Tercera Parte

Juegos

CAPÍTULO SÉPTIMO

Los juegos que juega la gente

Escribir buenos juegos en un ordenador, es indiscutiblemente un arte. Por eso es bastante difícil conseguirlo, leyendo simplemente unas pocas páginas de un libro. Sin embargo, espero dar aquí los consejos necesarios para colocarle en el buen camino.

En general se pueden programar dos tipos de juegos. Los primeros se basan en el movimiento de gráficos, y en ellos son muy importantes las rutinas para mover caracteres en la pantalla, dibujos bien hechos, etc. El segundo tipo es el juego de aventuras, donde las preguntas y el análisis de las respuestas están al orden del día. En el primer tipo también tienen mucha importancia los trucos para aumentar la velocidad del juego (especialmente si se escribe en BASIC); mientras que el segundo tipo, necesita de una planificación lógica, profunda, para que no se convierta en un juego tonto y simple. A estos dos tipos de juegos, los llamaremos respectivamente, juegos gráficos y juegos verbales.

JUEGOS VERBALES

Hay que tener en cuenta dos aspectos importantes a la hora de escribir juegos verbales: por un lado está todo el proceso lógico que debe determinar en qué lugar se encuentra el jugador (por ejemplo, en qué parte del laberinto), y por otro, está la programación del orden que deben seguir los problemas o tesoros con los que se encuentra a lo largo del juego. También son de vital importancia aspectos como: los criterios para aceptar o deshechar una res-

puesta, hacer las preguntas lo más claras posible, no perder de vista las "posesiones", la puntuación, etc.

LAS RESPUESTAS

En la primera parte del libro ya vimos cómo se puede hacer para detectar si un jugador había respondido "YES" a una pregunta. Uno de los métodos, simplemente consistía en dimensionar un conjunto con un solo elemento [por ejemplo, R$ (1)] y luego mirar si la variable era igual a "Y". Este método no era muy bueno puesto que se dejaba algunas posibilidades, como por ejemplo, que el jugador incluya un espacio (por error) al responder, o que su respuesta empiece por "y" pero que no signifique "YES".

El primer problema se solucionó con una rutina que buscaba las letras "YES" en cualquier lugar de la cadena que se daba como respuesta. Lo que no se incluyó para no complicar el programa es el hecho de que las letras Y E S pueden formar parte de una palabra más larga que no significa YES. Esto se puede solucionar comprobando que las letras YES se encuentran rodeadas de espacios. Aquí hay una manera de hacerlo aunque posiblemente no es la más elegante. Nota: asegúrese de que tiene puestas las mayúsculas (CAPS LOCK):

```
10 INPUT "OTRA VEZ?"; R$
20 IF R$ = "YES" THEN GOTO 100
30 IF LEN R$<3 THEN GOTO 80
40 IF LEN R$ = 4 THEN GOTO 200
50 FOR X = 2 TO LEN R$ - 1
60 IF R$(X TO X + 4) = " YES " THEN GOTO 100
70 NEXT X
80 PRINT "ADIOS": STOP
90 STOP
100 PRINT "DE ACUERDO, CONTINUO"
110 GOTO . . . (programa)
200 IF R$ ( TO 3) = "YES" THEN GOTO 100
210 IF R$ (2 TO) = "YES" THEN GOTO 100
220 GOTO 80
```

Esto se podría mejorar añadiendo un espacio al principio y al final de la respuesta, porque si la palabra YES se

encuentra por ejemplo al principio, la comparación de "YES " con " YES " daría negativa.

Comprobar si el jugador ha respondido sólo con letras es muy sencillo, usando el hecho de que las cadenas se pueden comparar así:

```
10 INPUT "Por dónde vas?", D$
20 FOR N = 1 TO LEN D$
30 IF D$(N)>"Z" OR D$(N)<"A" THEN GOTO 10
40 NEXT N
```

Pero hay que tener cuidado, ya que el Spectrum diferencia entre mayúsculas y minúsculas, y para tener esto en cuenta habría que ampliar así la línea 30:

```
30 IF CODE D$(N)>122 OR CODE D$(N)<65
   THEN GOTO 10
```

Esto comprueba todos los caracteres exceptuando los gráficos cuyos códigos van del 91 al 95 (corchetes, flecha, etc.). Pero se puede considerar esto suficientemente seguro, aunque si quiere puede añadir lo necesario para evitar que se introduzcan estos caracteres. Las entradas numéricas también se pueden comprobar usando CODE:

```
10 INPUT N$
20 FOR X = 1 TO LEN N$
30 IF CODE N$(X)>57 OR CODE N$(X)<48 THEN
   GOTO 10
40 NEXT X
50 LET N = VAL N$
```

Observe cómo uso INPUT para entrar los números como cadenas, y cuando han superado la comprobación se convierte la cadena en un número, usando VAL. Sin embargo, hay un problema al hacer INPUT con cadenas, y es que las comillas se imprimen después de INPUT y pueden ser borradas por accidente. Además, cualquier persona podría ver el listado de su programa, fácil y simplemente, borrando la comilla de la izquierda y pulsando STOP. Felizmente, el Spectrum tiene una manera de solucionar esto, usando LINE:

```
10 INPUT "¿DE QUÉ MODO", LINE A$
```

Esto coloca el mensaje con el interrogante en la parte baja de la pantalla, pero no las comillas. Si usted ha hecho un buen trabajo filtrando las respuestas que pueden causar la detención del programa por error, entonces será muy difícil para el programa, y casi imposible que esto suceda por accidente.

Otra cosa que puede pasar es que alguien no conteste absolutamente nada, sino que como toda respuesta pulse la tecla ENTER. Para evitar que se tome como respuesta, añada la línea:

```
15 IF A$ = ""THEN GOTO 10
```

Observe que no hay espacios entre las dos comillas. Esto detecta cuando la respuesta es nula, y vuelve atrás para pedirla de nuevo.

También se puede detectar cuando alguien escribe 2*3 en lugar de 6. Esto se consigue utilizando este tipo de sentencias:

```
10 INPUT C$
20 IF C$ < > STR$ VAL C$ THEN PRINT
"Escríbalo de nuevo"
```

Esto funciona mientras que la persona que use el programa introduzca sólo combinaciones de números y símbolos, pero el programa se para con un mensaje de error en el momento de evaluar la función VAL si la cadena C$ contiene letras o símbolos no matemáticos.