

## 14. Textos II

La escritura de textos fijos resulta interesante, pero no nos servirá de gran ayuda si no somos capaces de escribir información sobre valores de variables como puede ser la puntuación o la vida.

Vamos a aprender un poco más sobre textos y ya estaremos casi listos para escribir cualquier tipo de información por pantalla.

### 14.1 Escribir valores de variables en pantalla

La función `write_var()` tiene un funcionamiento prácticamente idéntico a la función `write()` que hemos estudiado en el tema anterior. La única diferencia es su último parámetro, ya que en lugar de tratarse de una cadena de texto entre comillas, en ese parámetro podemos poner el nombre de cualquier variable de nuestro videojuego, y éste será escrito en pantalla.

¿Qué ocurre si el valor de la variable cambia? Pues no hay ningún problema, ya que `write_var()` es una función que se encarga de actualizar automáticamente el valor, sin mayores complicaciones.

Vamos a sacarle un poco más de partido al uso de los valores de centrado. Imagina que quieres colocar en la parte superior de la pantalla un texto fijo "PUNTOS" y a su derecha el valor de la variable `puntos`.

Observa que el valor de la variable puede crecer más y más a medida que avanza el juego, y no sería en absoluto deseable que el número pase a sobrescribir el texto "PUNTOS". El valor de centrado nos ayudará a evitar ese tipo de cosas si centramos ambos textos en el mismo punto pero con distinta alineación horizontal, por ejemplo así:

```
BEGIN
...
write      ( fuente_texto, 400, 0, 2, "PUNTOS: ");
write_var  ( fuente_texto, 400, 0, 0, puntos );
...
END
```

### 14.2 Una especificación más formal de la sintaxis de `write()`

En el tema anterior hemos comentado que la escritura de textos consume muchos recursos y no es recomendable incluir instrucciones `write()` dentro de un bloque LOOP-END.

En algunas ocasiones puede interesarnos tener un texto capaz de moverse por la pantalla. En esos casos no nos queda más remedio que utilizar continuamente la función write() con distintas coordenadas x,y dentro de un LOOP, pero a cada nueva llamada a la función write() necesitaremos borrar el texto escrito anteriormente...

Para eso write() nos **retorna** un identificador numérico del texto, que podemos pasar como parámetro a la función delete\_text() para borrar el texto escrito. Formalmente la sintaxis de write() y delete\_text() es la siguiente:

```
int write ( int, int, int, int, string)           // write retorna un identificador numérico
delete_text (int)                               // delete_text recibe el identificador numérico retornado por write()
```

El funcionamiento es algo especial, ya que la escritura debe realizarse antes de la sentencia FRAME; y el borrado después, para asegurar que el texto se borra después de haber sido mostrado, y no antes.

Para probar todo esto vamos a crear un nuevo proceso encargado de mostrar el nombre de nuestro videojuego de izquierda a derecha de la pantalla:

```
PROCESS titulo()
PRIVATE
    int texto;
BEGIN
    x = 0;
    y = 300;
    LOOP
        x = x + 10;
        IF ( x > 1600 )
            BREAK;
        texto = write ( fuente_texto, x, y, 5, "BIENVENIDO A MI VIDEOJUEGO" );
        FRAME;
        delete_text ( texto );
    END
END
```

Observa que comprobamos cuando la coordenda x es suficientemente grande como para que el texto haya salido por la derecha de la pantalla, y es entonces cuando el proceso muere para no seguir escribiendo más el título.