

7. Procesos I

Ya hemos modificado el proceso principal de nuestro videojuego y hemos practicado con él las instrucciones fundamentales para modificar los aspectos de un proceso en Benu.

En el punto 3.2 de este temario comentamos que era posible **definir** el comportamiento de un proceso y a continuación poderlo **invocar** tantas veces como queramos.

Hasta ahora sólo hemos modificado el proceso principal de nuestro videojuego. Hemos **definido** cual era su comportamiento, y no hemos tenido necesidad de **invocarlo** porque el proceso principal se invoca siempre (Y una sola vez.) cuando ejecutamos el videojuego.

7.1 Definir un proceso en Benu

Vamos a definir un proceso. Sus instrucciones serán exactamente las mismas que habíamos incluido en el proceso principal de nuestro videojuego, la diferencia es que ahora podemos darle un nombre, el que nosotros queramos, por ejemplo **protagonista**. La sintaxis de un proceso en Benu es la siguiente:

```
PROCESS <Nombre del proceso> (  
BEGIN  
    <Asignaciones iniciales>  
    LOOP  
        <Asignaciones, incrementos y condiciones a ejecutar siempre>  
    FRAME;  
END  
END
```

Después del END del proceso principal (Al final del código), podemos añadir todas las definiciones de proceso que queramos, aunque por ahora sólo vamos a añadir una.

Observa que el nombre del proceso debe ir seguido de los caracteres "()", un paréntesis de apertura y uno de cierre que pronto veremos para qué sirven.

Entre las etiquetas BEGIN y END irán las instrucciones que determinan el comportamiento del proceso. Puedes cortar directamente todas las instrucciones entre el BEGIN – END que tenías en el proceso principal de tu videojuego y pegarlas aquí, a excepción de las instrucciones set_mode(); y load_fpg(); que sólo se deben ejecutar al principio para configurar el modo de vídeo y cargar el archivo FPG con los gráficos, respectivamente.

7.2 Invocar un proceso en Benu

Si has **definido** correctamente el proceso anterior, al compilar no deberías tener errores, pero al ejecutar es probable que no veas nada en pantalla. Esto se debe a que todavía no has **invocado** el proceso, y tu videojuego no hace nada más que establecer el modo de vídeo con `set_mode()`, cargar el archivo FPG con `load_fpg()`; y terminar.

Para invocar un proceso basta con escribir su nombre, es lo que haremos después de cargar el archivo FPG. Si nuestro proceso se llamaba protagonista, entonces la invocación tendría la siguiente sintaxis:

```
protagonista ( ) ;
```

Observa nuevamente que la invocación también incorpora los caracteres "`()`", y además un `;` al final, como las instrucciones de asignación e incremento que vimos en los temas anteriores.

Antes de seguir adelante, si no consigues resultados, recuerda que cada tema lleva asociado un videojuego donde puedes comprobar el código correcto. Si tienes algún problema lo más probable es que te hayas dejado alguna de las etiquetas END, que sirven para indicar el fin de los bloques de código que abren las etiquetas BEGIN, LOOP e IF, entre otras.

7.3 Ejercicio: Invocar un proceso disparo

Una vez hayas conseguido crear tu proceso Benu correctamente, vamos a hacer la prueba de crear otro proceso adicional que será el "disparo" de nuestro protagonista.

Lo llamaremos disparo (Por ejemplo.), y su definición se encontrará justo después del último END del proceso protagonista. Puedes darle las asignaciones iniciales que creas convenientes, por ahora no te preocupes por su posición inicial ya que más adelante aprenderemos a hacer que los disparos aparezcan en la posición exacta donde se encuentra el protagonista.

El disparo debe tener necesariamente un bloque LOOP - END donde se incluyan los incrementos necesarios para desplazarlo por la pantalla. Además, antes del END, deberemos tener una instrucción FRAME;, esto generalmente se cumplirá para todos los procesos dentro de su bloque LOOP - END, ya que de lo contrario no mostrarán nada en pantalla, e incluso haremos que el videojuego se quede bloqueado. Más adelante explicaremos la instrucción FRAME; en profundidad.

Para invocar el disparo añadiremos el siguiente condicional dentro del bloque LOOP - END del protagonista:

```
IF ( key( _space) )  
    disparo ( ) ;  
END
```