

## 44. Introducción a Benu 3D

Habrás observado que cambia mucho el chip cuando pasamos a Benu 3D. Son muchas las diferencias, como por ejemplo la carencia de utilidad de todas las variables LOCAL predefinidas, la ausencia de FPG y scroll, la ausencia del buffer habitual de pantalla en el que podíamos escribir cómodamente con write ( ), etc.

Para asentar un poco las bases hemos introducido sobre escenarios y otros modelos, en este tema vamos a realizar unos cuantos experimentos prácticos

### 44.1 Modificar el escenario en tiempo real

En el tema anterior ya explicamos cómo es posible aplicar funciones de rotación, escalado y traslación a cualquier identificador de modelo cargado con Benu 3D, incluidos los escenarios.

Ahora vamos a poner esto en práctica y vamos a utilizar funciones correspondientes para que en base a las teclas R, T, Y podamos rotar el escenario en cualquier eje, en base a las teclas F, G, H podamos escalarlo igualmente y en base a V, B, N podamos modificar su posición en el mundo. Observa que con estas teclas podremos gestionar incrementos positivos en cada eje, para hacer decrementos necesitaríamos obviamente otras tantas teclas.

Se trata de un comportamiento que queremos que perdure a lo largo de la ejecución, y por tanto debemos ir a un bloque LOOP-END, puesto que son los bloques de instrucciones en los que se repite el comportamiento. Observarás que en el fichero de código principal del videojuego de ejemplo tenemos un bloque LOOP-END en el que, como veremos más adelante, se actualiza la posición de la cámara. Será ahí donde introduzcamos el código correspondiente para la modificación del escenario.

Algunas de las funciones de transformación requieren que se les pase como parámetro una variable de tipo \_pos3d en la que se almacenen los valores de transformación a aplicar en cada eje. En primer lugar deberemos declarar unas variables PRIVATE a tales efectos:

```
PRIVATE
    _pos3d rotacion;
    _pos3d escala;
    _pos3d traslación;
```

Tendremos que dar unos valores por defecto a las anteriores variables, en el caso de la rotación asignaremos todos los campos a 0 para indicar que no queremos una rotación inicial, en el caso del escalado asignaremos todos los campos a 1 para indicar que queremos un tamaño normal y para la traslación asignaremos todos los campos a 0 para indicar que no queremos que el escenario aparezca desplazado inicialmente.

En cuanto a las instrucciones que tendremos dentro del bloque LOOP-END para gestionar las modificaciones, exponemos aquí el caso del escalado, puesto que todos los demás son análogos. Suponemos que la variable mapa almacena el identificador del escenario cargado:

```
LOOP
...
IF ( key ( _F ) )
    escalado.x = escala.x + 0.1;
ELSIF ( key ( _G ) )
    escalado.y = escala.y + 0.1;
ELSIF ( key ( _H ) )
    escalado.z = escala.z + 0.1;
END
M8E_MODELSCALE ( mapa , escala.x , escala.y , escala.z );
...
FRAME;
END
```

#### 44.2 Cargar modelos en la escena

Vamos a cargar unos cuantos modelos en la escena y vamos a aplicarles diferentes rotaciones, escalados y traslaciones. Puedes encontrar modelos en el repositorio de Benu, selecciona unos cuantos y no olvides tomar el fichero de imagen que llevan asociado, ya que es la textura y sin ella no podremos visualizarlos correctamente.

Ya hemos visto cómo proceder para la carga, rotación, escalado y desplazamiento, y puedes proceder de la misma forma que hasta ahora, pero nos faltará aplicar la textura.

Para aplicar la textura, una vez cargado el modelo con M8E\_LOADANIMODEL ( ) y almacenado su identificador en una variable, podemos hacer uso de las siguientes funciones:

```
int M8E_LOADTEXTURE ( string );
```

Recibe como parámetro la ruta del fichero de imagen que guarda la textura que queremos cargar. El fichero puede tener prácticamente cualquiera de los formatos de imagen habituales (.png, .bmp, .jpg, .gif, .tga, .pcx...).

La función retorna un identificador de la textura cargada, que podremos usar más adelante para aplicarla a cualquier modelo que hayamos cargado. Cada modelo tiene asociada una textura y podremos editarla, teniendo siempre en cuenta que el mapeo de los polígonos sobre la textura siempre se realizará de la misma forma.

M8E\_LOADTEXMODEL ( int , int );

Asocia al modelo cargado con M8E\_LOADANIMODEL ( ) cuyo identificador damos como primer parámetro la textura cargada con M8E\_LOADTEXTURE ( ) cuyo identificador damos como segundo parámetro.

Con todo esto sólo queda realizar unas cuantas pruebas que nos servirán para seleccionar los modelos para nuestro videojuego, como última recomendación no olvides que puedes separar el videojuego en módulos de la misma forma que lo hemos hecho hasta ahora. Conviene tener todo lo más ordenado posible, así que también sería recomendable que generes un directorio /map para los escenarios y otro directorio /model para los modelos.