

39. PUBLIC

Ya hemos trabajado con prácticamente todas las utilidades principales de Bennu, pero nos queda un apartado muy especial que sin duda te abrirá las puertas a diseñar lo que quieras.

Observa que hasta ahora el ámbito de las variables PRIVATE no nos permitía acceder a ellas a través del identificador de proceso, y eso nos impide realizar tareas tan sencillas como determinar el daño que produce un determinado disparo al colisionar con un determinado personaje.

El ámbito LOCAL sí que lo permite, pero no lo hemos estudiado porque está muy mal implementado, tanto que modifica los identificadores de **todos** los procesos para alojar todas las variables LOCAL, independientemente de su tipo, incrementando innecesariamente la memoria ocupada.

Para evitar lo anterior tenemos el ámbito PUBLIC, que sólo modifica los identificadores del proceso que tiene declaradas las variables. Parece muy útil y sencillo, pero su tratamiento es especial, como veremos a continuación.

39.1 Sentencia DECLARE

Aunque podemos declarar las variables PUBLIC en el mismo punto en el que declarábamos las variables PRIVATE eso no nos permitirá poder acceder a ellas a través del identificador de proceso, ya que sigue un tipo de dato único para todos los procesos y sólo tiene en cuenta los aspectos básicos graph, x, y, etc.

Para solucionarlo, es necesario aplicar una sentencia DECLARE a los procesos que incorporan variables PUBLIC. La sentencia debe aplicarse **antes** de la propia declaración del proceso. En nuestro caso vamos a declarar unas variables PUBLIC en nuestro protagonista y por tanto antes de su declaración deberíamos añadir:

```
DECLARE PROCESS protagonista ( <Parámetros propios del protagonista> )
    PUBLIC
        int vida;
        int fuerza;
        int agilidad;
    END
END
```

Puedes incluir la sentencia anterior justo antes de declarar el proceso protagonista igual que hasta ahora. Para el protagonista será como si tuviese las mismas variables en ámbito PRIVATE sólo que ahora vamos a ver cómo el resto de procesos pueden acceder a ellas y modificarlas.

39.2 El Tipo de Dato Asociado al Proceso

La sentencia DECLARE no solamente asocia las nuevas variables al proceso sino que además modifica el identificador de proceso para que permita acceder a ellas.

Observa que hasta ahora el identificador de proceso se almacenaba en una variable de tipo entero (int) y nos permitía acceder a todos sus aspectos básicos como x, y, size, etc. Como si se tratase de un TYPE.

Ésto se seguirá manteniendo y podremos utilizar el identificador de proceso como siempre para acceder a esos campos, pero ahora tenemos además **un nuevo tipo de dato** que nos permitirá acceder a todos los campos PUBLIC de la misma manera.

El nuevo tipo de dato que genera DECLARE tiene el mismo nombre que el proceso, en nuestro caso protagonista pasa a ser también un tipo de dato, por lo que cualquier proceso puede tener una variable PRIVATE de tipo "protagonista" y con ella acceder a sus aspectos básicos y también a todas sus variables de ámbito PUBLIC.

Cuidado: Las variables de tipo "protagonista" y todas las variables de tipo de dato asociadas a un proceso con variables de ámbito PUBLIC no pueden modificarse antes de tener asociado un identificador de proceso válido.

¿Que quiere decir lo anterior?

Pues que antes de poder acceder a cualquiera de los campos de una variable PUBLIC es necesario que ésta almacene un identificador de proceso activo, por ejemplo, cualquier enemigo que queramos que acceda la vida de nuestro protagonista necesitará primero la variable PRIVATE de tipo "protagonista":

```
PRIVATE
    protagonista objetivo;
```

Y en segundo lugar, antes de acceder a cualquiera de los campos de objetivo, es necesario que la variable esté asociada al proceso protagonista y que éste esté vivo. Podríamos hacer:

```
IF ( objetivo = collision ( type protagonista ) )
    objetivo.vida = objetivo.vida - 1;
END
```

Una vez más, observa que antes de poder acceder a objetivo.vida comprobamos que hay colisión y que por tanto objetivo almacena un identificador de proceso válido.

Y por supuesto, además de collision (), también get_id (), father, y en general, todo lo que retorna un identificador de proceso del protagonista resulta ahora asociable al nuevo tipo de dato generado por DECLARE.