Informática Musical

Python, numpy

En los siguientes ejercicios utilizaremos Python 3 con las librerías *numPy* y *matplotlib*. Puede utilizarse alguno de los entornos de programación para Python instalados en los laboratorios si se tiene experiencia previa o bien un editor como Notepad++ y trabajar desde línea de comandos (consola de Anaconda).

1. Definir una variable global *SRATE* con la frecuencia de muestreo del proyecto. (por ejemplo 44100). Implementar una función *osc(frec,dur)* que devuelva una señal de la frecuencia *frec* Hz y duración *dur* segundos, con frecuencia de muestreo *SRATE*. Por ejemplo, para *frec=1, dur=1* debe devolver una array *numpy* de 44100 muestras con un ciclo de la función seno. Utilizar matplotlib para dibujar la señal y probar con distintas frecuencias y duraciones.

   Implementar funciones *saw*, *square*, *triangle* que obtengan señales con las formas indicadas (véase `https://en.wikipedia.org/wiki/Waveform`). Utilizar matplotlib para dibujar las señales.

2. Implementar una función *vol(factor,sample)* que multiplique la señal *sample* por el volumen *factor*.

3. Implementar una función *fadeOut(sample,t)* que haga un efecto *fadeout* con la señal *sample* desde el instante *t* hasta el final (caída lineal de volumen). Análogo con *fadeIn(sample,t)* haciendo *faceIn* desde el inicio hasta el instante *t*.

4. En este ejercicio vamos a implementar un oscilador sinusoidal con una filosofía diferente, generando la señal por bloques (*chunks*). El tamaño de dichos bloques vendrá determinado por una variable global *BUF_SIZE* (que podemos inicializar con 1024 por ejemplo).

   Para ello implementaremos una clase *Osc*. El método constructor definirá la frecuencia y otro método *next* devolverá el siguiente *chunk* (de tamaño *BUF_SIZE*) con las "siguientes" muestras de la señal. Concatenar varios *chunks* consecutivos y utilizar matplotlib para verificar que se obtiene la señal esperada.