

## 12. Variables II

Ya sabes cómo declarar nuevas variables en tu videojuego, pero todavía tenemos muchísimo que aprender sobre ellas y sus aplicaciones.

Habrás observado que el comportamiento de tus enemigos no tiene demasiado sentido, ya que aparecen en las esquinas de la pantalla y su movimiento aleatorio no suele ser capaz de hacerlos llegar al centro de la misma. En este tema vamos a hacer uso de las variables para conseguir que los enemigos vayan siempre al centro de la pantalla.

### 12.1 Una variable tiene el significado que nosotros queramos darle

Todos los aspectos básicos de un proceso en Benu como pueden ser `graph`, `size` o `angle` tienen un significado concreto: Indican su número de gráfico dentro del FPG, su tamaño en tanto por ciento, su ángulo en milésimas de grado, etc.

En cambio las nuevas variables que creamos no tienen un significado por sí mismas sino que debemos ser nosotros, a través del código, los que les demos un significado concreto, por ejemplo haciendo que sucedan determinadas cosas cuando una de las variables que hemos declarado tome un determinado valor.

Recuerda que en el tema anterior hemos declarado la variable PRIVATE `vida` y nos ha servido para lograr que los enemigos sean capaces de resistir varios golpes antes de morir... Observa que mediante el lenguaje Benu cada vez nos resulta más fácil expresar comportamientos más y más complejos.

Ahora vamos a crear una nueva variable PRIVATE que llamaremos `dirección` (Sin acento), y nos servirá para indicar dónde aparecerá cada nuevo enemigo y hacia dónde debe moverse.

¿Por qué la dirección es PRIVATE y no GLOBAL? Pues porque cada enemigo podrá tener una dirección distinta, recuerda que el ámbito GLOBAL sólo nos sirve para declarar un valor que resulte único para todos los procesos del videojuego.

## 12.2 Ejemplo del significado que le podemos dar a una variable

Haciendo uso de la función rand(), vamos a hacer que la nueva variable PRIVATE direccion de nuestro enemigo tome un valor inicial aleatorio entre 1 y 4 antes de entrar en el bloque LOOP-END.

Consultando el valor obtenido, el enemigo decidirá su posición inicial y la dirección hacia la que se debe mover, según la siguiente tabla:

direccion == 1	El enemigo aparece en la esquina superior izquierda y debe moverse hacia la esquina inferior derecha.
direccion == 2	El enemigo aparece en la esquina superior derecha y debe moverse hacia la esquina inferior izquierda.
direccion == 3	El enemigo aparece en la esquina inferior derecha y debe moverse hacia la esquina superior izquierda.
direccion == 4	El enemigo aparece en la esquina inferior izquierda y debe moverse hacia la esquina superior derecha.

Para aplicar este concepto, el código del enemigo deberá cambiar ligeramente a la hora de decidir su posición inicial (Antes del bloque LOOP-END) así como deberá modificar su comportamiento al desplazarse por la pantalla (Dentro del bloque LOOP-END).

Un ejemplo de utilización de la variable dirección sería el siguiente:

```
PROCESS enemigo()
PRIVATE //Datos privados de cada proceso enemigo
    vida=10; //vida con un valor inicial de 10
    direccion;
BEGIN
    graph=50;
    direccion=rand(1,4); //Toma un valor de dirección aleatorio entre 1 y 4
    IF ( direccion == 1 ) //Condición: Si la dirección es 1
        x = 0; //Nos posicionamos arriba a la izquierda
        y = 0;
    ELSIF ( direccion == 2 ) //Alternativa: Si la dirección es 2
        x=800; //Nos posicionamos arriba a la derecha
        y=0;
    ELSIF ( direccion == 3 ) //Alternativa: Si la direccion es 3
        x=800; //Nos posicionamos abajo a la derecha
        y=600;
    ELSIF ( direccion == 4 ) //Alternativa: Si la dirección es 4
        x=0;
        y=600; //Nos posicionamos abajo a la izquierda
END
```

```
LOOP

    IF ( direccion == 1)                //Condición: Si la dirección es 1

        x = x + 4;                    //Nos desplazamos hacia abajo a la derecha

        y = y + 4;

    ELSIF ( direccion == 2)           //Alternativa: Si la dirección es 2

        x = x - 4;                    //Nos desplazamos hacia abajo a la izquierda

        y = y + 4;

    ELSIF ( direccion == 3)           //Alternativa: Si la dirección es 3

        x = x - 4;                    //Nos desplazamos hacia arriba a la izquierda

        y = y - 4;

    ELSIF ( direccion == 4)           //Alternativa: Si la dirección es 4

        x = x + 4;                    //Nos desplazamos hacia arriba a la derecha

        y = y - 4;

    END

...
FRAME;
END
END
```