

## 22. Animaciones I

No hay lugar a dudas de que hasta ahora has estado intentando conseguir cambios de animación especialmente en tu personaje protagonista. En este tutorial hemos hecho poco más que explicar cómo controlar una única animación, generalmente la de "andar", pero nos interesa que nuestros personajes se mantengan quietos, anden, salten, disparen, etc.

Si hemos esperado hasta ahora es porque queríamos que comprendieses perfectamente cómo gestionar las animaciones, ya que habrás comprobado que no es tan sencillo como podía parecer en un principio.

Para ello hemos separado en dos temas esta parte: En este primer tema aprenderemos a gestionar las animaciones gracias a variables y un proceso auxiliar, mientras que en el siguiente te ofreceremos un fichero de código .prg con todo hecho que sin duda te facilitará muchísimo las cosas.

### 22.1 Variables necesarias para gestionar una animación

El simple uso de asignaciones, incrementos y condiciones sobre la variable local graph de todo proceso en Benu no ofrece todo lo necesario para una correcta gestión de animaciones complejas.

Necesitaremos algunas variables auxiliares, y para comenzar poco a poco tendremos únicamente dos en la sección PRIVATE de todo proceso en el que queramos tener cambios de animación:

```
PRIVATE
    int graph_ini;
    int graph_fin;
```

Observa que los nombres graph\_ini y graph\_fin son nombres inventados, podíamos haber usado otros, pero son suficientemente representativos para la utilidad que van a tener.

Partiremos de que inicialmente las variables graph\_ini y graph\_fin guardan los valores que corresponden a la animación de "quieto" de nuestro personaje, en el caso de nuestro protagonista tendremos:

```
PRIVATE
    int graph_ini = 1;
    int graph_fin = 9;
```

Hasta ahora, para animar a nuestro protagonista teníamos varias instrucciones de incremento del número de gráfico como ésta:

```
graph = graph + 1;
```

Y esa instrucción venía siempre seguida de una comprobación para que el número de gráfico no se pasase del intervalo correcto, por ejemplo así:

```
IF ( graph > 9 )  
    graph = 1;  
END
```

Vamos a olvidarnos de toda esa parte, y haremos que nuestro protagonista **NO** vuelva a hacer referencia a su variable graph, sino que será un proceso el que se haga cargo de ello. Nuestro protagonista a cada FRAME invocará el siguiente proceso:

```
PROCESS animar ( int primero , int ultimo )  
BEGIN  
    father.graph = father.graph + 1;  
    IF ( ( father.graph < primero ) OR ( father.graph > ultimo ) )  
        father.graph = primero;  
    END  
END
```

La anterior función, dados dos valores de número de gráfico, se ocupa de incrementar en 1 el número del gráfico del proceso padre, y a continuación, comprobar si su número de gráfico o bien se pasa del valor dado por "último" o bien es menor que el valor dado por "primero", para retornar en ese caso al valor dado por "primero".

### ¿Qué significa esto?

Que si desde nuestro proceso protagonista invocamos continuamente la función animar dando como parámetros el valor de graph\_ini y graph\_fin tendremos a nuestro protagonista constantemente realizando la animación de quieto:

```
LOOP  
...  
    animar ( graph_ini , graph_fin );  
...  
END
```

## 22.2 Implementar cambios en la animación

Recuerda que hemos dicho que **nuestro protagonista ya no se ocuparía nunca más de modificar el valor de su variable LOCAL graph**, sino que ahora sería el proceso animar ( ) el encargado de ello.

Según lo anterior, observa que tal y como tenemos ahora mismo nuestro videojuego nuestro protagonista siempre se anima entre los valores de número de gráfico dados por graph\_ini y graph\_fin. Esto supone una ventaja, y es que si queremos modificar la animación de nuestro personaje basta con que modifiquemos los valores de graph\_ini y graph\_fin y la invocación de la función animar ( ) se encargará de hacer que nuestro personaje se anime en el intervalo que queramos.

Un ejemplo básico para hacer que nuestro protagonista se mantenga quieto cuando no pulsamos ninguna tecla y camine cuando pulsemos la tecla \_LEFT o \_RIGHT sería:

```
PROCESS protagonista ( )
PRIVATE
    int graph_ini = 0;
    int graph_fin = 9;
BEGIN
    ...
    LOOP
        animar ( graph_ini , graph_fin );
        IF ( key ( _LEFT ) )
            graph_ini = 10;
            graph_fin = 25;
            x = x - 8;
        ELSIF ( key ( _RIGHT ) )
            graph_ini = 10;
            graph_fin = 25;
            x = x + 8;
        ELSE
            graph_ini = 0;
            graph_fin = 9;
        END
    ...
    FRAME;
END
END
```