Sonido en videojuegos

Unity Audio 3

El ejercicio marcado con **Evaluable** debe entregarse a través del CV. Se subirá un único archivo `.zip` con los scripts pedidos con el nombre de los alumnos **NombreApellidos1-NombreApellidos2.zip**

1. Vamos a implementar un script de Unity para hacer un *dispersor* de sonidos extendiendo la idea vista en clase en el script `IntermitentSound.cs`. Este script permitirá cargar un conjunto de samples, que reproducirán con cierta aleatoriedad en el tiempo y con superposición entre ellos. Podremos configurar varios parámetros para controlar la *densidad sonora*:

   - `polifonía`: número de samples que pueden sonar simultáneamente. Este parámetro determina el número de audioSources que ha de crear el script (como en el script *Disparo5.cs*). Para reproducir un nuevo sample se busca un audioSource libre (`!isPlaying`); si no se encuentra ninguno libre (máxima polifonía alcanzada) no se reproduce el sample.
   - `minTime, maxTime`: intervalo temporal de lanzamiento de nuevos samples. Cada nuevo sample se lanza en un instante de tiempo aleatorio entre `minTime` y `maxTime`.
   - `minVol, maxVol`: cada sample se lanza con un volumen aleaotorio entre estos dos volúmenes.
   - `minPan, maxPan`: al lanzar un sample se asigna a `panStereo` un valor aleaotiro entre estos dos valores, de modo que suena más o menos lateralizado.
   - `pitchVar`: variación aleatoria de pitch; el pitch del sonido será un valor aleatorio dl intervalo `[1-pitchVar,1+pitchVar]`.

   Para desarrollarlo y probar, podemos utilizar las muestras *horn_[1-5]* de la carpeta *muestras/ciudad/*.

2. (**Evaluable**) En este ejercicio implementaremos un script de Unity para recrear el ambiente de una ciudad mediante *superposición de capas*, utilizando el dispersor del ejercicio anterior. 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.

   Todos los conjuntos de muestras anteriores utilizarán un dispersor (una instancia del script anterior) parametrizado adecuadamente, excepto *traffic_pad* que sonará de fondo en loop. 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 $[0, 1]$ que a su vez controlarán los parámetros de los dispersores implicados. 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 en el dispersor.
   - *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.

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.

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.

4. Los dispersores que hemos utilizado en los ejercicios previos simulan el posicionamiento en el plano mediante el volumne y el paneo. Podemos hacer un dispersor más sofisticado utilzando sonidos posicionales y aleatorizando dinámicamente su posición en el plano (o el espacio). Para ello tendremos que crear entidades, asociarles audioSources, etc.