

42. Mapas 3D

Bennu 3D es compatible con prácticamente todos los mapas de Quake III y otros juegos compatibles. Estos mapas tienen extensión .bsp y aunque podemos editarlos nosotros mismos con los programas **GtkRadiant** o **Quake Army Knife** (QuArK), el objetivo de este tutorial es partir de mapas ya hechos por otras personas, que podemos encontrar en internet en páginas web como <http://lvlworld.com>

42.1 Ficheros comprimidos

Los mapas son modelos 3D con un formato muy similar al de los personajes. Los formatos de modelo 3D ocupan relativamente poco, ya que únicamente almacenan información sobre coordenadas de cada vértice, vértices que forman cada polígono, textura asociada a cada polígono y poco más.

Aún así el caso de los mapas es especial, ya que tienen una gran cantidad de polígonos y sus ficheros suelen ocupar varios Mb. Es por eso que suelen venir guardados dentro de un fichero comprimido que Bennu descomprimirá antes de poder utilizar.

Las extensiones de fichero comprimido que almacenan mapas son .zip y .pk3, este último formato no es más que un .zip renombrado que podremos descomprimir normalmente con cualquier compresor.

Bennu 3D tiene una función que se encarga de descomprimir este tipo de ficheros al principio de la ejecución para que posteriormente podamos hacer uso del mapa .bsp que almacenan, además de todos los subdirectorios con las texturas, sonidos asociados y algún fichero más.

La función es:

```
int M8EE_ADDZIPFILE ( string );
```

Como parámetro recibe la ruta del fichero .zip o .pk3 que queremos descomprimir. Una vez descomprimido podemos hacer referencia a cualquiera de los ficheros que almacenaba como si estuviese descomprimido en el directorio del juego, aunque por lo general sólo haremos referencia en adelante el fichero .bsp con el mapa.

El valor de retorno nos indicará si la descompresión tuvo éxito (TRUE) o no (FALSE). Una posible causa de error a la hora de cargar un fichero comprimido sería que la ruta pasada como parámetro haga referencia a un fichero que no existe.

42.2 Mapas

Como ya hemos comentado anteriormente, el modelo 3D de un escenario no tiene un formato muy distinto del modelo de cualquier personaje. La principal diferencia es su alto número de polígonos y el tamaño de los mismos, y Benu nos ofrece funciones diferentes para cargar los mapas que resultan mucho más eficientes al trabajar con modelos grandes:

```
int M8EE_LOADMODELEX ( string , int );
```

Recibe como primer parámetro el nombre del fichero .bsp con el mapa, que puede hacer referencia al contenido de un fichero .zip o .pk3 cargado previamente con M8EE_ADDZIPFILE ().

En el segundo parámetro introduciremos un número comprendido entre 0 y 255 que indicará la calidad con la que se cargará el escenario. Recuerda que hemos dicho que los escenarios se cargan de una forma especial que optimiza el tiempo de carga, y esto puede lograrse a costa de reducir la fidelidad con la que se cargará el escenario. Salvo serias limitaciones de CPU podemos poner siempre 255 en el valor de la calidad.

El valor de retorno es importante, ya que se trata del identificador del modelo del mapa cargado y deberemos almacenarlo siempre dentro de una variable.

Gracias al identificador que nos retorna M8EE_LOADMODELEX () podremos aplicar diversas funciones para transformar el escenario y también podremos usar éste identificador como parámetro para otras funciones de cámara, respuesta a colisiones, etc.

La función M8EE_LOADMODELEX () también sirve para cargar modelos estáticos que no tienen por qué ser mapas, pero que por su tamaño nos resulta más eficiente cargar haciendo uso de esta función. En cambio tenemos otra función de carga de escenarios que sólo es aplicable a ficheros .bsp, y que está 100% optimizada para escenarios en este formato con respuesta a colisiones (Esto es algo que veremos más adelante).

```
Int M8EE_LOADQ3MAP ( string , int );
```

De la misma manera que la función anterior, el primer parámetro es el nombre del fichero .bsp a cargar como mapa y el valor de retorno es el identificador del mismo. La diferencia es que con esta función no es necesario pasar como parámetro un valor indicando la calidad. M8EE_LOADQ3MAP () siempre realiza la carga con la máxima calidad, la principal diferencia es que el segundo parámetro es una lista de colisiones, que como veremos más adelante, indica los modelos que deberán responder físicamente a las colisiones con suelos, paredes y techos del mapa. Por ahora podemos poner un 0 como valor de la lista de colisiones para indicar que no hay modelos que deban responder físicamente al mapa cargado con M8EE_LOADQ3MAP ().

42.3 Seleccionar mapas para nuestro videojuego

Haciendo uso del repositorio de recursos de Benu o bien buscando en <http://lvlworld.com> hazte con unos cuantos escenarios y prueba a cargarlos en tu videojuego usando las funciones vistas en este tema.

En algunos casos observarás que los mapas no cargan correctamente algunos suelos y paredes, esto se debe a que algunas de sus texturas forman parte de la instalación de Quake III Arena u otro juego que hace uso del mismo formato de mapas, y estas texturas no están incluidas en el fichero .zip o .pk3 que hemos descargado.

En esos casos lo más recomendable es buscar otro mapa que sí use texturas 100% propias, puesto que al faltar la textura asociada a un polígono se interpreta que no hay obstáculo y podemos atravesarlo, ya se trate de una pared o un suelo.

¿Por qué el comportamiento es así cuando falta una textura?

La transparencia (También la falta total de textura en este caso) nos servirá para tener algunas texturas a través de las cuales podamos ver. El caso de la falta de textura es demasiado extremo, pero piensa en el caso de que queramos una pared con agujeros, bastaría con aplicar un alpha en esos agujeros y podríamos ver a través de ellos e incluso atravesarlos.