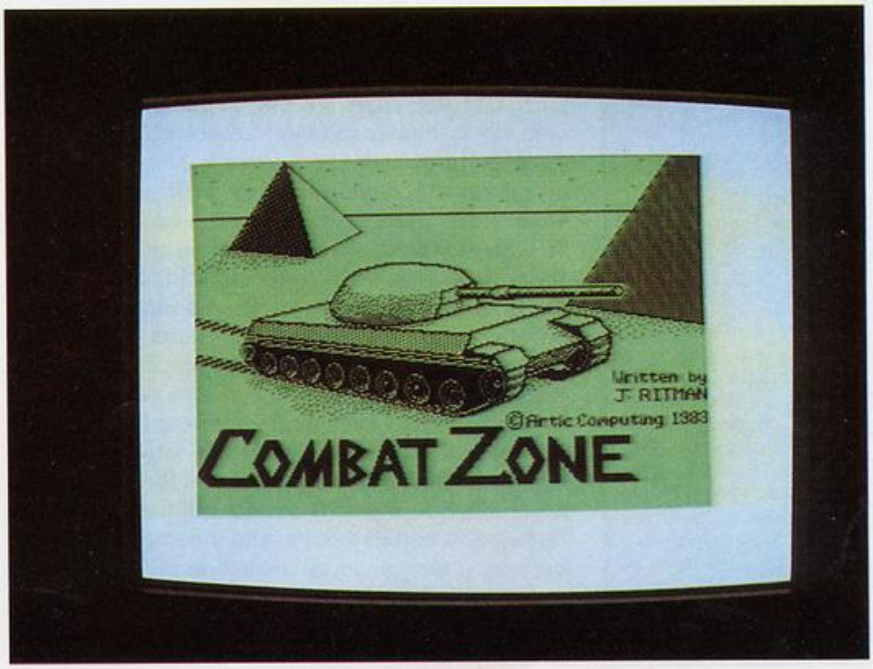
La sentencia **PLOT** admite, sin embargo, valores reales. En este caso se produce un redondeo automático al entero más próximo, labor que realiza el ordenador sumando .5 al valor y calculando su parte entera, de forma análoga a la descrita en la línea 40 del programa.

En cualquier caso, este hecho nos debe resultar familiar, puesto que se produce también cuando empleamos en la impresión convencional las funciones **TAB** y **AT** con números reales.

Basándonos en este hecho, podemos representar funciones en las cuales la diferencia entre los valores de la abscisa y la ordenada se hacen muy grandes, definiendo los incrementos del intervalo menores que la unidad, en nuestro ejemplo .5, para obtener una representación con una mayor densidad de puntos. A continuación veremos la representación de la función Y=X 2:

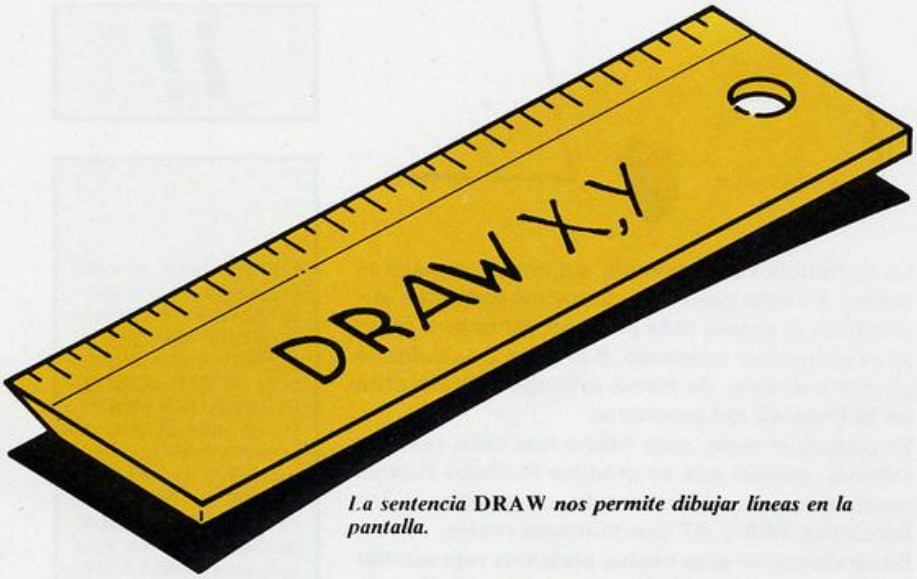
```
 10 REM - GRAFICA Y=X 2 - J.M. LOPEZ
MARTINEZ
 20 FOR X=-13 TO 13 STEP .5
 30 LET Y=X*X
 40 PLOT X+128,Y
 50 NEXT X
```

```
  10 REM - GRAFICAS SENO Y COSENO
J. M. LOPEZ MARTINEZ
  20 FOR X=0 TO 255
  30 LET Y=88+88*SIN (X/128*PI)
  40 PRINT AT 21,0;"X=";X;" Y=";INT
(Y+.5);"  "
  50 GOSUB 100
  60 LET Y=88+88*COS (X/128*PI)
  70 GOSUB 100
  80 NEXT X
  90 STOP
 100 PLOT X,Y: RETURN
```



¡!

Cuando cada uno de los *pixels* de una pantalla puede ser activado independientemente, se dice que la pantalla es DIRECCIONABLE PUNTO A PUNTO. Por lo general, cuando la pantalla de un ordenador supera los 40.000 *pixels* direccionables independientemente se dice que posee ALTA RESOLUCION.



La sentencia **DRAW** *nos permite dibujar líneas en la pantalla.*

LA SENTENCIA DRAW

¡!

La sentencia **CIRCLE** nos permite el trazado de círculos en base a tres parámetros: coordenada central (eje X y eje Y) y radio.

*

La función **POINT** tiene por parámetros la coordenada horizontal y vertical de un punto, y devuelve su estado (conectado=color de **INK** o desconectado=color de **PAPER**), siguiendo el convenio de 1 para conectado y 0 para desconectado.

La sentencia **DRAW** permite dibujar líneas en la pantalla. Su formato general es: **DRAW X,Y**. Donde **X** especifica el valor de la abscisa e **Y** el de la ordenada. Sin embargo, al contrario de lo que sucede con **PLOT**, donde los argumentos son absolutos, **DRAW** tiene argumentos relativos, positivos o negativos, en base a la última posición accedida de la pantalla de alta resolución. Cuando se conecta el ordenador o se realiza un **CLS**, **CLEAR**, **RUN** o **NEW**; el puntero de impresión de la alta resolución se coloca al valor de (0,0), de forma similar a como lo hace el puntero de impresión convencional en (0,0).

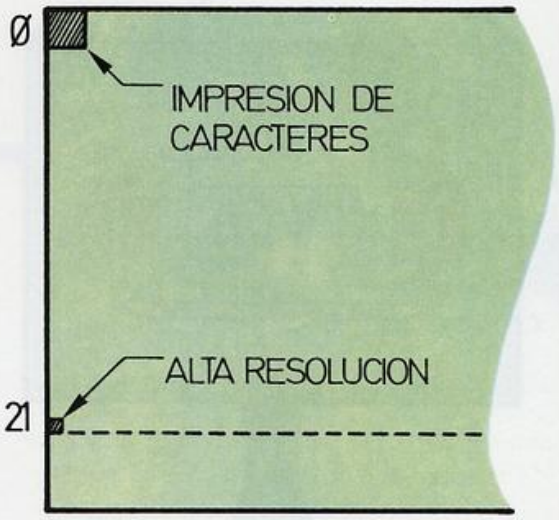
Obviamente, la diferencia estriba en que el puntero de impresión de caracteres se sitúa en el extremo izquierdo de la línea superior de la pantalla, mientras que el de la alta resolución lo hace en el extremo izquierdo de la última línea directamente accesible (última de la fila 21).

A partir de esta situación inicial, y cada vez que se ejecuta una instrucción que contiene **PLOT**, **DRAW** o **CIRCLE**, el puntero se sitúa automáticamente en el último punto accedido, de forma similar a lo que sucede cuando imprimimos caracteres haciendo uso del punto y coma (;), aunque en este último caso, el posicionamiento se hace concretamente al siguiente carácter al impreso.

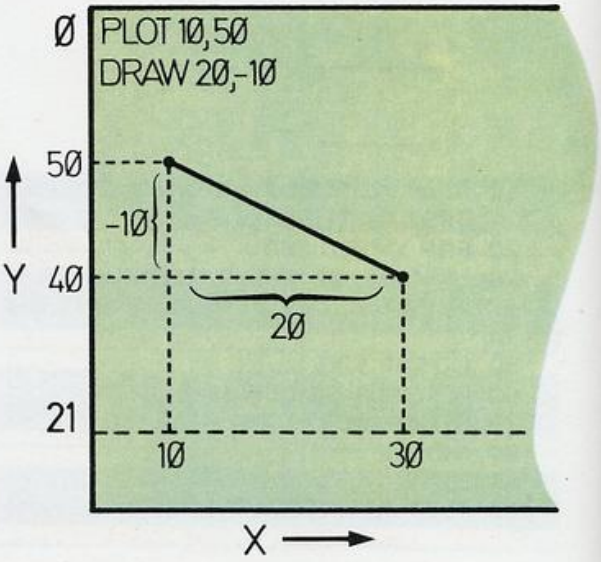
En este estado de cosas, la sentencia **DRAW** traza una línea, de forma que el primero de sus extremos es el puntero actual de la alta resolución, y el segundo, uno situado tantas posiciones a la derecha o izquierda como indique el argumento **X** y tantas hacia arriba o abajo como indique el argumento **Y**.

Esto es posible porque los argumentos de la función **DRAW** pueden ser positivos o negativos. Cuando el argumento **X** es positivo, la abscisa del

Los punteros de inicio de alta resolución y posición de impresión de caracteres están en filas opuestas.



La sentencia **DRAW** *considera sus parámetros como relativos del último punto afectado por sentencias de alta resolución.*



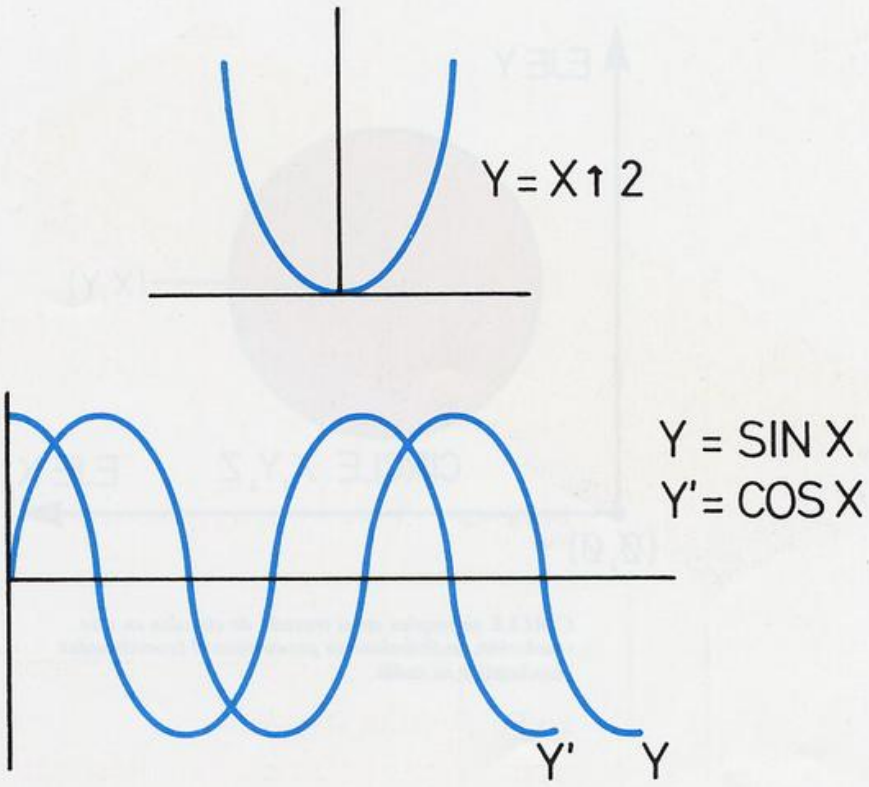
punto de destino se sitúa tantas posiciones a la derecha como indique el argumento, y si es negativo hacia la izquierda.
De forma similar, cuando el argumento **Y** es positivo la ordenada del punto de destino se sitúa tantas posiciones hacia arriba como indique el argumento, y si es negativo las mismas hacia abajo.
En definitiva, la sentencia **DRAW** traza la línea que une dos puntos. El primero de ellos el puntero de la alta resolución, y el segundo el situado en la posición suma algebraica (suma o diferencia según el signo) de los argumentos en el eje X e Y del puntero y los argumentos de abscisa y ordenada de la función **DRAW**.
Para fijar ideas, a continuación mostramos como trazar una línea a lo largo del perímetro de la pantalla:

```
DRAW 255,0:DRAW 0,175:DRAW
-255,0:DRAW 0,-175
```

Basándonos en esta idea, podemos codificar una subrutina que permita dibujar rectángulos en la pantalla *(boxes)*, donde **A** y **B** serán las coordenadas de comienzo del dibujo y **X** e **Y** las longitudes de los lados del rectángulo:

```
PLOT A,B: DRAW X,0: DRAW 0,Y: DRAW
-X,0: DRAW 0,-Y: RETURN
```

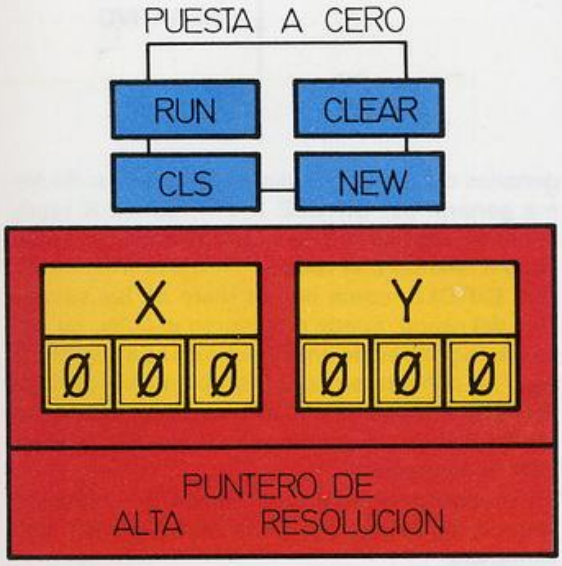
Existe también la posibilidad de trazar arcos de circunferencia por medio de la sentencia **DRAW**, para ello basta con especificar un parámetro adicional de la forma: **DRAW X,Y,Z**.

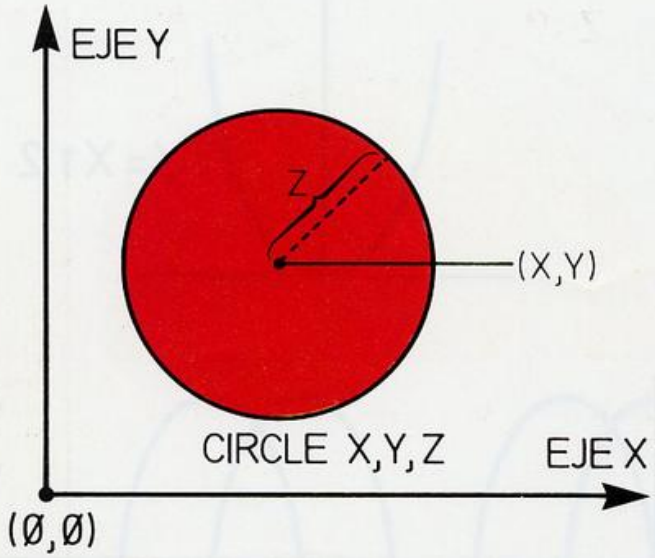


La sentencia **PLOT** *tiene una gran utilidad en la representación de funciones matemáticas.*

Donde **X** e **Y** son las coordenadas del punto de destino, y **Z** indica el número de radianes que debe girarse a medida que se avanza. Con un criterio similar al de la interpretación de **X** e **Y**, el parámetro **Z** implica un giro a la izquierda cuando es positivo y a la derecha en caso contrario.

CLS, CLEAR, RUN *y* **NEW** *ponen a cero el puntero de la alta resolución.*





CIRCLE *se emplea en el trazado de círculos en alta resolución, indicándose su punto central (coordenadas absolutas) y su radio.*

A continuación mostramos como trazar un cuarto de círculo a partir de la posición 128,88: **PLOT 128,88:DRAW 64,64,PI/2**.
Con todo lo dicho hasta ahora estamos en condiciones de trazar dibujos bastante complicados. ¡Vamos a ver como nos sale este!:

```
  10 REM - ANAGRAMA - J.M. LOPEZ MARTINE
Z
  20 INK 2
  30 PLOT 49,95: DRAW 30,0: DRAW 8,-8,-P
I/2: DRAW 0,-7: DRAW -8,-8,-PI/2: DRAW 8
,-8,-PI/2: DRAW -7,0: DRAW -8,7,PI/2: DR
AW -16,0: DRAW 0,-7: DRAW -7,0: DRAW 0,3
1
  40 PLOT 56,87: DRAW 19,0: DRAW 4,-4,-P
I/2: DRAW -3,-3,-PI/2: DRAW -20,0: DRAW
0,7
  50 PLOT 104,95: DRAW 7,0: DRAW 0,-15:
DRAW 8,-8,PI/2: DRAW 8,0: DRAW 8,8,PI/2:
 DRAW 0,15: DRAW 7,0: DRAW 0,-23: DRAW -
8,-8,-PI/2: DRAW -23,0: DRAW -7,8,-PI/2:
 DRAW 0,23
  60 PLOT 160,64: DRAW 0,23: DRAW 7,8,-P
I/2: DRAW 24,0: DRAW 8,-8,-PI/2: DRAW 0,
-23: DRAW -7,0: DRAW 0,15: DRAW -8,8,PI/
2: DRAW -8,0: DRAW -8,-8,PI/2: DRAW 0,-1
5: DRAW -7,0
```

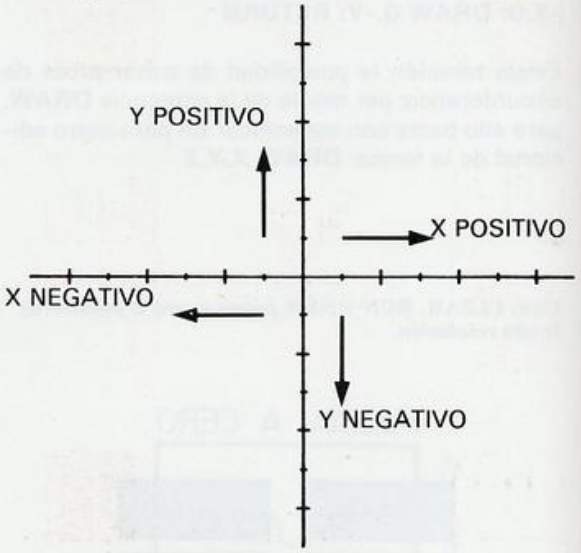
Los efectos conseguidos con **DRAW** pueden llegar a ser realmente curiosos, y si no lo creemos sólo tenemos que observar los siguientes ejemplos:

```
PLOT 90,80: DRAW 60,60,7689
PLOT 90,80: DRAW 60,60,8462
PLOT 90,80: DRAW 60,60,3572
PLOT 90,80: DRAW 60,60,1010
PLOT 90,80: DRAW 60,60,5387
PLOT 90,80: DRAW 60,60,3546
PLOT 90,80: DRAW 60,60,2547
PLOT 90,80: DRAW 60,60,4189
PLOT 90,80: DRAW 60,60,8888
```

LA SENTENCIA CIRCLE

La sentencia **CIRCLE** es la última de las encuadradas en el grupo de manejo de la alta resolución. Permite trazar círculos a partir de las coordenadas del centro y radio, especificados. Su forma general es: **CIRCLE X,Y,X,** donde **X** representa la abscisa e **Y** la ordenada del centro del círculo, siendo **Z** el radio.
Con **CIRCLE**, como con el resto de las sentencias del grupo, puede emplearse el color. La única limitación reside en el hecho de que cuando un *pixel* se conecta en color, tanto de **INK** como de **PAPER**, además de a él mismo afecta al resto de los 64 *pixels* que componen su cuadrícula (posición de impresión).
En el siguiente programa vemos un «olímpico» ejemplo de lo que podemos conseguir con esta sentencia:

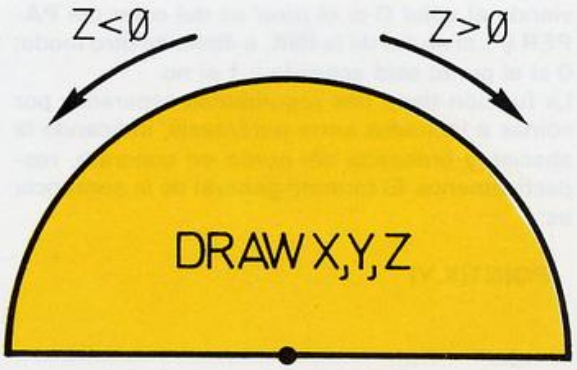
La sentencia* DRAW *admite tanto parámetros positivos como negativos.



¡!

Las sentencias de alta resolución del tipo absoluto son **PLOT**, **CIRCLE**, y **POINT**, que por tanto, sólo admiten parámetros positivos.

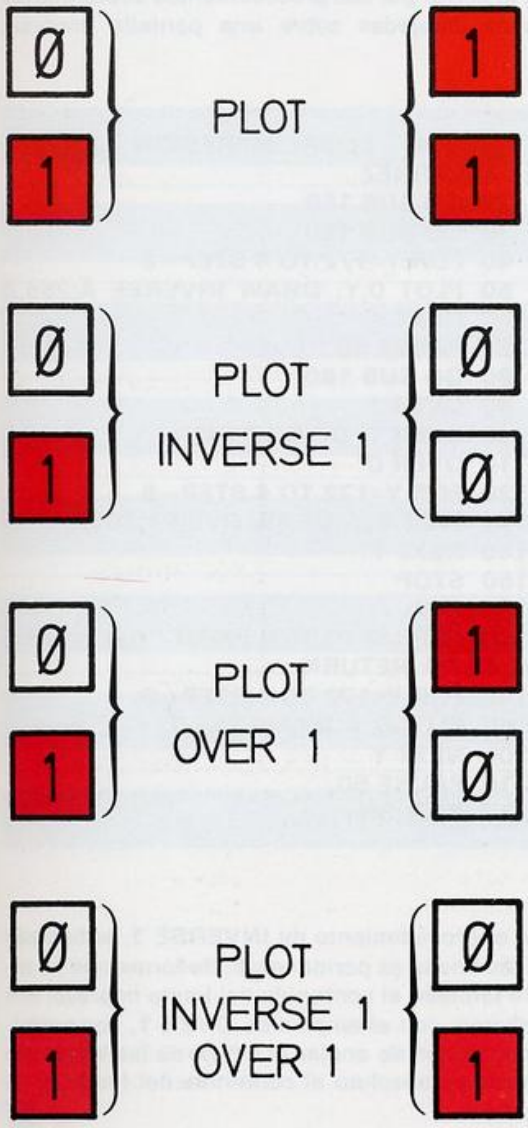
```
  10 REM - AROS OLIMPICOS - J.M. LO
PEZ MARTINEZ
  20 FOR X=77 TO 177 STEP 50
  30 READ J: INK J
  40 FOR I=0 TO 2: CIRCLE X,98,30+I:
NEXT I
  50 NEXT X
  60 FOR X=102 TO 152 STEP 50
  70 READ J: INK J
  80 FOR I=0 TO 2: CIRCLE X,58,30+I:
NEXT I
  90 NEXT X
 100 DATA 1,0,2,6,4
```



DRAW puede utilizar un tercer parámetro para trazar arcos circulares.

Ni más ni menos que la bandera olímpica ante nuestros ojos, aunque con algunas pequeñas deficiencias disculpables de color, debidas al hecho comentado anteriormente.

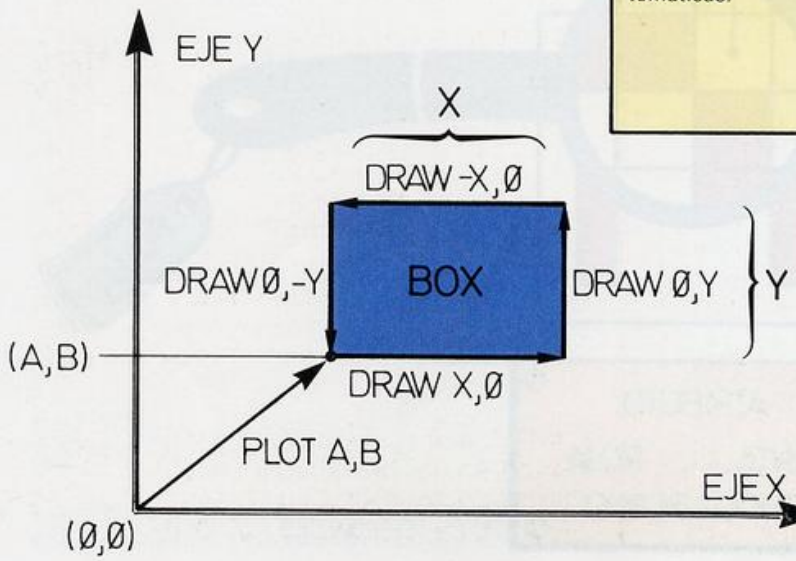
En base a la sentencia **PLOT** *y utilizando los atributos* **INVERSE** *y* **OVER** *podemos manejar de todas las maneras posibles los puntos de la pantalla.*



LA SENTENCIA POINT

Esta función suministra información sobre el color de un *pixel* determinado de la pantalla, devol-

Al rectángulo trazado en alta resolución en la pantalla se le denomina BOX.



¡!

El origen del puntero de alta resolución se encuentra en el *pixel* inferior izquierdo de la pantalla útil, perteneciente a la primera columna de la fila 23.

*

La sentencia **PLOT** nos permite controlar independientemente cada uno de los *pixels* de la pantalla.

*

La sentencia **DRAW** utilizada con sus dos primeros parámetros permite trazar líneas rectas; si se utiliza el tercer parámetro podemos obtener arcos circulares.

*

La sentencia **PLOT** se utiliza frecuentemente en la representación de funciones matemáticas.

¡!

Cualquier sentencia de alta resolución absoluta se puede convertir en relativa sumándole a las coordenadas X e Y, los **PEEK** de 23677 y 23678, respectivamente. De forma general, siendo X la coordenada horizontal e Y la vertical, un **PLOT** relativo podría ser **PLOT PEEK 23677+X, PEEK 23678+Y**; donde X e Y pueden adoptar valores positivos o negativos.

viendo el valor 0 si el *pixel* es del color del **PAPER** y 1 si es del de la **INK**; o dicho de otro modo: 0 si el punto está apagado y 1 si no.
La función tiene dos argumentos separados por comas e incluidos entre paréntesis, indicando la abscisa y ordenada del punto en concreto, respectivamente. El formato general de la sentencia es:

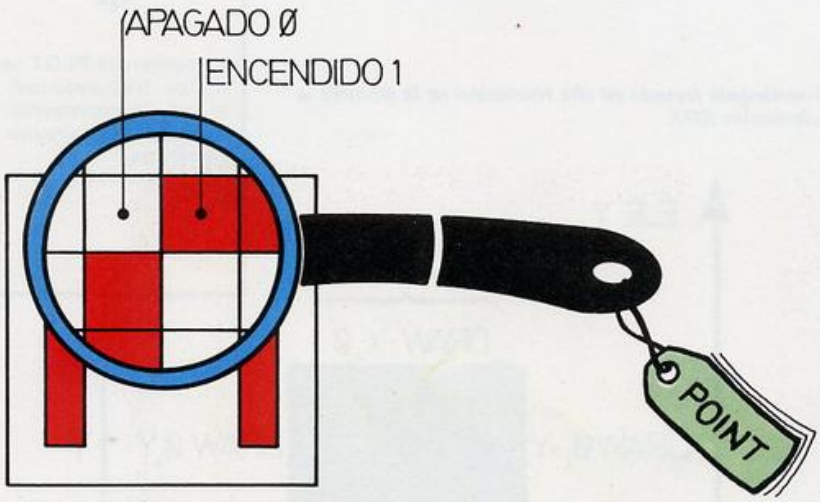
POINT(X,Y)

OTROS TRATAMIENTOS DE LOS *PIXELS*

Hasta ahora hemos hablado de como «encender» los puntos de la pantalla, sin comentar la forma de hacerlos desaparecer. Para ello existen diferentes posibilidades, dependiendo de lo que deseemos en cuestión.
PLOT INVERSE 1, se posiciona en el *pixel* indicado en el argumento y lo hace desaparecer, cambiando su color de **INK** por el del **PAPER**.
PLOT OVER 1, altera el estado del *pixel* a partir de su contenido actual, sea cual sea éste, es decir, si tenía el color de la **INK** pasa a tener el del **PAPER** y viceversa.
PLOT INVERSE 1 OVER 1, aparentemente no opera ningún cambio de estado en el *pixel*, aunque sí cambia el puntero de la alta resolución que quedará situado en las nuevas coordenadas.
La función desempeñada por **INVERSE** y **OVER**, para borrar lo dibujado por medio de las sentencias que conforman la alta resolución, hace que en muchas ocasiones sean sinónimos.
Sin embargo, sobre todo cuando se trata de dibujar sobre lo impreso, podemos ver más claramente la función que desempeña cada una de ellas.
En el siguiente programa vemos un ejemplo claro, al borrar por dos procedimientos diferentes las líneas dibujadas sobre una pantalla impresa:

```
 10 REM - SOBREIMPRESION - J.M. LOPEZ MARTINEZ
 20 GO SUB 160
 30 GO SUB 180
 40 FOR Y=172 TO 4 STEP-8
 50 PLOT 0,Y: DRAW INVERSE A;255,0
 60 NEXT Y
 70 PAUSE 50
 80 GO SUB 160
 90 OVER 1
100 OVER 1: GO SUB 180
110 OVER 0
120 FOR Y=172 TO 4 STEP-8
130 PLOT 0,Y: DRAW OVER 1;255,0
140 NEXT Y
150 STOP
160 CLS
170 FOR I=0 TO 703: PRINT "O";: NEXT I: PAUSE 50: RETURN
180 FOR Y=172 TO 4 STEP-8
190 PLOT 0,Y: DRAW 255,0
200 NEXT Y
210 PAUSE 50
220 RETURN
```

La función **POINT** *permite averiguar si un punto de la pantalla está encendido (color de* **INK***) o apagado (color de* **PAPER***).*

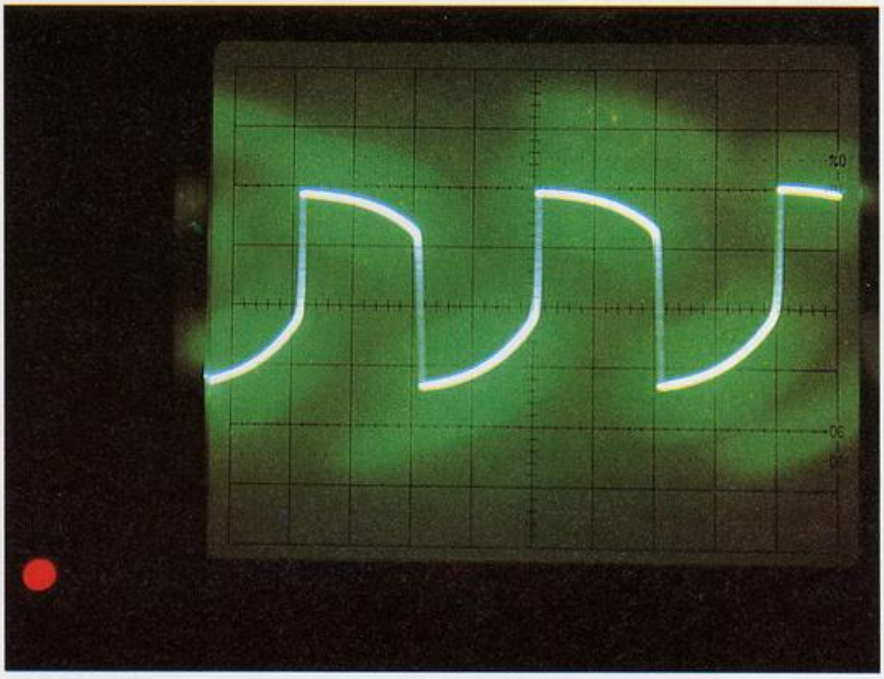


ATRIBUTO:
TINTA: ROJA
FONDO: BLANCO

Por el procedimiento de **INVERSE 1**, el borrado de las líneas es permanente, de forma que se altera también el contenido del fondo impreso. Sin embargo, con el empleo de **OVER 1**, nos permitimos el lujo de anular el dibujo de las líneas sin alterar en absoluto el contenido del fondo de la pantalla.

GRABACION DIGITAL

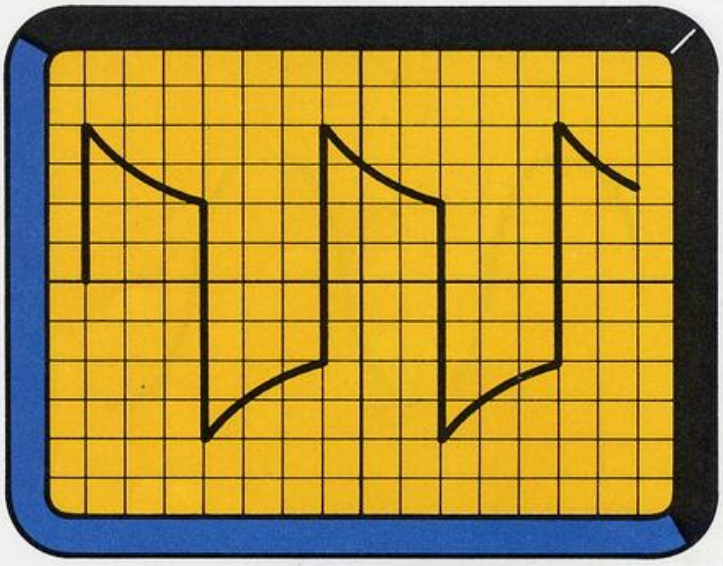
HEMOS comentado ampliamente en las páginas anteriores, las condiciones que deben reunir tanto las grabadoras como las cintas magnéticas empleadas en los procesos de transferencia de datos entre Spectrum y casete.
Analizaremos ahora la forma de comunicación utilizada por nuestro micro, para enviar y recibir información a través del casete, así como todos los circuitos internos que intervienen para llevar a buen término el proceso.
A estas alturas suponemos que estará suficientemente claro que el vocabulario manejado por nuestro Spectrum no va más allá de distintas combinaciones de unos y ceros (bits), agrupados de ocho en ocho (bytes). Estos últimos, combinados en cierto orden, conforman frases más o menos largas, las cuales denominamos programas.



Las tensiones correspondientes al uno y cero lógicos deben estar lo suficientemente separadas para evitar errores en la interpretación.

MAGNITUDES ANALOGICAS

Aspecto de la señal tomada de **EAR** *en la salida de casete.*



Dentro del mundo en el cual nos desenvolvemos, habitualmente realizamos ciertas medidas de algunas magnitudes de uso cotidiano, como la velocidad, la temperatura o, por ejemplo, la altura de las personas. Es fácil escuchar frases como: «El calor es sofocante. Al menos hay cuarenta grados a la sombra».
Pero efectuar un cálculo tan poco técnico supondrá, inevitablemente, un error bastante elevado. Podemos ayudarnos de un pequeño termómetro barométrico, pero igualmente, determinar si son 40 grados exactamente o algunas decimas de más o de menos, resultará del todo imposible. Aconsejados por algún maniático de la precisión, tal vez podamos tener acceso a un modernísimo equipo de los empleados en los centros de predicción meteorológica. Estos aparatos proporcionan medidas de la temperatura con varias cifras decimales.
Entonces, quizá nos sintamos satisfechos por co-

¡!

Magnitudes analógicas son aquellas que crecen y decrecen de manera constante.

*

Magnitudes digitales son aquellas que crecen y decrecen por valores constantes y de forma discontínua.

*

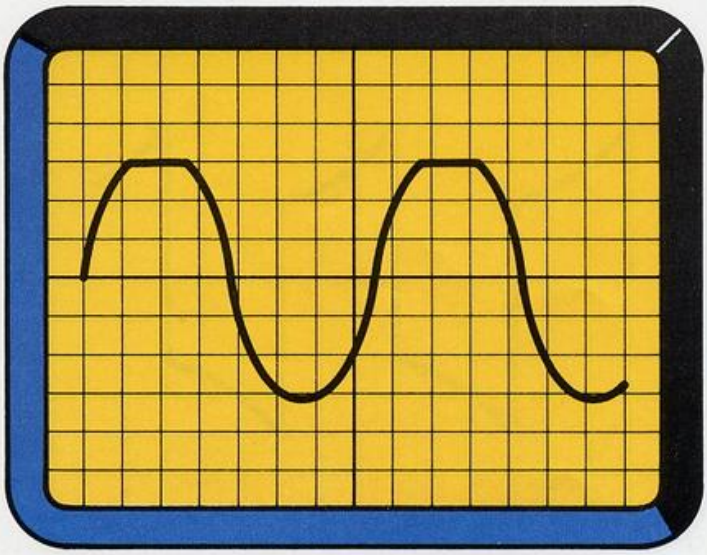
Los unos y los ceros deben ser claramente distinguidos en todos los procesos por el ordenador; de ahí que se empleeen magnitudes digitales, claramente diferenciadas en los dos valores.

A estas alturas suponemos que estará suficientemente claro, que el vocabulario manejado por nuestro Spectrum no va más allá de distintas combinaciones de unos y ceros (bits).

nocer la temperatura exacta reinante en ese momento. La primera consecuencia que debemos extraer, es que entre dos temperaturas se pueden dar infinitos estados diferentes. Es decir, uno de estos aparatos de precisión habrá detectado como al bajar y subir la temperatura lo hace de forma continua, atravesando infinitas temperaturas intermedias.

Y a esto es a lo que se denomina magnitud analógica, es decir, aquella que crece y decrece de forma continuada. ¿Pero qué demonios nos importa a nosotros si eran 39 ó 40 ó 40,745 grados?

Aspecto de la señal de entrada a la U.L.A. (pin 28) y salida hacia el casete tras el interface.



MAGNITUDES DIGITALES

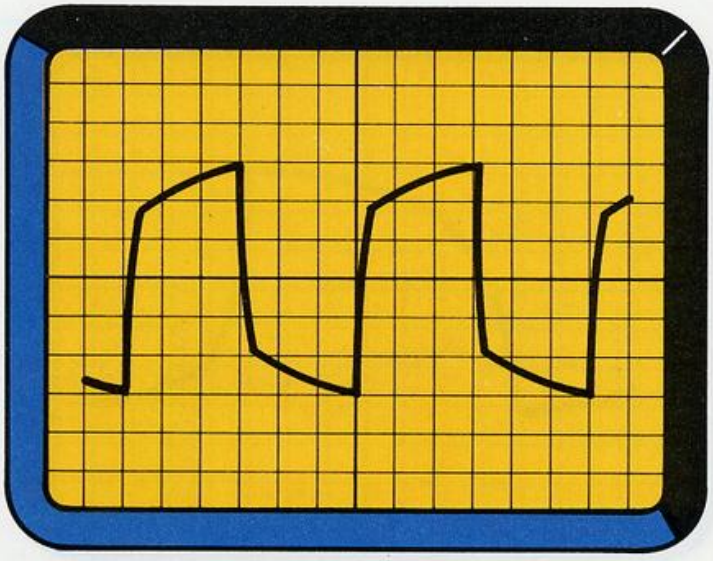
Puede que en temas como la temperatura no nos importe nada la precisión, y mucho menos lle-

gando al extremo de los decimales; sin embargo, para su correcto funcionamiento, el Spectrum no se puede permitir el lujo de la inexactitud en los procesos.

El funcionamiento de nuestro Spectrum está basado en la serie de impulsos eléctricos que circulan a través de sus circuitos. Como entre los cables y componentes electrónicos no pueden viajar ni palabras, ni marcianitos, sino sólo señales eléctricas, éstas deben de ser las menos posibles, con el objeto de evitar una posible confusión.

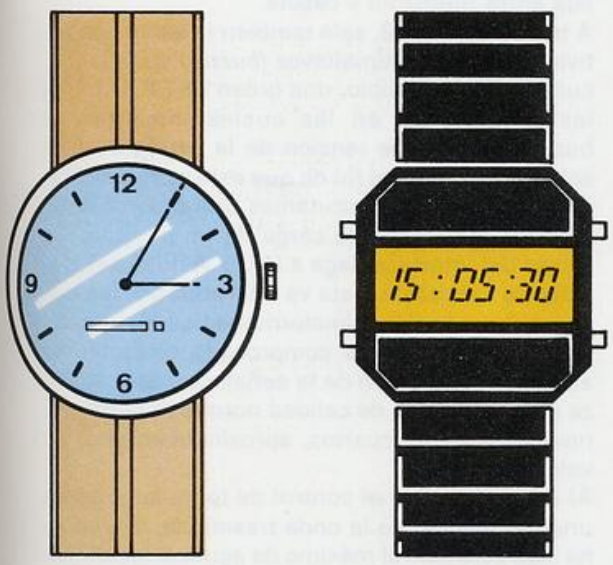
Esta es la razón por la cual las palabras de su vocabulario están siempre formadas por unos y ceros: es más fácil detectar dos tensiones que 256 diferentes, y para él debe quedar claro en todo momento cuál corresponde a un uno y cual a un cero. Además protestará enérgicamente, o lo que es lo mismo, no se enterará de nada, si intenta evaluar las tensiones que circulan entre sus circuitos de la misma manera que lo hacíamos con las magnitudes analógicas; es decir, creciendo y decreciendo de manera constante, pasando por infinitos estados intermedios.

Por ello, las tensiones correspondientes al uno y cero lógicos deben estar lo suficientemente separadas para evitar errores en la interpretación; por tanto, deben crecer o decrecer por impulsos de valor constante y forma discontinua, sin infinitos estados intermedios. Son lo que se denominan magnitudes o señales digitales: entre dos estados distintos, se produce siempre un salto fijo y determinado.



Cero lógico obtenido en el pin 28 de la U.L.A.

El tiempo es una magnitud continua (analógica); a pesar de ello, su medida puede ser tomada con aparatos analógicos o digitales.



MEDIDA ANALOGICA

MEDIDA DIGITAL

Lo mismo debe ocurrir con los sonidos que el Spectrum envía hacia el casete o recibe de éste. El proceso es siempre el mismo. Durante la grabación, las señales eléctricas deben convertirse en impulsos sonoros, y al contrario mientras dure la carga de un programa almacenado en cinta. Concretando, la frecuencia de las señales sonoras, o lo que es lo mismo, el tiempo que están activas, debe ser diferente para el cero y el uno lógico.

Como es habitual, estas frecuencias deben ser específicas y estar lo suficientemente separadas, de manera que no pueda caber el error de interpretación. La U.L.A. es la encargada de gestionar todo el proceso

TRANSMISION AL CASETE

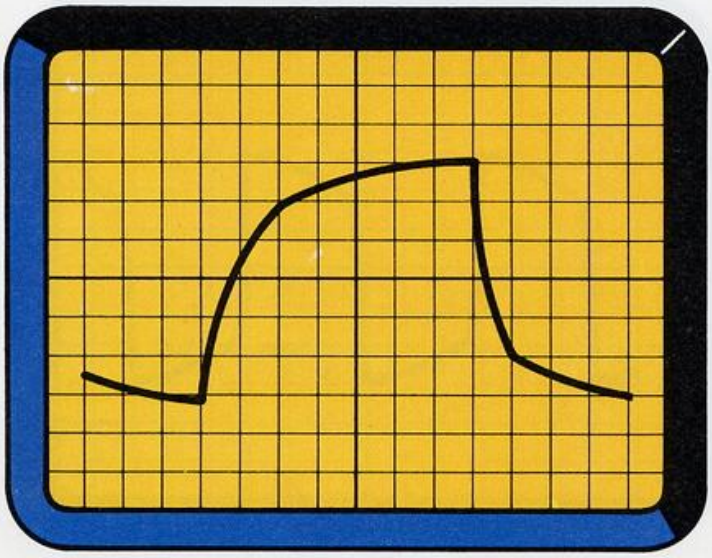
El proceso de transmisión de datos entre Spectrum y casete, se gestiona satisfactoriamente mediante una parte del *software* contenido en la R.O.M., denominado genéricamente *Cassette Handling Routines* (Rutinas de manejo del casete), que están comprendidas entre las direcciones

¡!

Esta misma distinción clara debe cumplirse entre los sonidos que el Spectrum recibe y emite desde y hacia el casete.

*

El proceso de transmisión de datos entre el Spectrum y el casete, se controla mediante una zona del *firmware* denominada *Cassette Handling Routines* (Rutinas de Manejo del Casete).

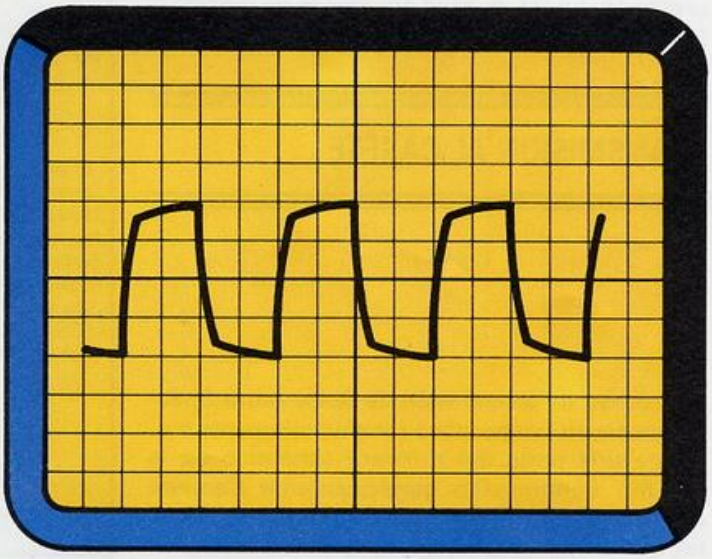


Uno lógico a la salida del **pin** *28.*

decimales 1218 y 2547. Este se encarga de descomponer cada byte en sus ocho bits respectivos, para ser enviados uno tras otro hacia la grabadora.

A través del *pin* (patilla) 28 de la U.L.A. *(Uncommited Logic Array)* salen en serie, a una velocidad media de 1500 baudios (bits/seg). Para evitar posibles confusiones, la frecuencia de los unos y los ceros es distinta. Como podemos observar en la fotografía tomada de la pantalla de un osciloscopio, la frecuencia del cero, del orden de los 2000 Hertzios, es doble de la del uno, aproximadamente 1000 Hertzios.

*Aspecto de la onda producida por la cabecera de transmisión (***pin** *28).*



El tono guía *(leader)* que emite el Spectrum previamente al envío de los datos, tiene por objeto elevar al nivel correcto el control del volumen de grabación en los casetes que lo tienen automático, con el fin de evitar saturaciones en la cinta por ser el volumen demasiado alto, o impedir que las grabaciones se realicen a niveles sonoros bajos que imposibiliten su reconstrucción.

Sobre la información contenida en la cabecera, tras el tono guía, nos remitimos al comentario que se hace sobre el tema en el programa INDEX. El *leader* no puede ser nunca confundido con un bloque de información, dado que su frecuencia es menor que la de uno, y por tanto que la del cero.

En las fotografías, observamos el aspecto de las señales digitales que la U.L.A. envía hacia el casete. Estas tienen forma de pulso casi cuadrado, pero el casete no es capaz de manejar señales digitales, debido a lo cual es necesario disponer de un interface entre ambos, que sea capaz de mediar en el proceso.

Otra característica a tener en cuenta es la ausencia de tiempos muertos entre la señal que compone un impulso y el siguiente. Gracias a ello, se evita la posibilidad de interferencia por ruido eléctrico, asociada a todo proceso en el cual se utilice la cinta magnética, y la inevitable profusión de errores en la interpretación, que por esta causa se podrían provocar.

Los resultados obtenidos usando el osciloscopio, muestran que el interface amplifica ligeramente la señal procedente de la U.L.A., a la vez que adapta impedancias (resistencia) de entrada/salida entre Spectrum y casete.

A través del *pin* 28, sale también la señal que activa el pequeño minialtavoz *(buzzer)* cuando ejecutamos, por ejemplo, una orden **BEEP**. Durante las operaciones en las cuales interviene el buzzer, el nivel de tensión de la patilla 28 está sobreelevado con el fin de que este suene, no haciéndolo mientras ejecutamos una orden **SAVE**.

Durante el proceso de carga de un programa, a través del interface llega a la U.L.A. la señal proviniente del casete. Esta va interpretando las distintas frecuencias transformándolas en unos y ceros lógicos. Hemos comprobado conforme se aumenta el volumen de la señal, que éste alcanza para un casete de calidad normal el punto óptimo en los tres cuartos, aproximadamente, del volumen total.

Al accionar sobre el control de tono, se produce una atenuación de la onda trasmitida, si éste no ha sido colocado al máximo de agudos. Igualmente, al manejar casetes estéreo, el menor desajuste en las cabezas provoca distorsiones en las señales transmitidas.

MOTOS DE LUZ

SEGURAMENTE, muchos de nosotros hemos oído hablar, o hemos llegado a ver la película TRON; un homenaje del Séptimo Arte al mundo de la informática.

El protagonista de su argumento es un competente programador: Flynn, que a causa de las artimañas del malvado CCP (Control Central de Programas), se materializa en el interior del ordenador de una gran empresa de *software* para videojuegos, intentando descubrir al «pirata» que le espía sus creaciones.

Dentro de la enorme máquina, convertido en una instrucción, va en la búsqueda de la instrucción TRON, que envió poco antes de ser desintegrado por el CCP, para realizar una investigación. En su recorrido por los buses, memorias, *ports*, etc... Flynn se ve amenazado por sus propias creaciones, debiendo competir por su existencia en el duelo de los aros de energía, o en la mortal carrera de las motos de luz.

MOTOS DE LUZ

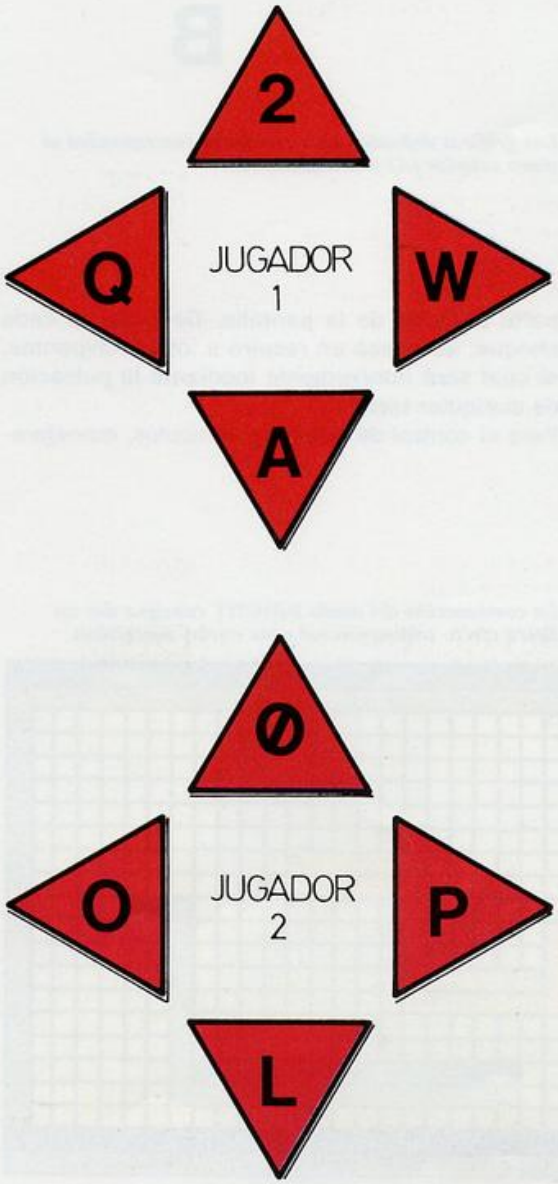
Nuestro programa recrea en la pantalla del Spectrum el duelo que Flynn y el CCP protagonizan en las fantásticas motos de luz. El tablero de juego es una malla que ocupa la práctica totalidad de la pantalla, rodeada de unos muros de energía, con los cuales el choque es mortal.

El juego necesita, evidentemente, el concurso de dos participantes, los cuales guiarán sus motos de luz intentando que el contrincante se estrelle contra algún muro de energía.

Estos especiales vehículos van dejando a su paso un sólido rastro energético, de forma que debemos intentar cercar a nuestro oponente, para conseguir que se choque, ya sea contra nuestro haz de energía, contra los muros, o contra su propia estela energética; naturalmente, todo ello antes de que nosotros cometamos alguno de los mencionados errores.

El vencedor del juego es aquel que consigue llegar a diez puntos antes que el contrario, teniendo en cuenta que cada choque proporciona un punto al superviviente. La posición de inicio de

En el gráfico se detallan las teclas para el movimiento de los vehículos de cada uno de los jugadores.



¡!

Los caracteres que aparecen en el listado con un subrayado simple, deben ser introducidos como los caracteres gráficos de las teclas correspondientes.

*

Los caracteres que figuran doblemente subrayados corresponden a los gráficos cambiados, es decir, los que se obtienen pulsando simultáneamente **CAPS SHIFT** y la tecla afectada, en el modo **GRAPHICS**.

*

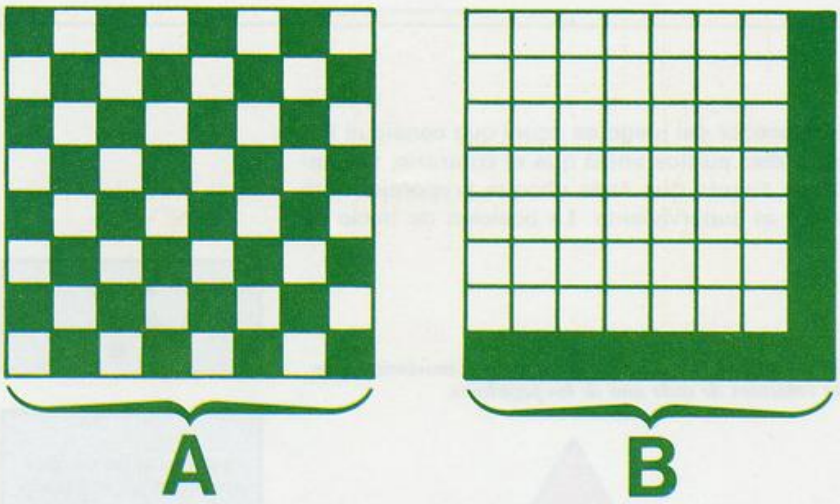
Para la toma de datos se ha recurrido a la lectura directa del teclado mediante la función **IN**.

*

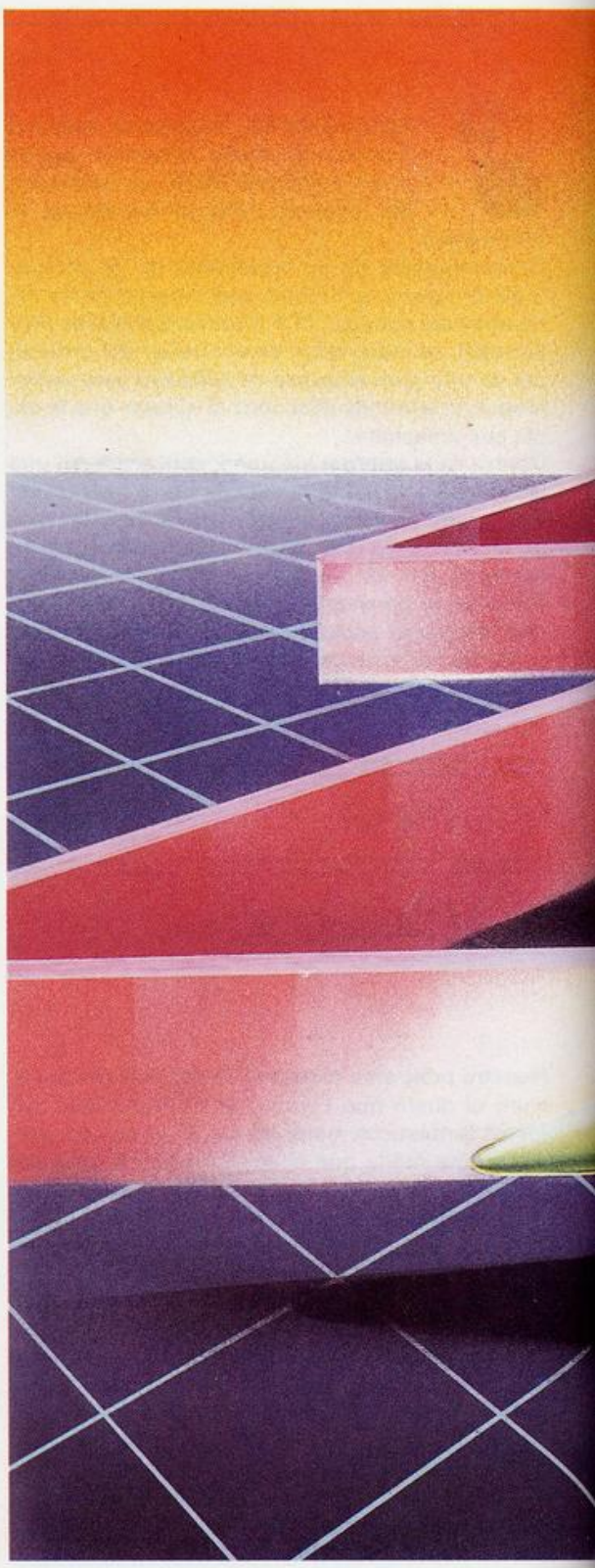
Para la grabación del programa procederemos del modo habitual, mediante el comando **SAVE "MOTOS LUZ"**, o bien, con la opción de autoejecución **SAVE "MOTOS LUZ" LINE 10**.

cada juego es siempre la misma: el jugador uno, que manejará la parte izquierda del teclado, aparece en la fila diez y columna cuatro de la pantalla, y su oponente, en la misma fila, pero en la parte derecha de la pantalla. Después de cada choque, se ofrece un respiro a los participantes, el cual será interrumpido mediante la pulsación de cualquier tecla.

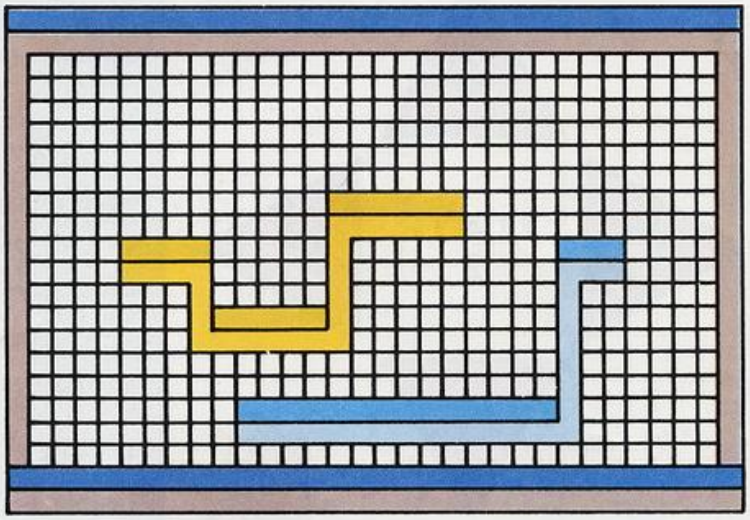
Para el control de nuestros vehículos, manejaremos cuatro teclas: el jugador uno, utilizará las teclas 2, Q, W y A, para arriba, izquierda, derecha y abajo, respectivamente; su oponente, controlará las teclas 0, O, P y L, con los mismos significados respectivos. A este respecto, las motos co-



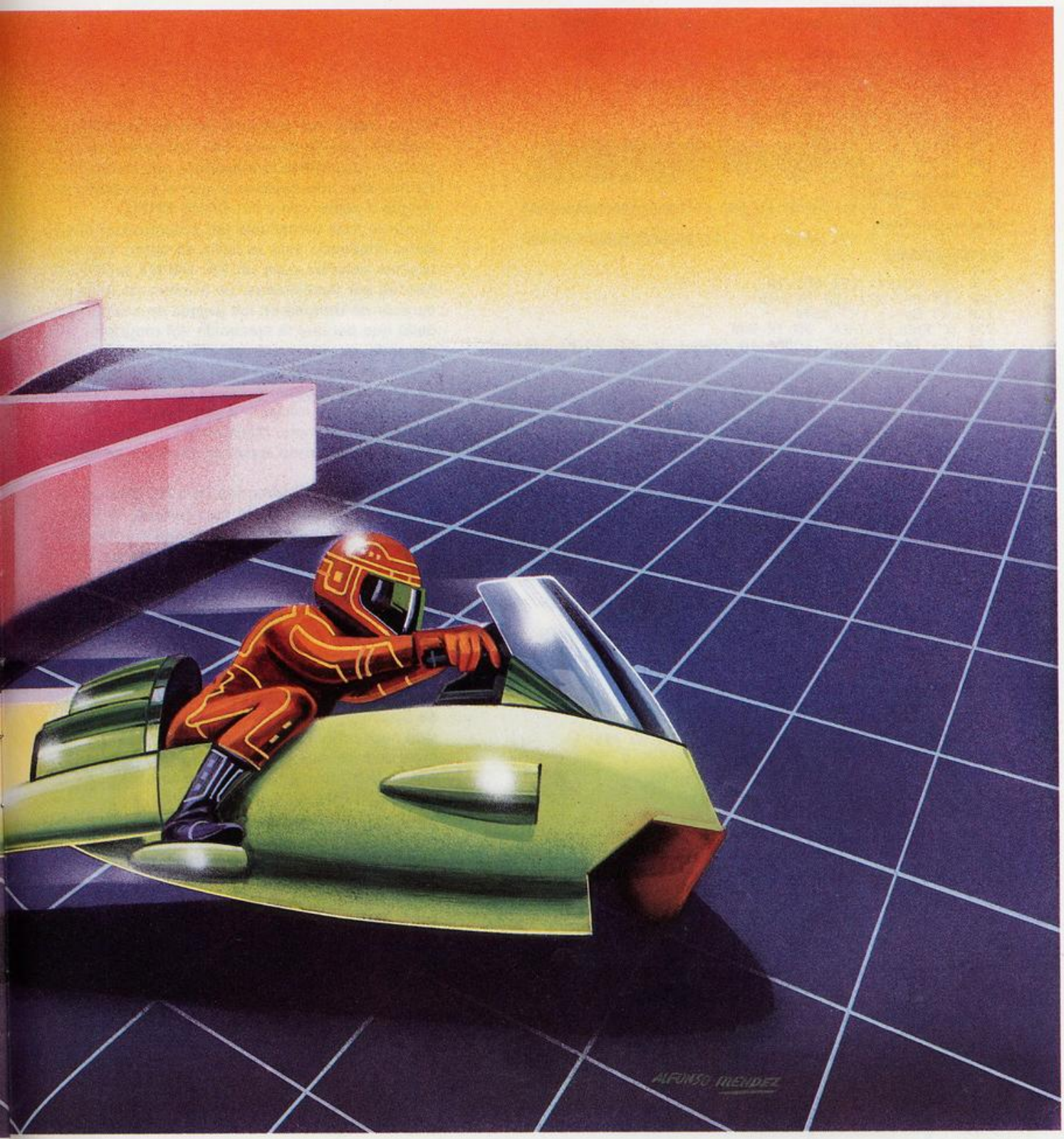
Los gráficos definidos para este juego corresponden al muro exterior (A) y al mallado (B).



La combinación del modo **BRIGHT** *consigue dar un ligero efecto tridimensional a las estelas energéticas.*



mienzan siempre con un mismo sentido de desplazamiento, que no alteran hasta que se pulsa una y sólo una de las teclas a tal fin.
Antes de cada partida podremos elegir los colores de nuestras motos de luz, comprendidos entre el dos (rojo) y el siete (blanco), dado que el cero y el uno (negro y azul) serían difícilmente distinguibles sobre el tablero, y siempre y cuando cada jugador escojamos un color diferente, para evitar confusiones.



```
  10 REM ************************
  20 REM * J.M. MAYORAL SERRANO *
  30 REM ************************
  40 PAPER 1: BORDER 1: INK 7: CLS : POKE 23609,30: P
OKE 23562,1
  50 FOR I=0 TO 7 STEP 2: POKE USR "A"+I,170: POKE US
R "A"+I+1,85: NEXT I
  60 FOR I=0 TO 6: POKE USR "B"+I,1: NEXT I: POKE USR
 "B"+7,255
  70 INPUT "COLOR PARA JUGADOR 1 ? (2 A 7) ";C1
  80 IF C1<2 OR C1>7 THEN  GO TO 70
  90 INPUT "COLOR PARA JUGADOR 2 ? (2 A 7) ";C2
 100 IF C2<2 OR C2>7 THEN  GO TO 090
 110 IF C1=C2 THEN  GO TO 70
 120 LET P1=0: LET P2=0
 130 PRINT AT 1,0;"8888888888888888888888888888888888"
 140 PRINT AT 20,0;"8888888888888888888888888888888888"
 150 PRINT  PAPER 2; INK 6;AT 21,0;"AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA"
 160 PRINT AT 0,0; PAPER 2; INK 6;"AAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA"
 170 FOR I=2 TO 19: PRINT AT I,0;"8BBBBBBBBBBBBBBBBB
BBBBBBBBBBB8"
 180 NEXT I
 190 LET PC1=4: LET PF1=10
 200 LET PC2=27: LET PF2=10
 210 LET D1=2: LET D2=4
 220 IF RND<.5 THEN  GO TO 330
 230 LET T$="": POKE 16384,IN 63486: IF NOT POINT (6,
175) THEN  LET T$="1"
 240 POKE 16384,IN 64510: IF NOT POINT (7,175) THEN
LET T$=T$+"4"
 250 IF NOT POINT (6,175) THEN  LET T$=T$+"2"
 260 POKE 16384,IN 65022: IF NOT POINT (7,175) THEN
LET T$=T$+"3"
 270 IF LEN T$=1 THEN  LET D1=VAL T$
 280 LET PC1=PC1+(D1=2)-(D1=4): LET PF1=PF1+(D1=3)-(D
1=1)
 290 IF SCREEN$ (PF1,PC1)<>"" THEN  GO TO 470
 300 PRINT AT PF1,PC1; BRIGHT 1; INK C1;"8"
 310 IF SCREEN$ (PF1+1,PC1)="" THEN  PRINT AT PF1+1,P
C1; PAPER C1; INK C1;"A"
 320 BEEP .01,0
 330 LET T$="": POKE 16384,IN 61438: IF NOT POINT (7,
175) THEN  LET T$="1"
 340 POKE 16384,IN 57342: IF NOT POINT (6,175) THEN
LET T$=T$+"4"
 350 POKE 16384,IN 57342: IF NOT POINT (7,175) THEN
LET T$=T$+"2"
 360 POKE 16384,IN 49150: IF NOT POINT (6,175) THEN
LET T$=T$+"3"
 370 IF LEN T$=1 THEN  LET D2=VAL T$
 380 LET PC2=PC2+(D2=2)-(D2=4): LET PF2=PF2+(D2=3)-(D
2=1)
 390 IF SCREEN$ (PF2,PC2)<>"" THEN  GO TO 440
 400 PRINT AT PF2,PC2; BRIGHT 1; INK C2;"8"
 410 IF SCREEN$ (PF2+1,PC2)="" THEN  PRINT AT PF2+1,P
C2; PAPER C2; INK C2;"A"
 420 BEEP .01,10
 430 GO TO 230
 440 PRINT AT PF2,PC2; FLASH 1;" "
 450 LET P1=P1+1
 460 GO TO 490
 470 PRINT AT PF1,PC1; FLASH 1;" "
 480 LET P2=P2+1
 490 FOR F=30 TO 60: BEEP .001,F: NEXT F
 500 PRINT  INVERSE 1;AT 9,10;"JUGADOR 1:";P1
 510 PRINT  INVERSE 1;AT 11,10;"JUGADOR 2:";P2
 520 FOR I=60 TO 40 STEP -1: BEEP .001,I: NEXT I
 530 IF P1>9 OR P2>9 THEN  GO TO 580
 540 IF INKEY$<>"" THEN  GO TO 540
 550 PRINT )0; FLASH 1; PAPER 5; INK 9;" PULSA UNA TE
CLA PARA CONTINUAR "
 560 IF INKEY$<>"" THEN  CLS : GO TO 130
 570 GO TO 560
 580 PRINT  FLASH 1; PAPER 6; INK 2;AT 13,4;"    GANA
 EL JUGADOR ";(P2=10)+1;"   "
 590 INPUT "OTRO JUEGO ? "; LINE X$
 600 IF X$="" THEN  GO TO 590
 610 IF X$(1)<>"N" THEN  RUN
```

EL PROGRAMA

El programa no presenta para su introducción grandes complicaciones. Tan solo aparecen unos gráficos (caracteres subrayados), que como es habitual serán introducidos, según el subrayado sea simple o doble, con o sin **CAPS SHIFT**.

El punto más destacable del programa es el sistema empleado para la toma de datos. Las sentencias usuales para tal fin: **INPUT** e **INKEY$** han de ser desechadas. La primera de ellas por carecer de utilidad en los juegos de tiempo real, dado que detiene la ejecución del programa; y la segunda, porque no es capaz de registrar la pulsación de más de una tecla al mismo tiempo, y aunque el programa ignore la pulsación simultánea de más de una tecla a cargo de un mismo jugador, si contempla, lógicamente, que cada jugador pulse su mando correspondiente a un mismo tiempo.

Dados estos condicionantes, ha sido necesario recurrir a la función **IN**, la cual permite la lectura de las direcciones del *port* correspondientes al teclado. Cada dirección afecta a una de las veinte semifilas del teclado del Spectrum, cada una de los cuales comprende cinco teclas; así pues, del byte leído, sólo tienen sentido para nosotros cinco bits, concretamente los menos significativos (del cero al cuatro).

Puesto que los restantes bits del byte leído pueden adoptar diferentes valores, según la versión de Spectrum en que se ejecute el programa, ha sido necesario hacer un análisis del byte no a su nivel de byte sino de bit, para lo cual se ha recurrido a la traducción directa que se consigue realizando un **POKE** en la pantalla, y que puede ser fácilmente investigada mediante la función **POINT**.

El hecho de que se vayan consultando todas las posibles pulsaciones de los mandos de cada jugador, y almacenándolos en una variable de cadena (**T$**), se debe a la consulta que después se hace de la longitud de la misma, decidiéndose no alterar el sentido de la marcha en dos casos: la longitud es cero (no se ha pulsado ninguna tecla), o bien la longitud es mayor que uno (se ha pulsado más de un mando del mismo jugador).

Como es habitual, para la grabación del programa emplearemos el comando **SAVE**, ya sea en su forma simple **SAVE "MOTOS LUZ"**, ya sea con la opción de autoejecución **SAVE "MOTOS LUZ" LINE 10.**