Sonido en videojuegos

Nyquist en Audacity. Renderizado 2d elemental

1. La carpeta *motores* contiene varias muestras de un motor a distintas revoluciones. En este ejercicio utilizaremos el archivo *1121_on_Exh.ogg*. Utilizando efectos de Audacity, como el *Cambio de velocidad*, obtener varias muestras de motores a distintas revoluciones, y después mezclarlas para simular el sonido de un coche acelerando.

2. Utilizando el resto de muestras de la carpeta *motores* hacer una mezcla convincente para simular el sonido de un coche acelerando desde las mínimas a las máximas revoluciones proporcionadas en las muestras (puede utilizarse además el efecto de *Cambio de velocidad*).

3. (Opcional) La función `scale-srate` de Nyquist permite cambiar la velocidad de reproducción de una muestra (por ejemplo `(scale-srate *track* 2)` duplica la velocidad de reproducción). Podemos crear un efecto de aceleración reproduciendo sucesivamente un loop de motor incrementando paulatinamente su velocidad de reproducción. Implementar esta idea desde el prompt de Nyquist en Audacity con la muestra del primer ejercicio.

   Para ello, se puede implementar una función recursiva que tome como argumento el incremento de velocidad para el sample (1 lo deja intacto, 2 duplica la velocidad), el tope de incremento y el valor del incremento. Consultar la documentación de Nyquist en el enlace `https://www.cs.cmu.edu/~rbd/doc/nyquist/`

4. En este ejercicio vamos a hacer una implementación básica de renderizado de audio 2D apoyándonos en OpenAL. Comenzaremos creando un proyecto en C++ para integrar la librería OpenAL y reproducir algono de los sonidos de motor proporcionados. Para ello enlazamos las librerías de enlace estático (.lib), incluimos los archivos de cabeceras correspondientes (.h) y ubicamos las librerías de enlace dinámico (.dll).

   Después, cargar ese mismo sonido en dos *sources* balanceadas a izquierda y derecha, tal como se ha visto en clase, y realizar un renderizado de audio (en consola) que permita situar el sonido en el plano 2D con las teclas habituales `asdw`, mostrando en pantalla la posición des *source* en cada momento.

   A continuación, implementar este mismo programa con otra filosofía. Se utilizará un solo source `src` y modelaremos la posición con la ganancia (dependiente de la distancia) y el balance (dependiente de la posición izquierda-derecha). Utilizaremos sendas variables `ALfloat gain` y `ALfloat bal`, que denota: -1.0 balance a la izquierda, 0.0 centrado, 1.0 a la derecha. Para enviar este valor a OpenAL declararemos la posición de `src` `ALfloat pos[] = {bal, 0.0f, 0.0f};` y después, para actualizar el balance:

   ```
   pos[0] = bal;
   alSourcefv(src,  AL_POSITION, pos);
   ```

   Por último implementaremos un efecto de *aceleración* que subirá o bajará las revoluciones del motor con las teclas `rR`. Este efecto puede conseguirse alterando el parámetro `AL_PITCH` (ver la guía de referencia de OpenAL).