Sonido en videojuegos

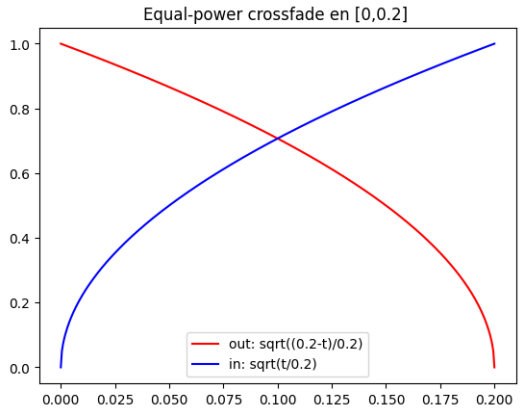
Unity Audio

1. **Ejercicio evaluable** (se subirá al CV el script pedido)

   En este ejercicio vamos a implementar un efecto disparo de rifle mediante la técnica de *slicing* vista en clase (script `Slice.cs`), pero añadiendo crossfades para solapar y mezclar las muestras. La carpeta `muestras/disparo` contiene:

   - *head_[1-2]*: fragmentos iniciales de disparo, con el el "golpe" de energía (transitorio)
   - *tail_[1-2]*: fragmentos de cola de disparo, que sonarán después del transitorio.
   - *casing_[1-4]*: muestras de casquillos contra el suelo.

   Para reproducir un disparo seleccionaremos aleatoriamente una muestra de cada tipo (*head, tail, casing*). Reproduciremos la muestras *head* y *tail* de manera consecutiva, pero con pequeño solapamiento temporal mezclándolas con envolventes de tipo *equal power crossfade*, según se muetra a continuación:

   En este gráfico se muestra la envolvente para fadeout en rojo y la de fadein en azul. En la leyenda se muestra la función que genera cada una de las curvas, para un intervalo temporal de 0.2 segundos (que se parametrizará en el script). En nuestro caso, aplicaremos la envolvente de fadeout en el clip `head` al final del clip de audio (en los últimos 0.2 segundos) y la envolvente de fadein al clip de `tail` al principio. Para conseguir la mezcla, el clip de `tail` comenzará a reproducirse 0.2 segundos antes de que termine `head`. Para aplicar las envolvente pueden obtenerse el clip de audio en un array de muestras con `AudioClip.GetData` y volver a incorporarlo con `AudioClip.SetData` (véase Unity documentation).

   Modificaremos el script `Slice` incluyendo una atributo público *overlapTime* para parametrizar el tiempo de solapamiento (0.2 en el ejemplo de arriba) y dos métodos `FadeIn((AudioClip clip)`, `FadeOut((AudioClip clip)` que apliquen la envolvente correspondiente a un clip de audio. EL método `Update` lanzará los clips (el segundo con el retardo correspondiente).

   Para completar el efecto, añadiremos al final un sonido aleatorio de casquillo.

   Opcional: la carpeta también contiene muestras *loop_[1-3]* con loop de disparo (ráfagas), que pueden activarse con otra tecla. Pueden además incluirse otras mejoras:

   - Para enriquecer el sonido pueden incluirse pequeñas variaciones aleatorias en el volumen y el pitch de cada muestra (haciendo las correcciones oportunas en la duración de los samples para sincronizarlos correctamente).
   - Variar el comienzo de las muestras *casing* en la cola de los disparos, tanto en el simple como en la ráfaga. En la ráfaga pueden incluirse pequeñas demoras temporales aleatorias entre disparos.
   - Incluir una reverb parametrizada para simular el recinto en que se hacen los disparos.

2. En este ejercicio vamos a recrear el ambiente de una ciudad mediante *superposición de capas*. La carpeta `muestras/ciudad` contiene varios tipos sonidos con los que realizaremos la mezcla:

   - *traffic_pad*: ambiente de tráfico.
   - *passing_[1-7]*: distintos vehículos pasando (se aproximan y se alejan).
   - *train_[1-2]*: trenes pasando.
   - *horn_[1-5]*: bocinas.
   - *siren*: paso de sirena.
   - *chatter_pad*: gente hablando.
   - *chatter_[1-17]*: fragmentos aislados de conversación.

   Vamos a realizar dos mezclas independientes, que luego sonarán juntas: una para los sonidos de vehículos y otra para las conversaciones. Cada una de ellas vendrá controlada por un parámetro de intensidad *Itraffic* y *Ichatter* respectivamente, ambos en el intervalo $[0, 1]$. Estos parámetros quedarán disponibles en el inspector de Unity para controlar fácilmente la mezcla.

   Para los vehículos utilizaremos las muestras del siguiente modo:

   - *traffic_pad*: sonido base de ambiente, que sonará de continuo. El parámetro *Itraffic* determinará el volumen.
   - *passing_[1-7]*: para $Itraffic < 0.2$ no suena ninguna de las muestras. Para $Itraffic \geq 0.2$ hay un doble efecto: se incrementa gradualmente el volumen y además se incrementa gradualmente la probabilidad de lanzamiento de cada pista (pueden sonar varias simultáneamente). Además, en cada reproducción se incluirá un ligera variación aleatoria de pitch (por ejemplo ± 0.05) para darle aun más variedad.
   - *train_[1-2]*: funciona igual que el anterior pero con menor probabilidad de reproducción.
   - *horn_[1-5]* + *siren*: en este caso, con $Itraffic < 0.5$ no suenan y partir de ese valor se incrementa la probabilidad de cada una de ellas. Además, el sonido se reproducirá en 3D en una posición aleatoria dentro de un círculo centrado en el listener.

   Para las conversaciones haremos algo similar:

   - *chatter_pad*: sonido base de ambiente, que suena de continuo. El parámetro *Ichatter* determinará el volumen.
   - *chatter_[1-17]*: para $Ichatter < 0.5$ no suena. A partir de ese valor funciona de modo similar a *passing_[1-7]*, incrementando volumen y probabilidad de reproducción. Se pueden ubicar también en posiciones 3D aleatorias.

   Una vez implementado lo anterior, podemos incorporar un sonido de motor a la cámara utilizando el ejercicio de la Hoja 5.

3. En este ejercicio haremos algo similar al anterior para recrear el sonido ambiente de un bosque con tormenta, utilizando las muestras de la carpeta La carpeta `muestras/bosque`. En este caso utilizaremos las muestras del siguiente modo:

   - *forest_loop*: sonido ambiente de base del bosque.
   - *bird_[1-5]*: sonidos de pájaros que se reproducirán en posiciones aleatorias del plano. Además se implementará un parámetro de intensidad, *Ibirds*, que controlará la probabilidad de lanzamiento de las muestras y la proximidad de las mismas al listener.

   Para implementar el sonido de tormenta utilizaremos las muestras de lluvia, viento y truenos, controladas con un parámetro de intensidad, *Istorm*, del siguiente modo:

   - *rain_[small,medium,big]* y *wind_[small,big]*: el parámetro controla la intensidad de la lluvia y el viento. Para 0 no suena ninguna de las muestras y a medida que se incrementa se introducen las muestras (de *small* a *big*) y el volumen de las mismas.

- *thunderstorm_[1-3]*: sonidos de trueno que suenan aleatoriamente con más probabilidad y volumen a medida que se incrementa *Istorm*. Esos sonidos, por su naturaleza, se pueden reproducir de modo parcialmente posicional (entre 2D y 3D).
- *drop_[1-9]*: sonidos de gotas de lluvia que pueden reproducirse en posiciones aleatorias del plano. El parámetro *Istorm* controlará el volumen el la densidad del sonido de gotas.