Diseño sonoro en videojuegos

PID_00248064

Aniol Marín Atarés

Universitat Oberta de Catalunya

Índice

Introducción

El sonido suele ser el aspecto más fácilmente abandonado los videojuegos y de los audiovisuales en general, que suelen dar mucha más importancia a los aspectos visuales. A la hora de planificar la creación de un juego, es fácil aproximar los plazos necesarios para el motor de juego, diseño de niveles, modelado, texturizado, animación... pero del mismo modo también es fácil olvidarse de dar al diseño sonoro el espacio que requiere para ser desarrollado al cien por cien de sus posibilidades. El proceso de creación sonora, además, no es una disciplina separada: depende en gran medida de cómo sea el resto de producto para el que se diseña. Un compositor acostumbrado a la producción musical convencional, por ejemplo, puede tener problemas para crear los arreglos de una película, donde las imágenes marcan el ritmo y la estructura por encima de la lógica musical. El trabajo de audio en los videojuegos, a su vez, presenta algunas características de interactividad que lo diferencian en este sentido del resto de audiovisuales.

Un buen trabajo sonoro puede establecer una diferencia importante a la hora de determinar el éxito o el fracaso de nuestra aplicación. Estamos hablando de toda la información que llegará al usuario a través del canal acústico, mientras casi todo el resto lo harán en exclusiva por el visual. Y, en general, se trata de un aspecto más complejo de lo que de entrada puede parecer. Tanto es así que, en cuanto estemos trabajando en proyectos de suficiente envergadura, es necesario confiar todo el diseño sonoro a un equipo o especialista. En este módulo trabajaremos en cierta profundidad el diseño sonoro para videojuegos con el único objetivo de dar las herramientas necesarias para hacer una banda sonora simple en pequeñas producciones, dando algunas pinceladas de técnicas más avanzadas para quien esté interesado. En ningún caso se requieren conocimientos previos fuera de Unity, aunque los estudiantes que ya tengan algunas nociones de producción sonora apreciarán mucho mejor algunos de los contenidos.

Cuando pensamos en la banda sonora de un juego, solemos pensar en la música. En realidad, sin embargo, la «banda sonora» se refiere a todos los sonidos presentes en el producto (el término proviene del cine clásico, donde el sonido se almacenaba en una banda óptica o magnética del filme, paralela a las imágenes). Estos sonidos se pueden analizar por categorías separadas, que se relacionan entre ellas para formar un auténtico ecosistema, con sus relaciones internas e interacciones externas. La totalidad de los sonidos constituyen pues una esfera mucho más compleja, que bien trabajada debe servir para crear atmósfera, dar feedback al jugador y proporcionar una coherencia inmersiva a la experiencia de juego.

El sonido tiene dos funciones muy importantes:

1) Comunicativa: nos sirve para introducir sea diálogos que informaciones que pueden estar dentro o fuera de la pantalla. Por ejemplo, pueden anticipar los pasos de un personaje que se acerca y aún no hemos visto, o bien introducir un sonido de alarma que avisa al jugador de un peligro o situación crítica.

2) Emotiva: modula la percepción del usuario. Por ejemplo, un sonido tétrico y creciente puede anticipar las malas intenciones del personaje que se acerca, o bien una música divertida puede predisponer a pensar que, en cambio, el mismo personaje será bienvenido e inofensivo.

Algunos de los sonidos que hemos comentado son claramente parte de lo que se podría considerar «real», como los pasos de un personaje, mientras que otros podrían considerar «artificiales» desde el punto de vista interno del juego, como por ejemplo la música. Aunque se pueden clasificar desde estos puntos de vista (sonido diegético, sonido objetivo, sonido subjetivo,) a nosotros nos interesa más clasificarlos según la función que desempeñan en nuestro juego.

En general podemos dividir las esferas del ecosistema sonoro en 4 grandes bloques:

- **Efectos sonoros**. Constituyen en los sonidos cortos que asociamos directamente a objetos presentes en la escena o en efectos subjetivos que incorporamos a la misma. Por ejemplo el sonido de una botella de vidrio que cae al suelo y se rompe o un sonido de alarma que avisa al jugador de que se está acabando el tiempo en un nivel.

- **Sonido ambiente**. Consiste en un sonido relativamente neutro, fruto de la suma muchos elementos indefinidos, que asociamos a un ambiente en particular. Por ejemplo el rumor de coches y conversaciones indefinidas que nos rodean en una gran vía urbana. Especialmente en caso de que un juego o escena no tenga música, es importante tener en cuenta que todos los espacios pueden tener un sonido ambiente asociado, por silenciosos que parezcan de entrada.

- **Narración**. Consiste en el conjunto de voces inteligibles y relevantes, que tendrían que ser traducidas en caso de distribuir el producto en países con otro idioma, por lo tanto es importante mantenerlos separados del resto también desde un punto de vista práctico. Incluimos en esta categoría tanto los diálogos de los personajes en escena como las voces en off.

- **Música**. Se trata en general de sonidos pregenerados que no existen en el interior de la escena original (no diegéticos), pero que ayudan muchísimo a modular el estado de ánimo del jugador o espectador. En algunos casos puede reemplazar por completo al sonido ambiente, aunque también se

puede eliminar en una parte del juego o en la totalidad del mismo si se hace un buen trabajo en las otras esferas sonoras.

Para hacer un buen diseño sonoro, en especial en los entornos 3D, hay que tener en cuenta estas cuatro dimensiones por separado, de modo que funcionen independientemente. Durante la fase de desarrollo sería interesante, por ejemplo, testear el juego entero teniendo activa sólo la capa de sonido ambiente, para ver si realmente tiene una coherencia interna y ayuda a mejorar la experiencia del usuario con respecto al uso del juego sin audio. Paralelamente, claro está, es necesario que todas las dimensiones del audio interactúen entre ellas correctamente, creando un conjunto uniforme.

A la hora de crearlos, en general, estos bloques también se pueden trabajar por separado. Podemos por ejemplo comenzar desarrollando una música y luego continuar por las narraciones, el sonido ambiente, y acabar con la parte de los efectos sonoros. Pero comenzar con una planificación global de los bloques sonoros es siempre recomendable para facilitar el trabajo manteniendo una coherencia global. A poder ser, esta planificación se debe hacer en las primeras fases de desarrollo del juego, aunque el resto de diseño se puede hacer en simultáneo o más hacia el final. Hay que tener en cuenta que la elección de sonidos puede afectar aspectos del juego como el modelado o la física del juego y, viceversa, el diseño sonoro final dependerá también de la elección estilísticas del resto del proyecto. Desarrollar los bloques sonoros al final de todo el desarrollo de la aplicación y hacerlo estrictamente por capas, una a una, puede traer problemas de congruencia y limitación. En este módulo lo haremos así simplemente a modo didáctico, para ir introduciendo los aspectos, pero en las producciones reales se debería evitar este enfoque y empezar por una planificación metódica.

Comenzaremos pues con un ejemplo sencillo, diseñando los efectos sonoros en un juego 2D simple preexistente, que se ha desarrollado por completo sin un diseño sonoro previo.

Objetivos

Al finalizar este módulo se habrán logrado los objetivos siguientes:

1. Concienciarse de la complejidad e importancia de un buen diseño sonoro durante el desarrollo de videojuegos.

2. Aprender las bases de un diseño sonoro eficiente.

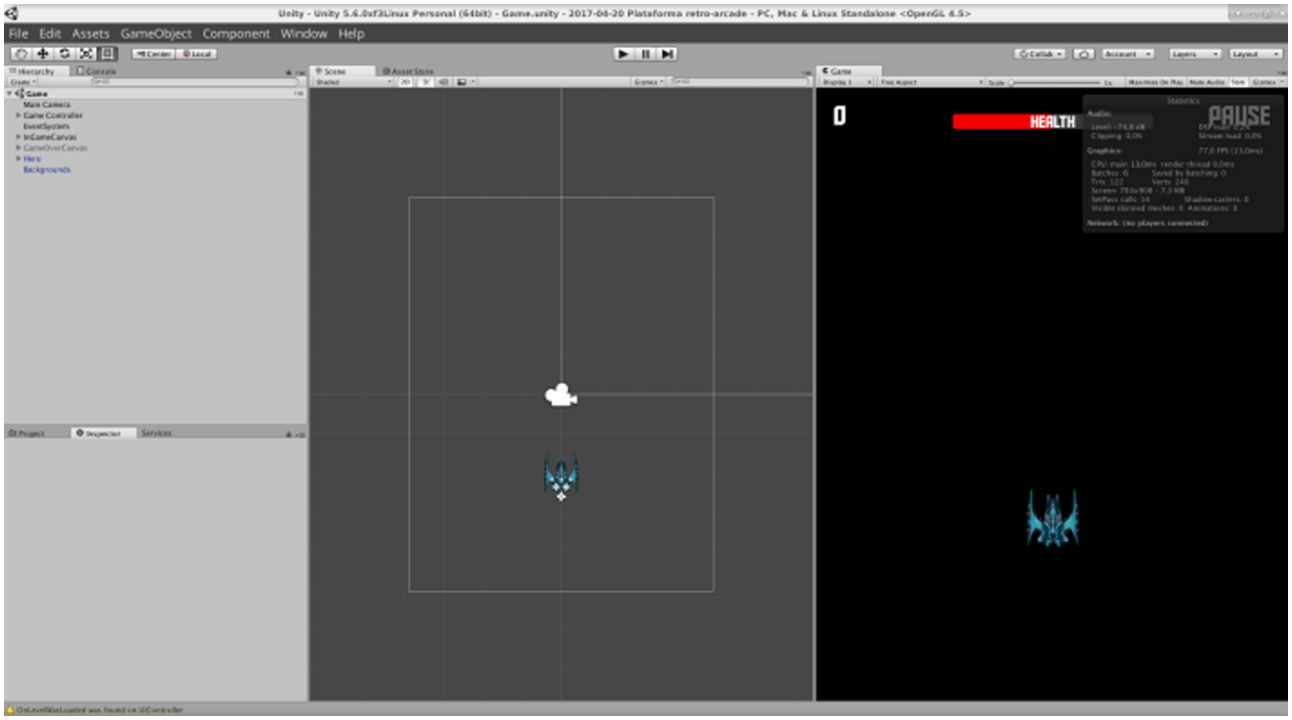
3. Conocer una batería de herramientas básicas para generar autónomamente el diseño sonoro completo de pequeñas producciones.

4. Optimizar el material sonoro de origen para su integración en el motor de juego de Unity.

1. Efectos sonoros en un juego 2D

Los efectos sonoros pueden ser uno de los bloques sonoros más «objetivos» y fáciles de planificar. Un buen trabajo de planificación incluye analizar detalladamente todas las situaciones de juego y detectando donde sería importante o útil que un *trigger* o animación llamara un efecto sonoro. Los más obvios son interacciones tales como pasos, explosiones, clic de ratón en los menús, etc... Otros son menos evidentes, como chirridos de objetos irrelevantes en un espacio ventoso, o atmósferas sonoras que anticipan la importancia un giro en el guión.

Como ejemplo de trabajo importaremos a Unity el paquete del Asset Store *Space Shooter Mobile* (gratuito) y que ya incluye un proyecto desarrollado sin sonido.



Conocemos ya la manera de importar los sonidos en nuestro proyecto. De entrada pues tendríamos suficiente con buscar un banco de sonidos que contenga los que nos interesan, siguiendo algunos criterios de los que detallaremos más adelante. De todos modos, en este caso de trabajo, vamos a crear los sonidos desde cero.

La elección de los sonidos se basa en primer lugar en un criterio estético. El juego tendrá diferentes modulaciones emotivas si utilizamos por ejemplo sonidos realistas, estridentes, divertidos o tétricos. En general será necesario que

este criterio estético vaya de acuerdo con el resto de elección estéticas, como las animaciones y las infografías. En el ejemplo, intencionadamente, utilizaremos el diseño sonoro para modular el juego original y darle un aire más cercano a los primeros Arcade de los años 70 como *Space Invaders*.

Los primeros videojuegos se crearon en una época en la que las posibilidades de las plataformas para crear sonidos eran muy limitadas, tanto en los Arcades como en los ordenadores personales (Commodore 64, Spectrum,...) y hasta en las consolas de primeras generaciones (Atari, NES,...). Hasta el 1987, se solía sintetizar el sonido directamente a través de un chip dedicado; a partir de la introducción del primer tracker para Amiga se empiezan a generalizar las composiciones por capas y con síntesis FM, que permiten mezclas por hardware más complejas en tiempo real. Sin embargo las composiciones hechas con trackers siguen siendo limitadas técnicamente y también tienen sonoridades características: los juegos de la GameBoy, por ejemplo, creaban todos los sonidos sólo a partir de la síntesis de cuatro notas simultáneas, dos de las cuales están limitadas a ondas cuadradas y una tercera a sólo ruido blanco o marrón. A pesar de las carencias técnicas, o precisamente a causa de estas, la creatividad de los músicos y diseñadores sonoros fue bastante prolífica, creando como consecuencia un género musical propio, llamado chiptune, y que ha influenciado enormemente los videojuegos posteriores y algunos géneros musicales en general (en especial la música electrónica, que ha hecho uso intensivo de los trackers). En general se trata de sonidos y músicas bastante reconocibles, basados en elementos muy simples pero a menudo con una gran creatividad o de gran calidad compositiva.

La mejor manera de entender el diseño sonoro retro es utilizando directamente las herramientas originales o bien clones de las mismas. De todas formas, a parte de los trackers (de edición clásicamente vertical), actualmente existe una gran variedad de DAWs y secuenciadores (de edición clásicamente horizontal) que suelen ser más completos y versátiles. El programa LMMS, bajo licencia, GPL, es una buena herramienta de partida ya que contiene varias recreaciones de chips históricos, integrados dentro de un secuenciador moderno. Para sonorizar nuestro juego, sin embargo, vamos a crear clips de sonido de tipo PCM con dos herramientas mucho más modernas, que nos permitirán generar rápidamente sonidos aleatorios y modularlos a voluntad. Hay muchos, nosotros en concreto utilizaremos SFXR, un simple generador de sonidos bajo licencia MIT; y *Din Is Noise*, un sintetizador microtonal bajo licencia GPL que nos servirá para ilustrar fácilmente algunos aspectos más avanzados. Los estudiantes que ya tengan experiencia con el uso de secuenciadores podrán encontrar una recreación de SFXR en LMMS, así como también *ZynAddSubFX*, un sintetiza-

dor con posibilidades casi ilimitadas. Cualquier otra técnica o software para producir sonidos que ya se conozca también son útiles para aprovechar lo que explicaremos.

En este módulo, como ya hemos dicho, explicaremos sólo el funcionamiento de la versión standalone de SFXR y de Din is Noise (DIN), de manera que todos los estudiantes puedan seguir fácilmente las explicaciones sin tener que conocer otros programas.

1.1. Análisis de los sonidos necesarios

El primer paso en todo diseño sonoro es prever de antemano qué sonidos vamos a necesitar. De momento, obviaremos otras esferas sonoras como las del sonido ambiente o la música. El proyecto *Space Shooter*, pues, contiene relativamente pocos sonidos para recrear:

- **Sonido de inicio**: este juego en concreto tiene una ventana espacial bastante larga entre que se pulsa Start y se comienza la partida. Así pues, sería recomendable diseñar una canción o un efecto sonoro complejo para llenar el vacío.

- **Movimiento de la nave protagonista**: nos acompañará durante toda la partida. Se puede crear un sonido constante para el movimiento constante, así como un segundo sonido para reforzar los desplazamientos a derecha e izquierda. De momento crearemos solo uno.

- **Movimientos de los enemigos**: ayudará a aumentar la tensión en los momentos que estos se acumulan en la pantalla. Se pueden crear dos, uno para las naves ligeras y otro para las naves pesadas.

- **Proyectiles**: aunque en este juego los proyectiles son disparados automáticamente, se les pueden asociar sonidos para aumentar la ambientación.

- **Daños menores**: cada vez que recibimos un impacto de un enemigo podemos reproducir un sonido para dar *feedback* al jugador.

- **Explosiones de los enemigos**: igual que en el caso de los movimientos, es recomendable hacer un por las naves ligeras y otro para las naves pesadas.

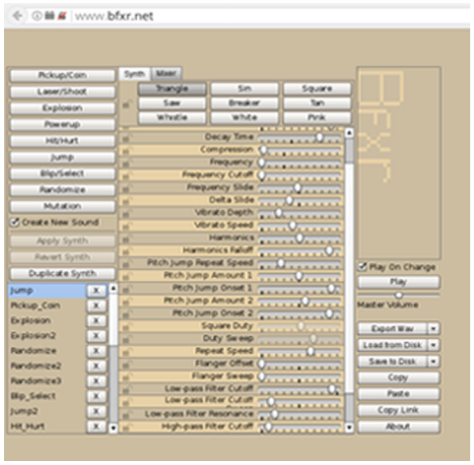
- **Explosión de la nave protagonista**: aparte de ser el último efecto sonoro, también puede tener la función de indicar que se acaba la partida.

- **Sonido de finalización**: al ser un juego que sólo puede acabar con la destrucción de la propia nave debe ser un sonido implícitamente de tipo *game over*, pero que al mismo tiempo debe ser válido si conseguimos un nuevo récord.

Comenzaremos pues diseñando la mayoría de estos sonidos con los programas que hemos comentado.

1.2. Diseño sonoro con SFXR

La mayoría de sonidos que hemos planificado son cortos, simples y estereotipados. Los podemos crear perfectamente con alguna de las herramientas especializadas que existen para ello, muestra de la ubicuidad de estos sonidos. La que utilizaremos nosotros, SFXR, la podemos descargar de la página web del proyecto (http://www.drpetter.se/project_sfxr.html), el programa no requiere de instalación y está disponible para Windows, Mac OS X, distribuciones Debian y NetSDB. De todos modos también se podrían utilizar fácilmente otras alternativas libres como por ejemplo BFXR (www.bfxr.net) o Chiptone (http://sfbgames.com/chiptone), o cualquier otra que nos permita obtener sonidos similares.



1.2.1. Sonido de inicio

Como ya hemos comentado, este sonido debería ser más bien largo y complejo. De momento, sin embargo, empezaremos por generar un sonido muy simple, similar a una señal horario radiofónico. Por defecto el programa ya nos genera sonidos relativamente complejos, así que ajustaremos los parámetros para simplificarlos al máximo:

- **Forma de onda**: SFXR nos deja elegir entre cuatro tipos de onda que luego podremos modificar: *Squarewave*, *Sawtooth*, *Sinewave* y *Noise*. Comenzaremos eligiendo Sinewave, que a nivel sonoro es la más simple ya que consiste en un solo armónico fundamental (de forma sinusoidal, como su nombre indica). Más adelante explicaremos las características del resto de ondas.

- ***Attack time***: controla la dureza del inicio del sonido. Si su valor es cero, el sonido empezará de golpe al máximo volumen. Si es ligeramente superior, el inicio será más suave, lo que nos puede interesar para rebajar el impacto

de sonidos demasiado duros como explosiones o guitarras distorsionadas. Un valor muy alto creará un efecto de fundido de entrada o fade in. En este caso nos interesa que el sonido empiece de golpe, así que lo dejaremos a cero.

- ***Sustain time***: controla la duración del sonido a volumen máximo. En el caso de SFXR es el único modo que tenemos de modificarla, a parte del *attack* y el *decay*. También es importante por ejemplo en caso de utilizar instrumentos midi como pianos, ya que un valor demasiado corto de *sustain* puede crear un vacío al tocar notas largas, mientras que un valor demasiado largo puede crear colas sonoras no deseadas. En este caso nos interesa un sonido muy largo, así que le daremos un valor alto o máximo.

- ***Sustain punch***: nos da un volumen extra en la primera parte del *sustain*, útil por ejemplo en explosiones. En el ejemplo lo dejaremos a cero.

- ***Decay time***: determina la velocidad en la que el sonido se atenuará después del tiempo de *sustain*, por tanto un valor cero lo hará acabar de golpe, mientras que un valor positivo tendrá como efecto un suavizado o *fade out*. Lo dejaremos a cero o casi para que termine abruptamente, aunque en *attack* y en *decay* a veces hay que vigilar, ya que si la onda sonora empieza o se corta en un punto diferente al eje x puede crear artefactos sonoros en los altavoces.

- ***Start Frequency***: determina la frecuencia base del sonido, que se mantendrá o se modificará según indiquemos en otros parámetros. Un valor más bajo dará como resultado un sonido más grave, mientras que un valor más elevado dará como resultado un sonido más agudo. En este caso nos interesa un valor medio, indiferentemente de qué frecuencia tenga exactamente.

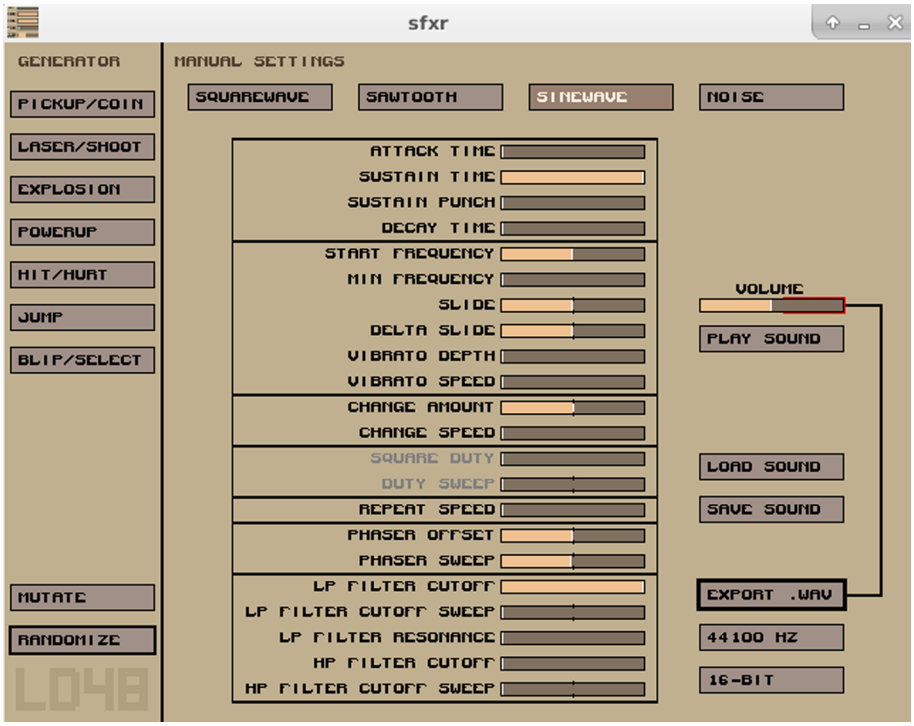
El resto de parámetros, de momento, no nos interesan. Para dejarlos inactivos, sin embargo, algunos deben tener valor 0, otros son neutros al 50%, y algunos se inactivan sólo con el valor máximo. Hay algunos parámetros que pueden tomar cualquier valor, ya que dependen de otro parámetro principal para surgir efecto. Así pues los agruparemos por valores:

- **Valor mínimo**: Min frequency, Vibrato depth, Vibrato speed, Change speed, Repeat speed, LP filter resonance, HP filter cutoff.

- **Valor 1/2**: Slide, Delta slide, Change amount, Phaser offset, Phaser sweep, LP filter cutoff sweep, HP filter cutoff sweep.

- **Valor máximo**: FP filter cutoff.

Nuestro programa pues deberá tener un aspecto similar al siguiente:



Si pulsamos *Play Sound* deberíamos oír un sonido largo y simple, similar al de los señales horarios radiofónicos. No es suficiente para el diseño sonoro final de un juego, claro, pero para ir entendiendo el programa de momento ya nos sirve.

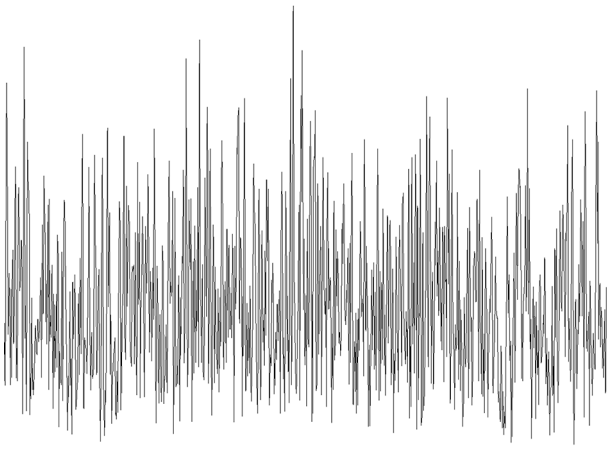
Para exportar el sonido sólo tenemos que elegir los parámetros preferidos (por norma general, 44,1KHz y 16-bit, o superior si se utiliza otro programa) y darle clic a *Export .wav*. Seguidamente sólo tendríamos que importar el sonido a Unity y asegurarnos de darle un volumen adecuado (hablaremos de los ajustes de volumen más adelante). Este sonido sólo se tendrá que ejecutar una vez, al inicio de la partida. También podemos hacer dos versiones del sonido, una corta y una larga, y encadenarlas para recrear los señales horarios o cuenta atrás. Para ellos podemos utilizar un programa de edición de sonido como Audition o bien crear un pequeño script, reproduciendo el primer sonido tres veces y el segundo una vez, típicamente a intervalos regulares de un segundo.

1.2.2. Sonido de la nave protagonista

El sonido de la nave, a diferencia del sonido anterior, nos acompañará durante toda la partida, casi a modo de sonido ambiente. Es importante pues que sea un sonido bastante neutro como para que no nos canse, aunque en este juego las partidas suelen ser muy rápidas y este factor no será tan importante como podría ser, por ejemplo, en el caso de los pasos de un personaje en una aventura que puede durar horas.

Para recrear el sonido del motor en SFXR partiendo del sonido anterior sólo necesitaremos ajustes en dos parámetros. En primer lugar, vamos a cambiar el tipo de sonido de *Sinewave* a *Noise*. Esto nos generará una onda sonora base completamente aleatoria, que como su nombre indica se percibe como ruido.

Forma de onda aleatoria del *white noise*.



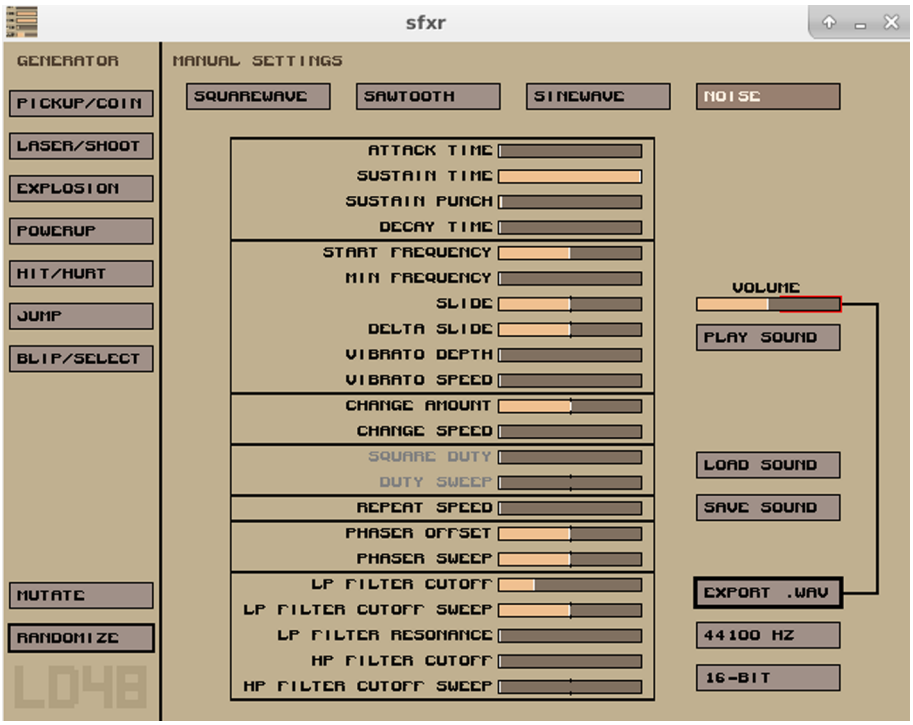
Fuente: By The original uploader was Bautsch at German Wikipedia - Transferred from de.wikipedia to Commons by Leyousing *CommonsHelper., Public Domain*, https://commons.wikimedia.org/w/index.php?curid=9935455.

El resultado directo, sin embargo, es lo que se conoce como *white noise* y es bastante agresivo. Tal y como hemos dicho nos interesa que sea un sonido más suave, lo que normalmente se conocen como *pink noise* o *brown noise* y que se utilizan también para simular la lluvia o el viento. SFXR no tiene otras opciones de *noise* pero sí que es posible recrearlas manualmente. Para ello, recortaremos las frecuencias agudas con un filtro de paso bajo. La herramienta adecuada en SFXR es el *LP Filter* (*Low Pass Filter*), que en nuestro caso podemos situar aproximadamente al 25%. También tenemos la opción de aplicar LPF más adelante, directamente en Unity.

Los «filtros de paso» recortan todas las frecuencias en uno de los extremos del espectro. Su nombre indica literalmente el tipo de frecuencias que dejan pasar. En este caso, los LPF dejan pasar sólo las frecuencias bajas. Los otros, llamados HPF (High Pass Filter) dejan pasar sólo las altas. A menudo hay que tener en cuenta este detalle ya que de lo contrario es fácil confundirse y pensar que el nombre se refiere a las frecuencias que descarta, haciéndonos perder tiempo a la hora de elegir entre uno u otro.

Hay que tener en cuenta que el filtro es afectado también por el parámetro *sweep*, que es neutro al 50%, y por la resonancia, que es neutra al 0%. En el caso del sonido que estamos creando es importante que estos y el resto de parámetros, en especial el ataque y la atenuación, sean neutros. De lo contrario tendremos problemas a la hora de hacer *loops*, tal y como comentaremos más adelante.

El programa debería aparecer aproximadamente así:



A la hora de importar el sonido a Unity nos interesará marcar la opción de repetir el sonido en *loop*. Si lo hemos hecho bien, oiremos un sonido continuo sin interrupciones. Esta técnica nos ayudará en muchos sentidos, ya que los archivos pequeños no sólo se pueden repetir infinitamente, sinó que además consumen menos recursos. De todos modos los *loops* a veces son difíciles de hacer bien hechos, ya que pequeñas diferencias entre el inicio y el final son fáciles de percibir, o incluso se puede notar el cambio si la posición de comienzo y de final de la onda no mantienen una continuidad (en este sentido, nos encontraremos mucha diferencias según el software o motor de juego que utilicemos). Si nos lo podemos permitir, poner en *loop* un clip de sonido con unos instantes de silencio final o bien basado en sonidos completamente aleatorios, como en este caso, minimiza las posibilidades de tener problemas.

1.2.3. Movimiento de los enemigos

Como ya hemos dicho, es importante mantener una cierta coherencia en el diseño sonoro. Si hemos elegido un tipo de sonido por el movimiento de nuestra nave, los movimientos de las naves enemigas no pueden ser completamente diferentes. Así pues también los basaremos en un ruido blanco modificado. A diferencia de la nave principal, aquí no tendremos que ponerlos en *loop*, y de hecho nos interesará que el sonido coincida con su movimiento en pantalla.

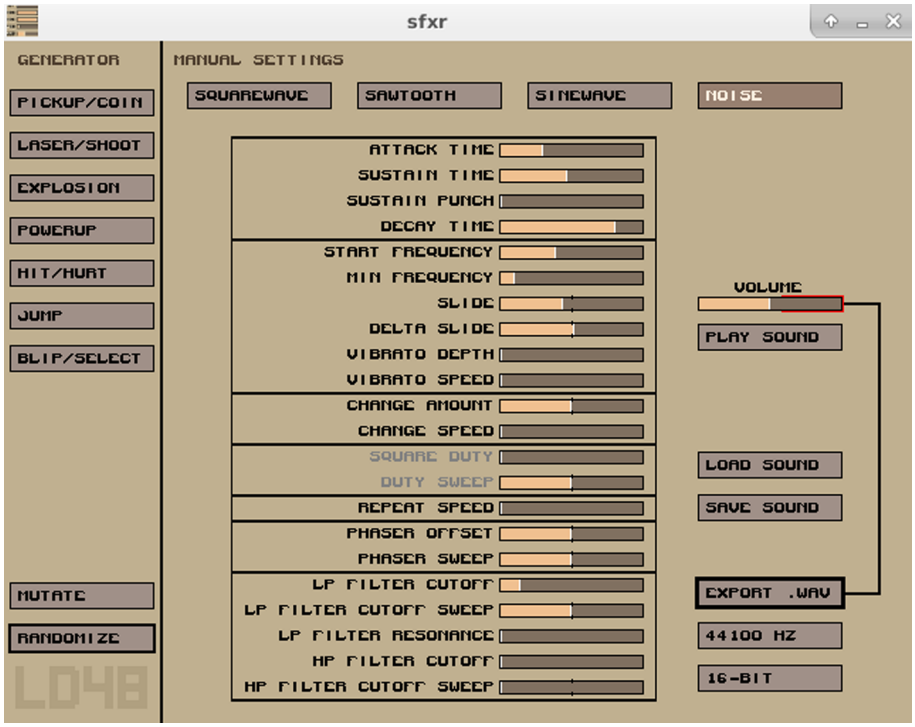
En este caso los enemigos se aproximan por un extremo y, si no son destruidos, se alejan por el otro extremo. Para ser más realistas podríamos utilizar los controles de volumen y de efecto Doppler de Unity mismo, pero de momento simularemos estos efectos dentro del mismo clip.

Para la nave ligera nos interesará hacer un sonido relativamente corto, ya que se mueve más rápidamente. Así pues bajaremos el valor de *sustain*. De todos modos, antes de ajustarlo, hay que tener en cuenta que no nos interesa que el sonido empiece y termine de golpe, como antes. Para modificar estos parámetros sólo tendremos que aumentar los valores de *Attack* y *Decay* respectivamente. Veremos que estos parámetros se añaden al valor de *Sustain*, así que es recomendable editarlos antes y corregir posteriormente la duración total ajustando este por último. La duración total del sonido debe ser aproximadamente el tiempo que el *prefab* estará en pantalla.

En este caso, además, las naves se aproximan y después se alejan. Para imitar un efecto Doppler nos interesará que la frecuencia base se modifique ligeramente. Para ello, reduciremos el valor del parámetro *Slide*. Un aumento, en cambio, haría subir la frecuencia, lo que nos podría ser útil en caso de querer simular aceleraciones o saltos. Alternativamente podemos jugar con un valor positivo de *slide* (velocidad constante) y un valor negativo de *delta slide* (aceleración), de modo que la frecuencia primero suba y luego baje.

También es interesante observar que, si el clip es suficientemente largo, el sonido pronto sería suficientemente grave como para generar subfrecuencias, que en este caso no nos interesan. Según cómo nos interesaría utilizar un HPF que cortara todas las frecuencias a partir de un cierto nivel. Como alternativa, sólo tendremos que dar un valor ligeramente superior al parámetro Min *frequence*, que nos evitará que el sonido siga bajando pasado un cierto nivel. Sin embargo, hay que tener en cuenta que en SFXR este parámetro nos interrumpirá el sonido, interfiriendo por tanto con el *sustain* o el *decay*.

Tal y como lo hemos descrito, el sonido generar debería tener unos parámetros similares a los siguientes:



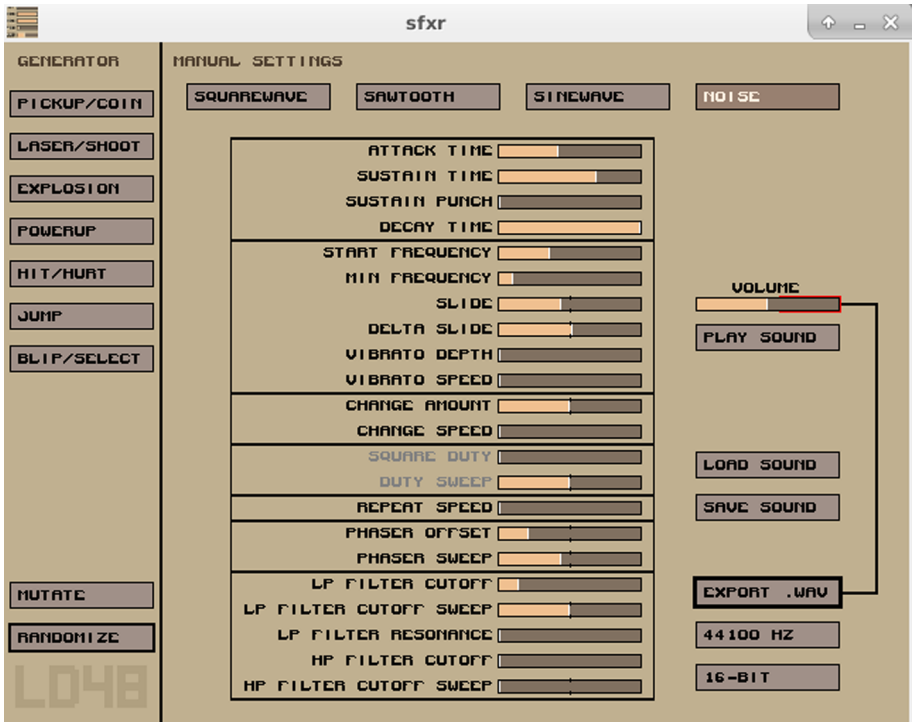
Hay que tener en cuenta que en este juego las naves aparecen en grupos, por lo que hay que prestar especial atención para que la suma de estos sonidos no resulte demasiado agresiva.

Para las naves pesadas podemos utilizar un sonido parecido. Basta que modifiquemos el sonido base para reflejar el mayor tamaño y la menor velocidad de las naves. También podríamos modificarlo a través del parámetro *pitch* directamente en Unity, con lo que nos ahorraríamos un sonido, pero a modo de ejemplo lo crearemos en SFXR.

Comenzaremos modificando otra vez el *Attack* y el *Decay* para hacer una entrada y salida más gradual, dado que la nave se mueve más lentamente. Adaptaremos de nuevo el *Sustain* para que el conjunto se adapte a la duración de la nave en pantalla. Después bajaremos ligeramente la *Start frequency* para acentuar la pesadez de las naves. Si hace falta ajustaremos los parámetros *Min frequency* y *Slide* para conseguir que la frecuencia no decrezca demasiado deprisa, dada la nueva duración del sonido.

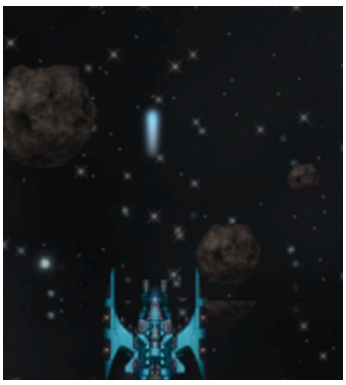
De todos modos el sonido resultante no acaba de ser óptimo. En este caso podemos aprovechar para modificarlo con un *phaser*. Los *phasers* consiguen modificaciones de timbre añadiendo una segunda copia del sonido con cambios temporales mínimos, con lo que las diferentes frecuencias sonoras interfieren constructiva y negativamente modificando el timbre. En este caso es suficiente modificar ligeramente los parámetros *Phaser offset* y *Phaser sweep* para conseguir un sonido más adecuado para la nave enemiga pesada.

El sonido propuesto sería aproximadamente este:



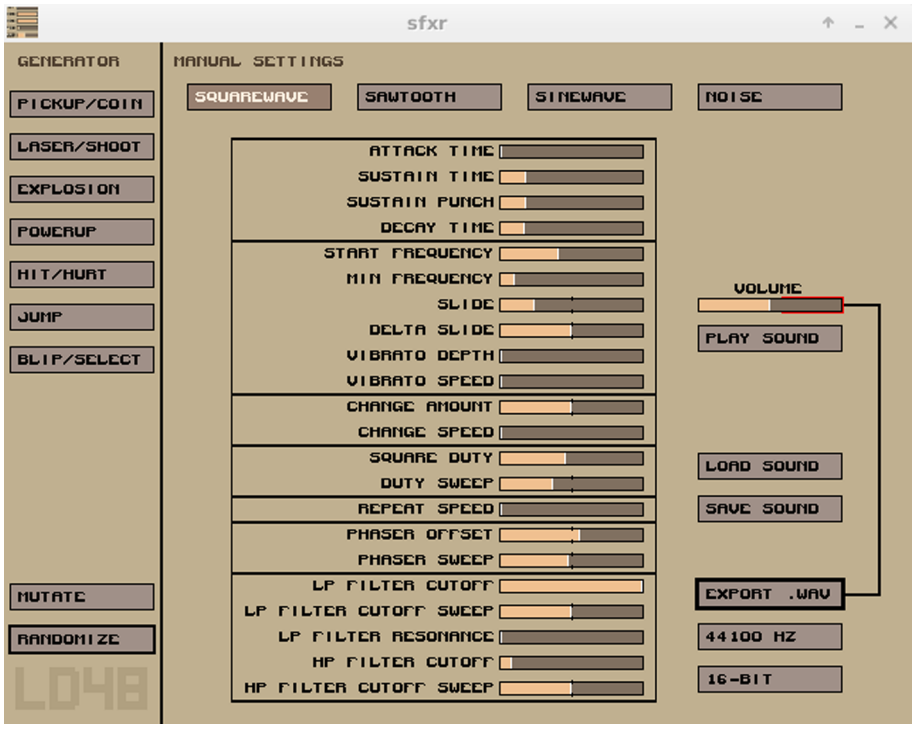
A la hora de importarlos en Unity hay que vigilar el efecto que producen, ya que en el juego de ejemplo todas estas naves aparecen al mismo tiempo y nos puede interesar que individualmente tengan un volumen muy bajo o bien modificar el sonido de manera que en los momentos en que se amontonan en pantalla la suma de sonidos no sea demasiado caótica.

1.2.4. Proyectiles



Los sonidos de armamento en los juegos retro suelen estar bastante estereotipados, así como también ocurre con muchos de los otros sonidos. Buen ejemplo de ello es que programas como el SFXR están pensados precisamente para generarlos aleatoriamente, sin mucho esfuerzo de nuestra parte. Para ello, sólo tenemos que pulsar el botón *Laser / Shoot* de la columna *Generator*. Cada vez que lo pulsemos nos generará un sonido nuevo, randomizando un grupo de parámetros específicos. Si el sonido generado se parece a lo que buscamos, lo podemos modificar manualmente o bien utilizando la herramienta *mutate*, que genera un nuevo sonido pero haciendo modificaciones graduales sobre la anterior, sin aleatorizar completamente los parámetros.

Ahora que ya hemos creado varios sonidos de nuestro juego debería ser fácil identificar uno que sea coherente con el resto de sonidos que hemos ido creando. Por ejemplo, algo parecido a este:



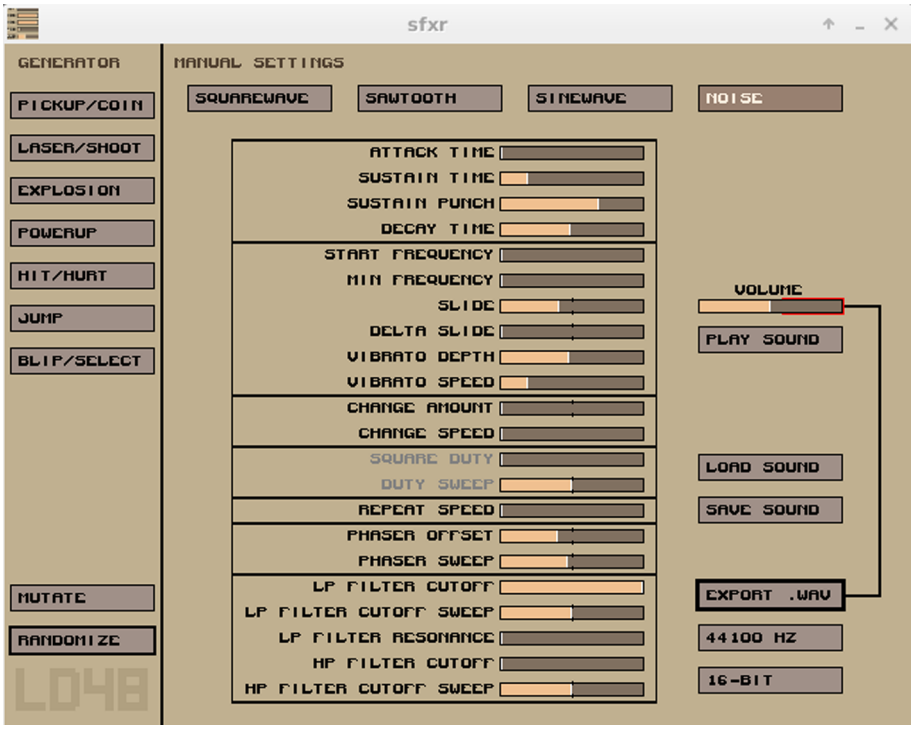
Como podemos ver este es el primer sonido de nuestro diseño sonoro donde utilizamos una onda cuadrada. Las ondas cuadradas se llaman así por la forma que tienen, es decir, en vez de describir un patrón sinusoidal, más gradual, saltan rápidamente de un extremo al otro. Técnicamente son más ricas en armónicos que las ondas sinusoidales, pero a diferencia de las ondas *noise* sí obedecen un patrón regular. También podríamos utilizar una onda en forma de diente de sierra (*sawtooth*), que también se denominan así por su forma. El sonido *Sawtooth* de SFXR, sin embargo, es ligeramente más agresivo, lo que en este caso no nos interesa tanto dado que los proyectiles se disparan continuamente, y por lo tanto nos arriesgaríamos a que se hicieran pesados durante la experiencia de juego. Necesitamos buscar pues un sonido muy corto y discreto.

Si nos dedicamos a analizar los sonidos dispares que el programa genera aleatoriamente podremos ver que, en general, se trata de sonidos muy cortos, que comienzan con una frecuencia relativamente alta y que baja rápidamente. Manualmente también podríamos experimentar con los parámetros de *vibrato*, *squareduty* y *repeatspeed*, que nos ayudarían a crear sonidos más agresivos. De todos modos para este juego es preferible dejarlos en valores bajos, ya que de lo contrario es fácil que acaben cansando.

1.2.5. Daños menores

Cuando nuestra nave recibe daños, aparte de perder vida, este juego simula una vibración en la pantalla para dar *feedback* al usuario. Así pues sería interesante generar un sonido que refuerce este *feedback* visual.

Para ello, ahora sí, intentaremos que nuestro sonido genere sufrecuencias, es decir, frecuencias extremadamente bajas que se perciben de modo especial. Estas frecuencias son similares a las que podríamos oír en un terremoto, y aunque en algunas plataformas se pueden conseguir solo a través de vibraciones, en caso de utilizar altavoces con subwoofer o auriculares de buena calidad se pueden conseguir a través del audio mismo. Aunque SFXR no nos permite explotar este recurso al máximo, nos podemos aproximar si hacemos partir nuestra explosión de una frecuencia muy baja (*Start frequence* cercano a 0) y nos aseguramos de que ninguno de los filtros de paso alto están activos (*HP Filter*, *Min frequency*). Por ejemplo, después de haber modificado un sonido de explosión aleatorio hemos obtenido la siguiente configuración:

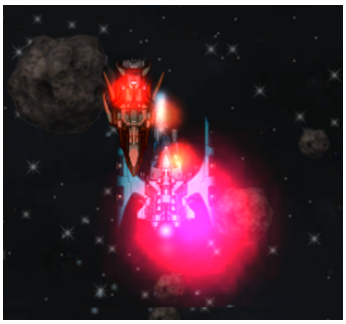


Como hemos dicho, este sonido acompaña la vibración de la pantalla y junto con el indicador superior deben dar *feedback* al usuario respecto al estado de salud de la nave. Hacerlo destacar con sus frecuencias bajas es también una manera de resaltarlo por encima de otros sonidos que se ejecutarán simultáneamente, ya que seguramente se activarán en los momentos de más caos y se perdería fácilmente si utilizara las mismas frecuencias que el resto.

Este sonido, bien hecho, entraría dentro de lo que se denominan sonidos no lineares o fuera de rango. Consisten en sonidos con frecuencias extremas, ya sean bajas como en este caso o altas como en el caso de silbidos agudos. Hay que tener en cuenta que en nuestro diseño sonoro retro los sonidos fuera de rango no tienen mucha importancia, pero sí pueden tener en juegos más inmersivos, en especial los que sabemos que se jugarán acompañados de unos

buenos altavoces o auriculares. Como veremos más adelante, jugar con estos sonidos cuando es posible crea efectos especialmente interesantes que no se deben menospreciar en absoluto.

1.2.6. Explosiones de los enemigos



Las otras explosiones también dan *feedback* al usuario, en este caso positivo ya que indican que el jugador se libra de un enemigo y suma puntos en el marcador. Nos interesa hacerlos destacar. Una opción con SFXR es generar una primera versión de explosión aleatoriamente y luego, si hace falta, hacerlos más presentes con la onda de diente de sierra, combinándolo con *vibrato*, *repeatspeed* y *phaser*. En general, sin embargo, debe ser un sonido corto, ya que se repite continuamente y, una vez cumplida su función informativa, se convierte más bien en un estorbo o distracción.

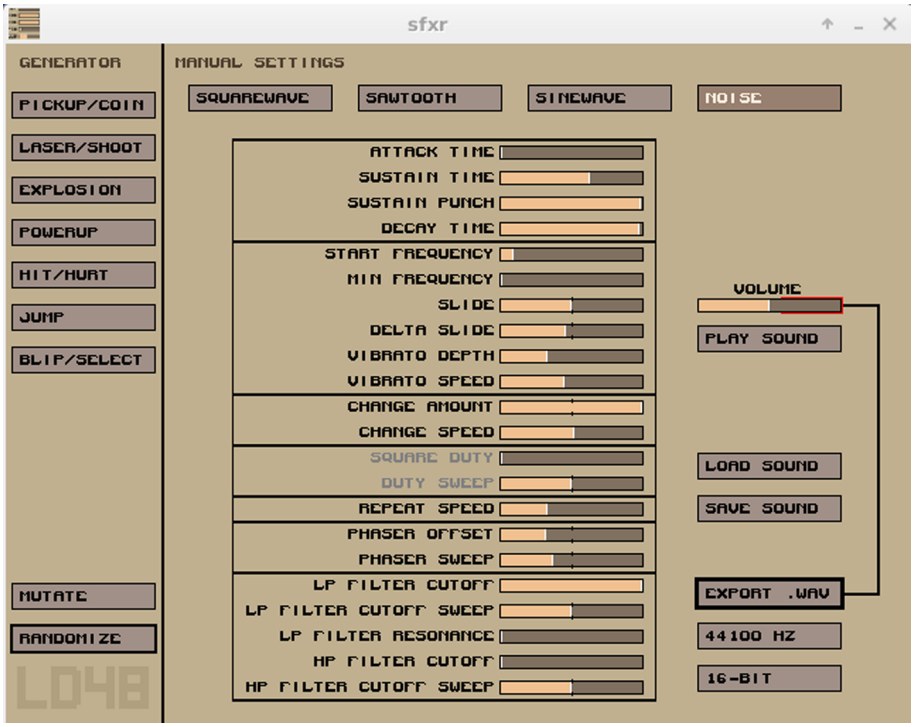
Como se podrá observar, la opción *repeatspeed* sólo es interesante aplicándola en cierta medida, ya que utilizada en exceso repite unidades muy cortas y crea un sonido uniforme, que seguramente nos aleja de la idea de explosión. Es una opción interesante en cambio para generar sonidos con timbres complejos, parecido a la síntesis FM muy utilizada en los *trackers*.

Podemos utilizar un solo sonido para los dos tipos de naves o bien crear dos de diferentes, con modificaciones similares a las que ya hemos hecho anteriormente.

1.2.7. Explosión de la nave protagonista

En el caso de recibir demasiados daños, la partida de *Space Shooter* terminará immediatamente. Aunque seguramente hubiera sido interesante haber desarrollado el juego para que se viera la explosión en pantalla, en este módulo sólo la estamos utilizando como ejemplo y por tanto no entraremos a cambiar su contenido. Así pues podemos aprovechar para utilizar este sonido como finalización.

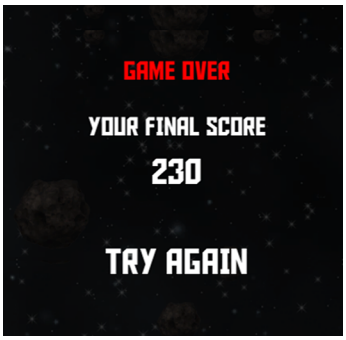
Cuando este sonido se activa sería interesante que todo el resto de efectos sonoros parasen de golpe. En general pues nos interesará que sea un sonido largo, contundente, y que deje claro que la partida se ha acabado. Centrándonos siempre en los sonidos de explosiones, podemos dar una punta de *delta slide*, *repeat speed* y una gran cantidad de *sustain punch*, de *vibrato* y de *change*:



Como ya hemos dicho, el *delta slide* modifica la frecuencia, de modo parecido a como lo hace el *slide*, pero haciendo un cambio acelerado en lugar de hacerlo lineal. En este caso lo hacemos hacia frecuencias bajas, ya que así contrastará con el *change* que aplicaremos después (aunque también podríamos contrastar con el *slide* que ya conocemos). El *repeat*, tal y como su nombre indica, reinicia el sonido independientemente de parámetros como *attack*, *sustain*, *decay* o los filtros de paso, pero afectando algunos de los otros parámetros que estamos utilizando. *Vibrato* sirve en cambio para hacer que el sonido aumente y disminuya de intensidad regularmente, hablaremos de él con más detalle en otra parte del módulo. *Change*, por su cuenta, nos hace saltar la frecuencia del sonido bruscamente, con el tiempo e intensidad que le indiquemos a través de sus dos parámetros. En este caso lo hacemos saltar hacia frecuencias altas tras el cambio en sentido contrario del *slide*, y justo antes de que el *repeat* reinicie el proceso. Se trata pues del sonido más complejo que hemos propuesto por el diseño sonoro de este juego, así que cambios mínimos en los parámetros pueden afectar mucho el sonido final.

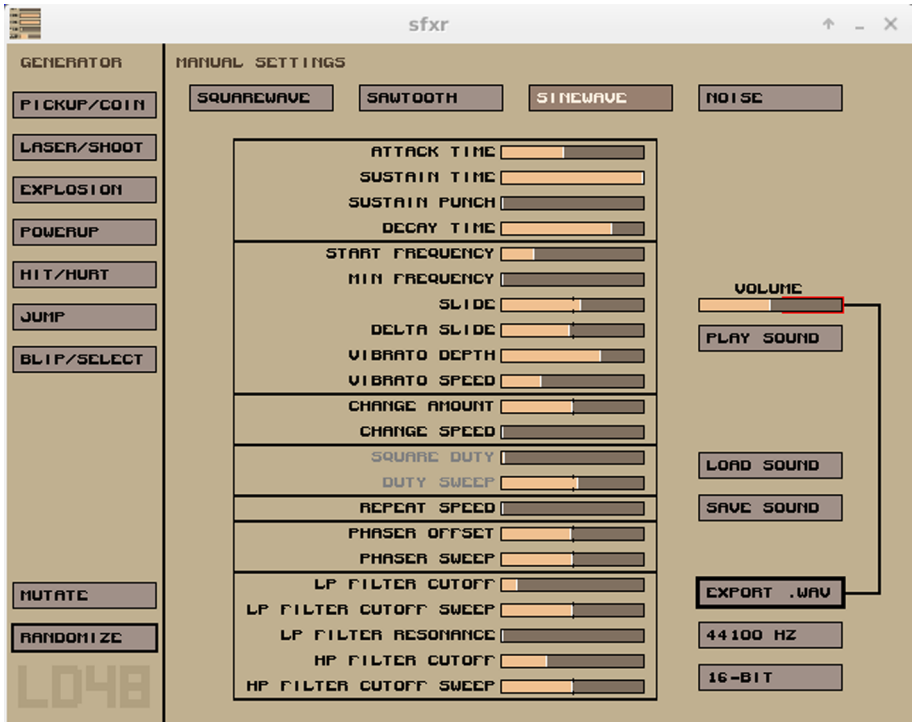
Con esta propuesta, más que una explosión realista, estamos creando un sonido informativo. Al mismo tiempo, conseguiremos un sonido emocional, con presencia, que el jugador puede asociar con el momento desagradable de «perder» la partida.

1.2.8. Sonido de finalización



Como ya hemos comentado, el sonido anterior ya se puede usar como finalización. En caso de optar por un sonido extra posterior, sin embargo, es interesante utilizar un sonido que contraste con el anterior y que, eventualmente, anime al jugador a disfrutar de la puntuación conseguida y a empezar una partida nueva.

Así pues, sería recomendable optar por ondas sinusoidales (u otras, pero dulces), largas, con entrada y salida suavizadas, y con frecuencias tendiendo al alza:



De todos modos, evidentemente, en este caso sería mucho más interesante disponer de una música o efecto complejo que acompañara la interfaz de usuario y acompañara a la nueva partida, tal y como ocurre con el sonido de inicio. SFXR es muy útil para ciertos efectos sonoros simples y estereotipados, pero para otros se queda corto. Para generar fácilmente otra variedad de sonidos, utilizaremos *Din is Noise*.

Reto 1:

Elige un proyecto sencillo de Unity y planifica la banda sonora del juego dos veces: una con sonidos retro como se ha comentado, y otra con sonidos realistas, que puedes encontrar en páginas como www.freesound.org. Puedes buscar música *creative commons* y añadir una canción *chiptune* o de cualquier otro género musical respectivamente. Compara las experiencias de usuario de uno y otro diseño sonoro.

1.3. Diseño sonoro con DIN

Como hemos podido ver, herramientas como SFXR son muy útiles para generar sonidos simples pero son muy limitadas a la hora de crear efectos sonoros más complejos. Para ello necesitamos pues una herramienta más potente.

A pesar de ser un sintetizador bastante heterodoxo, Din Is Noise nos servirá para ilustrar visualmente conceptos que serían difíciles de entender con otras herramientas más habituales. Los estudiantes que ya tengan conocimientos musicales pueden aplicar perfectamente los mismos conceptos sin necesidad de utilizarlo, de todos modos consideramos que se trata de una herramienta complementaria muy interesante para quien ya conozca otras.

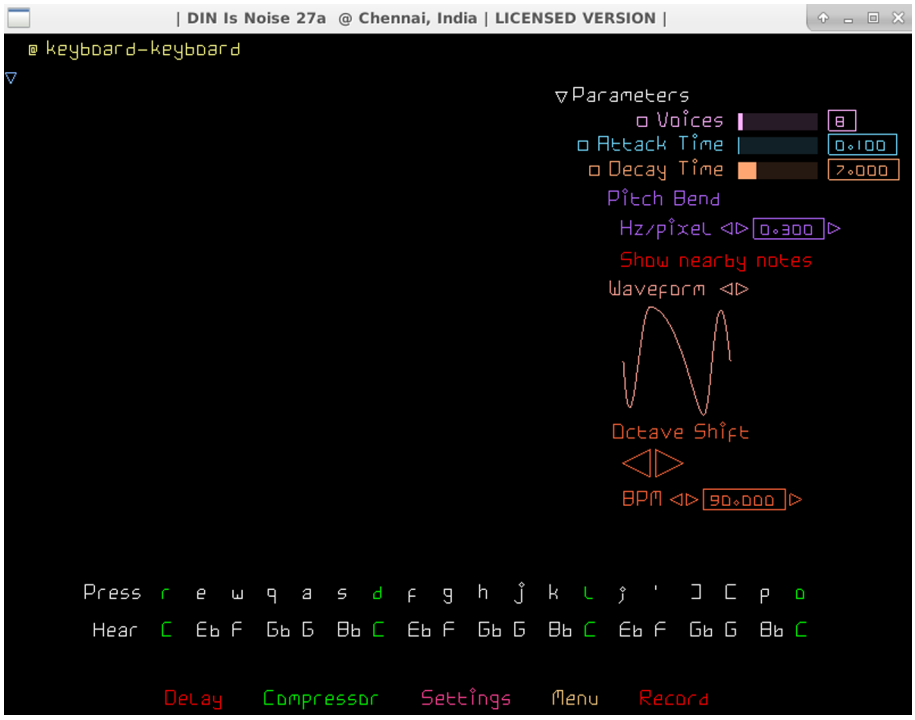
Din Is Noise es un programa con código fuente bajo licencia GPL, aunque dispone de binarios para Windows y Mac OS X con un periodo gratuito de evaluación de 30 días (http://dinisnoise.org/), aunque sin posibilidad de exportar los sonidos directamente. Se recomienda por lo tanto compilar el código fuente en entorno GNU / Linux, tal y como se ha hecho a la hora de desarrollar este módulo, para acceder gratuitamente al programa con la licencia completa.

Si se quiere utilizar los contenidos de este módulo para crear música, lo que es bastante lógico dados los contenidos, hay que tener siempre presente que las composiciones para videojuegos deben ser bastante fáciles de seguir para que todos las puedan apreciar sin distraerse de la experiencia de juego, y al mismo tiempo suficientemente complejas para que sean interesantes y no repetitivas. Es siempre recomendable buscar el equilibrio manteniendo la armonía y la melodía relativamente simples, potenciando mucho más en cambio su dimensión rítmica para que tenga un buen groove, ya que es la manera más fácil y efectiva de conseguir canciones que inviten al usuario a continuar jugando.

1.3.1. Sonido de inicio con escala blues

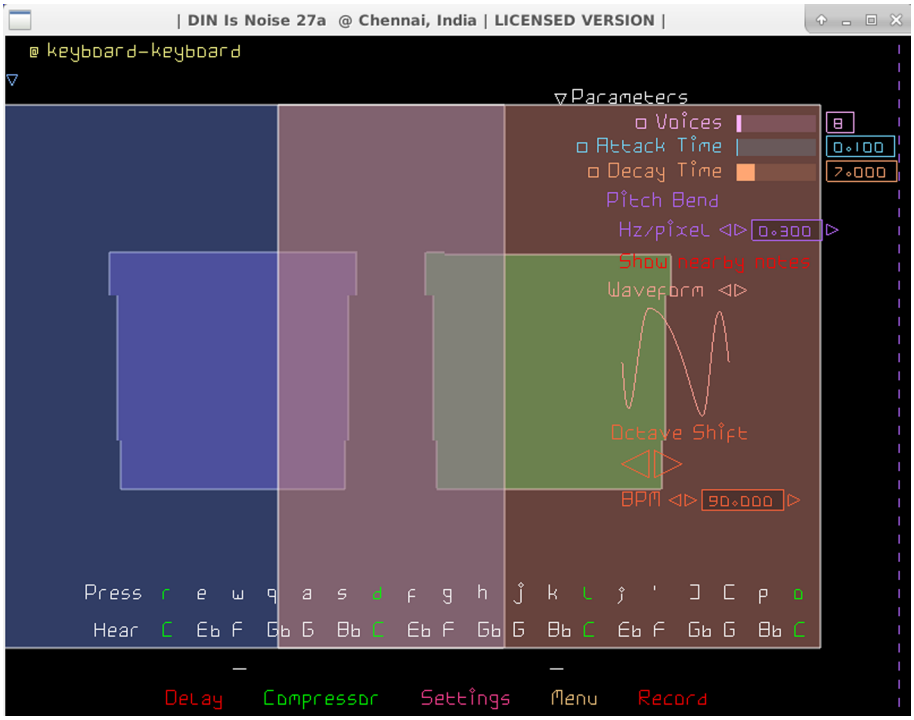
DIN ofrece tres modos de funcionamiento, o instrumentos, que pueden funcionar simultáneamente. Por defecto, mostrará el modo *keyboard-keyboard*, que ya es lo que nos interesa en primer lugar. Se puede acceder a otros instrumentos a través del menú o bien con la tecla 1 del teclado. Dada la complejidad del sintetizador y de la interfaz de usuario, es recomendable hacer *reset* a menudo a los parámetros de fábrica para no perder tiempo cambiando las configuraciones residuales de sesiones anteriores. Se puede hacer a través de *Settings* → *Reset to Factory Settings!* y reiniciando la aplicación.

La primera vez que ponemos en marcha el programa, o bien después de resetearlo, nos mostrará una pantalla como la siguiente, correspondiente al instrumento *keyboard-keyboard*:



Si disponemos de controladores midi los podemos configurar para utilizarlos con DIN, de todos modos en el módulo asumiremos que sólo disponemos del teclado qwerty estándar (es recomendable configurarlo en el *layout* inglés americano para la máxima comodidad a la hora de utilizarlo, aunque no es imprescindible). Podemos empezar a generar sonidos enseguida con las teclas «r», «e», «w», «q», «a», «s», «d», «f», «g», «h», «j», «k,«l»,«p»,«o», y si utilizamos el *layout* adecuado también«; »,« '»,«] »y« [»que llenan el vacío entre l y p. La razón de esta selección es puramente para hacer la interpretación más intuitiva, y no guarda relación con lo que trabajaremos en sí; evidentemente los estudiantes que dominen otros instrumentos musicales los pueden utilizar como alternativa.

Cuando pulsamos una o más teclas de las mencionadas, el programa nos mostrará cuadrados de colores y generará los sonidos correspondientes:

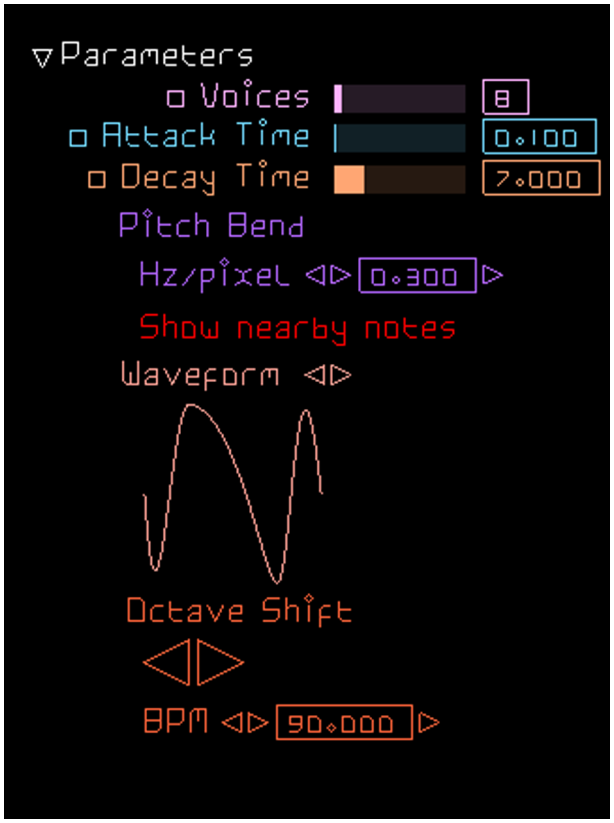


Podemos hacer tantas pruebas como queramos, o bien podemos empezar a grabar directamente en un archivo *wav* (recordemos que esta opción no está disponible en la versión de evaluación, por lo que habría que utilizar métodos alternativos de captura del sonido generado). Para grabar, simplemente hay que pulsar el botón *record*, abajo a la derecha. Una vez queramos parar la grabación nos preguntará dónde queremos guardar el archivo.

Por defecto en DIN podemos generar sonidos basados en la escala hexatónica de blues, que técnicamente consiste en una pentatónica menor con un # 5º grado extra. Más adelante explicaremos mejor el concepto de escala, ahora lo que es importante entender es que la escala utilizada determina la sonoridad de lo que produciremos, limitando al mismo tiempo las notas que podremos utilizar.

La escala predeterminada de DIN es interesante porque, aunque no nos permite recrear la mayoría de canciones que seguramente conocemos, nos permite crear melodías bastante buenas sin necesidad de entender lo que estamos haciendo. De todos modos, si se utilizan todas las notas disponibles, es fácil ver que se consigue una sonoridad blues muy característica, que en el caso del sonido de inicio para nuestro videojuego de ejemplo no es ideal. Seguramente nos interesará cambiar la escala para conseguir un sonido más adecuado.

De todas maneras, en primer lugar, comenzaremos modificando los parámetros más evidentes que aparecen en la interfaz:



- ***Voices***: determinará el volumen de cada nota introducida con el teclado qwerty, en función del número máximo de voces que pensamos utilizar. Un valor 1 dará el volumen máximo por nota, pero distorsionará el sonido si apretamos dos teclas a la vez o más. Es preferible dejarlo a 8 tal y como viene por defecto.

- ***Attack Time***: igual que en el SFXR determina el *attack* o *fade in*. A no ser que queramos un valor diferente, de momento no es necesario cambiarlo.

- ***Decay Time***: también lo conocemos, determina el *fade out*. El valor por defecto de decay en *DIN* es bastante largo. Seguramente nos interesará darle un valor más bajo, por ejemplo 1.

- ***Pitch Bend***: nos permite cambiar el pitch o afinación moviendo el ratón. A no ser que sepamos que queremos utilizarlo, es recomendable darle un valor 0 para desactivarlo.

- ***Waveform***: nos permite elegir una forma de onda predeterminada, aunque el sintetizador permite crear ondas mucho más complejas a partir de una curva de Bézier. Para el ejemplo aconsejamos elegir una onda cuadrada.

- ***OctaveShift***: nos permite cambiar la octava del teclado, es decir, asociar las mismas teclas a un sonido más grave o más agudo. Lo podemos modificar libremente según nos interese, aunque el valor por defecto ya nos sirve.

Una vez modificado los parámetros podemos crear una pequeña melodía para la sección de inicio del juego que hasta ahora tenía solo el tono de cuenta atrás. Por ejemplo, una simple secuencia como «a - s - d - f - k - j». Si no queremos editar ulteriormente el sonido, sólo necesitamos empezar a grabar justo antes, teclear las seis notas y parar la grabación una vez el último cuadrado haya desaparecido de la pantalla. El programa nos generará automáticamente un archivo .wav que podemos importar directamente a Unity.

1.3.2. Sonido final con escala cromática

Como ya hemos comentado, la escala musical que estemos utilizando determinará en gran medida la sonoridad de nuestro clip. Para el ejemplo anterior hemos utilizado una escala blues, aunque la secuencia de notas propuesta para el sonido de inicio intenta evitar la sonoridad blues utilizando sólo una parte de las mismas.

Para eliminar las limitaciones, sólo tenemos que indicar al programa que queremos utilizar más sonidos. El instrumento *keyboard-keyboard* nos permite introducir cualquiera de las 12 notas (semitonos) de la escala cromática. Para modificar la escala, sólo tenemos que ir a *Settings* y activar las notas en rojo para que aparezcan en verde:

Aprovecharemos también para comentar que las líneas superiores (*Key* y *nearest note*) nos permiten, respectivamente, cambiar la afinación de la nota base y cambiar la nota fundamental de la escala (altura). En general sólo nos interesará cambiar estos parámetros si queremos hacer composiciones musicales con intérpretes reales, ya que cada instrumento particular (pianos, trompetas, voces humanas...) tiene sus propias limitaciones que hacen que sea más adecuado, difícil, o incluso imposible utilizar una afinación o altura determinada. Para experimentar con las sonoridades de cada escala, no es necesario alterarlos.

A la hora de redactar estos materiales, la modificación de la escala a DIN limita automáticamente el número de teclas válidas para generar melodías. Para crear melodías con un rango más amplio habría que utilizar un controlador midi, aunque para entender el funcionamiento de las escalas no nos interesa ya que el controlador nos permite introducir todas las notas, sin las limitaciones artificiales que establecemos a través de la configuración de escala de DIN.

Con todas las notas activas, pues, vamos a crear un pequeño sonido para el final del juego. Recrearemos las archiconocidas notas de final de partida con la secuencia «h - g - f - d». También la podemos recrear, con un poco más de complejidad, pulsando las teclas de dos en dos: «h+l - g+k - f+j - d+h». Esto nos permite crear un sonido algo más interesante.

La *escala cromática* es la más "completa" de las escalas musicales occidentales, ya que por definición incluye todos los sonidos posibles en la misma. Como base para efectos sonoros de videojuegos ha sido muy utilizada, ya que es la que permitía recrear más sonidos con los recursos disponibles. Para simular un efecto Doppler, por ejemplo, se puede hacer una nota aguda y larga, y encadenando las notas descendientes hasta llegar a la nota larga final. También se ha utilizado mucho para sonidos estereotipados de saltos (cromática ascendente) o caídas (cromática descendente). De manera menos evidente, también se utilizan mucho cambios de tonalidad cromáticos para aumentar los efectos dramáticos de las composiciones musicales.

Para los estudiantes que estén interesados tanto en este como en muchos otros de los aspectos musicales que comentamos, las composiciones originales de Koji Kondo para la saga Super Mario Bros son un auténtico compendio de recursos sonoros generados sólo con los doce semitonos de la música occidental.

1.3.3. Sonidos musicales y de ambientación con otras escalas y recursos

Tal y como hemos comentado, la escala cromática es la que nos permite utilizar todas las notas de la música occidental, y por tanto de entrada es la que tiene más posibilidades. Debemos mencionar que estas doce notas no son más que proporciones concretas entre una frecuencia dada (en Hz) y el doble de la misma (octava), por ejemplo las frecuencias 220Hz, 440Hz y 880Hz suelen corresponder a la nota la, mientras que el resto de semitonos se consiguen multiplicando la frecuencia de la nota anterior por una doceava parte armónica de la octava, es decir, $f_n = f_{n-1} \times (\sqrt[12]{2})$. Si establecemos las frecuencias de otras maneras obtendremos sonidos diferentes a través de una misma escala.

Dentro del mismo sistema dodecafónico que comentamos, existen otros métodos para determinar las frecuencias relativas entre notas. Algunos son propuestas modernas, otras son soluciones que se han utilizado históricamente para conseguir que los instrumentos de afinación fija (como la guitarra) puedan tocar en cualquier tonalidad. Actualmente se suele utilizar lo que se llama temperamento igual, que es el que resuelve los doce semitonos de la escala distribuyéndolos armónicamente en la octava, creando por tanto proporciones iguales entre las frecuencias de intervalos iguales de todas las notas. De todos modos existen sistemas de afinación alternativos, como el temperamento justo o el temperamento mesotónico, que hacen que las relaciones entre notas particulares cambien y por tanto que cada escala tenga una sonoridad di-

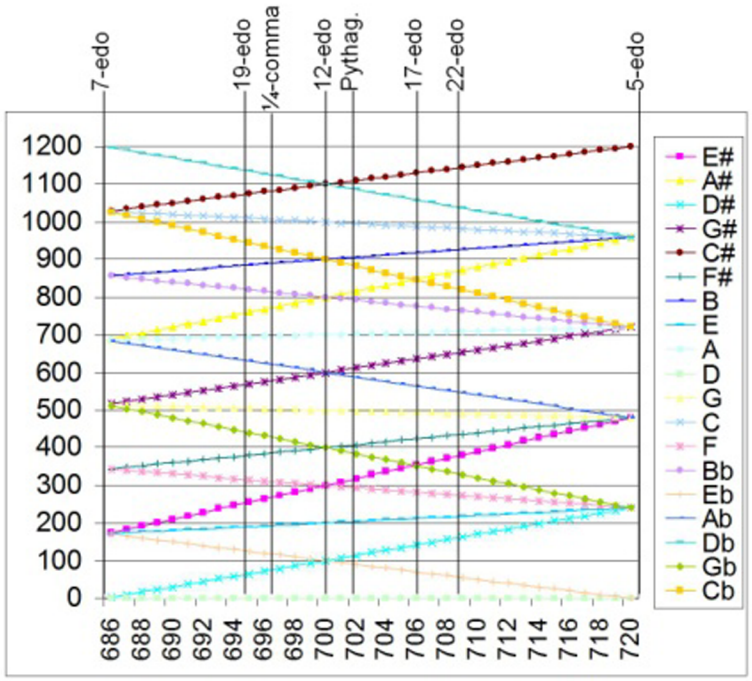
ferente. En algunos casos concretos, pueden sonar incluso mejor que en temperamento igual (como las quintas de la afinación pitagórica, o las terceras mayores de la afinación mesotónica), aunque las mismas afinaciones pueden sonar considerablemente peor en otras situaciones (como por ejemplo la escala de fa# en afinación mesotónica de cuarto de coma basada en do mayor). En general sólo nos interesará entender estos fenómenos para experimentar con sonoridades diferentes o para recrear fielmente sonoridades históricas.

A parte de los sistemas de afinación, las frecuencias exactas también dependerán de la frecuencia base elegida. Actualmente se utiliza sobre todo la afinación estándar, que hace coincidir los 440Hz con un la (y por tanto los 261,626Hz con un do, tal y como nos indica por defecto DIN). De todos modos, teóricamente, cualquier otra afinación es posible, como por ejemplo la llamada afinación científica que hace coincidir un do con 2^8Hz (256Hz), haciendo corresponder por tanto los 432Hz con un la. En general estos aspectos no tienen trascendencia a la hora de crear sonidos para videojuegos, a no ser que se trabaje en equipo con otros músicos o bien en proyectos muy concretos donde la interacción del usuario con la música tenga un peso muy importante.

Opciones del menú de DIN para cambiar la afinación utilizada

Existen otras escalas no occidentales, como el *Maqam Bayati* de la música árabe, que utilizan notas no presentes en los doce semitonos cromáticos que nos son habituales, y que son los únicos que muchos instrumentos y programas musicales que conocemos están preparados para ejecutar fácilmente. Algunas escalas por ejemplo utilizan notas que podríamos aproximar como cuartos de tono ($f_n = f_{n-1} \times (\sqrt[24]{2})$), aunque existen propuestas alternativas basadas en otras divisiones de la octava, en por ejemplo 53 partes iguales ($f_n = f_{n-1} \times (\sqrt[53]{2})$). El uso de estas escalas, otros temperamentos o, directamente, de sonidos que podrían considerarse «desafinados» desde un punto de vista occidental tiene efectos interesantes para el diseño sonoro. Más adelante, por ejemplo, experimentaremos algunos efectos similares con DIN. Por el momento, para experimentar, sólo necesitamos dar un valor ligeramente superior a cero en el parámetro *Pitch Blend* que hemos desactivado antes y tocar cualquier melodía con una mano mientras con la otra movemos el ratón a derecha e izquierda (aunque, técnicamente, esto modifica la afinación de las notas sólo después de que hayan empezado a sonar).

Diferentes sistemas de afinación presentados en un continuo de posibilidades. 12-edo corresponde al temperamento igual.



Fuente: By JimPlamondon - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4467295

Volviendo a las 12 notas que nos son más habituales, en teoría con esta escala cromática ya podemos conseguir cualquier sonoridad musica conocida. De todos modos nos será más fácil conseguir una sonoridad específica si nos dedicamos a «simplificar» la escala cromática, evitando completamente algunas de sus 12 notas o bien limitando su uso. Cuando en un fragmento utilizamos un conjunto específico de notas que presentan entre ellas relaciones reconocibles podemos hablar de escalas. Según el uso que hacemos de esta escala, también podemos hablar de arpegios y acordes, de los que hablaremos más adelante.

El objetivo de este módulo, como ya hemos ido repitiendo, no consiste en entrar en detalles de composición musical. Hay muchos conceptos que hay que dominar, ya sea por conocimiento que por intuición, para crear canciones. Introducirlos queda fuera de nuestro abaste en este módulo. Los estudiantes que no tengan experiencia musical previa y que quieran aprovechar al máximo los recursos que proponemos deberán consultar otras fuentes complementarias. De todos modos, siguiendo el módulo es posible utilizarlos como recursos sonoros autónomos independientemente de los conocimientos previos. Para los alumnos que se encuentren en esa situación, proponemos experimentar libremente con las escalas mencionadas para entender los conceptos, sin necesidad de entender las explicaciones teóricas que se proponen. Éstas deben entenderse sólo como guías adicionales para los estudiantes ya familiarizados con teoría musical occidental clásica o moderna.

A continuación detallamos algunas escalas que nos pueden resultar útiles:

a) Escalas diatónicas: consisten en las escalas de siete notas que nos son más habituales. Muchos instrumentos musicales están pensados para facilitar el uso de algunas de estas escalas, como por ejemplo los pianos y teclados (todas las teclas blancas, en escala diatónica de do) o los saxofones (uso casi lineal de las claves principales, normalmente en escala diatónica de mi^{b} o si^{b}). Se utilizan en la mayoría de composiciones musicales, en especial las canciones populares más simples, aunque también son muy presentes en las composiciones complejas.

Del mismo modo que una escala diatónica es una selección de notas de la escala cromática, a la hora de componer es muy útil dar protagonismo a subconjuntos específicos de notas para desarrollar las ideas.

Por ejemplo, los acordes tríadas se forman si se toman tres notas de la escala separadas entre ellas, es decir, con al menos otra nota de separación entre una y otra. Teniendo en cuenta que las octavas se encadenan formando un continuo, sólo hay tres ordenaciones posibles entre cada conjunto de tres notas (por ejemplo, en una escala de do mayor, re la y fa se pueden ordenar como la-re-fa, re-fa-la, fa-la-re). Buscando la combinación que presenta las menores distancias entre las tres (en este caso de ejemplo re-fa-la, que es la única que no presenta dos notas entre alguna de ellas) se pueden ordenar siempre como primera (re), tercera (hace) y quinta (la), llamándolas así según la distancia respecto a la primera de ellas (re en este caso). Las otras dos combinaciones se consideran inversiones del mismo acorde, dado que la sonoridad entre ellas no difiere sustancialmente. Si se utilizan simultáneamente se generarán acordes, mientras que el uso consecutivo ordenado de las notas creará arpegios. Este principio se puede extender de manera similar a otros acordes y arpegios más complejos, como los de séptima o novena, aunque entonces hay que considerar dos octavas para calcular las distancias. También se pueden formar otros acordes y arpegios que no siguen directamente esta regla, tales como acordes suspendidos, arpegios de cuartas perfectas, etc. en cualquier caso el principio es el mismo: la elección de las notas protagonistas determina la evolución armónica de la composición o, viceversa, una armonía dada nos ayudará a decidir qué notas tendrán protagonismo en cada tramo musical.

El uso de melodías diatónicas basadas en arpegios de tres notas es extremadamente extendido entre las músicas de videojuegos, ya que genera melodías muy fáciles de reconocer pero que no distraen la atención del jugador.

Escala diatónica de do mayor. Se corresponden con las notas blancas de un piano.

b) Escalas pentatónicas: se llaman así las escalas que utilizan sólo cinco notas por octava, normalmente consisten en una simplificación de las escalas diatónicas. Un buen ejemplo son las pentatónicas que se generan tocando sólo las notas negras de un piano (pentatónica mayor de fa#). Como ya hemos comentado, las pentatónicas suelen ser fáciles de utilizar a la hora de componer incluso para quien no tiene experiencia musical previa. Son muy utilizadas especialmente en música rock y similares (desde blues a rock progresivo, punk

o metal). También se utiliza mucho en músicas tradicionales de todo el mundo, en especial algunos de sus modos son muy útiles para crear melodías de reminiscencia asiática.

Pentatónica mayor de fa#. Se corresponde con las notas negras de un piano. Otras escalas de 5 notas con relaciones diferentes entre ellas también se consideran pentatónicas.

c) Escalas de tonos enteros: consisten en escalas hexatónicas aumentadas o, lo que es más importante, presentan distancias uniformes de un tono entre sus notas. Son interesantes desde un punto de vista matemático ya que los intervalos que utilizan son exactamente dobles respecto la escala cromática ($fn = f_{n-1} \times (\sqrt[6]{2})$). También es interesante utilizar puntualmente las notas que se derivan del arpegio de esta escala ($f_n = f_{n-1} \times (\sqrt[3]{2})$). Son especialmente utilizadas en jazz moderno y en espectáculos musicales, tanto de teatro como de cine, ya que permiten crear fácilmente melodías fantásticas e irreales.

Escala de tonos enteros. La elección de las notas alternativas (D^b, E^b, F, G, A, B) también crearía una escala de tonos enteros complementaria y equivalente.

d) Otras escalas: existen muchísimas escalas con nombre propio, cada una de ellas con una sonoridad particular. A menudo la misma escala tiene varios nombres, especialmente cuando se utilizan en ámbitos diferentes. Algunas nos son relativamente habituales, especialmente a través del jazz, como su escala melódica menor. En general, sin embargo, son escalas con sonidos menos «estables» o más disonantes que los de las escalas diatónicas o pentatónicas, es decir, crean sonoridades que no se asocian con los mayores y menores que nos son habituales y por tanto transmiten fácilmente misterio, inestabilidad, o incomodidad, como hace fácilmente la escala de terceras disminuidas ($f_n = f_{n-1} \times (\sqrt[4]{2})$). De todos modos también son importantes a la hora recrear músicas concretas, en caso de querer ambientar musicalmente nuestros videojuegos en lugares o comunidades que se asocian con estas escalas, como podría ser el uso de la escala bizantina que se asocia a culturas árabes y especialmente a comunidades romaníes.

Arriba, escala melódica menor de jazz en su modo VI, también conocida como escala semidisminuida o Locrio#2. Abajo, escala bizantina, también conocida como escala mayor doble armónica, magiar o zíngara mayor

En caso de componer, es importante entender que la elección de una escala principal no excluye el uso de otras escalas. Del mismo modo que podemos basarnos en arpegios o pentatónicas para crear una melodía diatónica, podemos cambiar la escala momentáneamente, por ejemplo utilizar notas de paso cromáticas, cambiando completamente las relaciones a través de frases basadas en dominantes secundarias, utilizando escalas disminuidas de transición (construidas sobre el semitono entre dos notas fundamentales consecutivas), etc. Si el uso de estas escalas es breve y se vuelve enseguida a la escala original puede pasar desapercibido para el oyente, aumentando sutilmente la complejidad de la composición al mismo tiempo.

Por otro lado, si hacemos un análisis de las distancias entre las escalas anteriores basándonos sólo en las imágenes, podremos ver fácilmente que las podríamos clasificar en dos tipos: las que presentan patrones regulares y son simétricas dentro de la octava (escala cromática, escala de tonos enteros, escala de terceras disminuidas), y las que no lo son (diatónica, pentatónica...). Las segundas, que son las más habituales, presentan fenómenos de modalidad, hecho que aumenta sus posibilidades sonoras. Retomaremos el concepto de modalidad más adelante, antes de todos modos hace falta introducir otros recursos con DIN que nos permitirán entenderlo mejor.

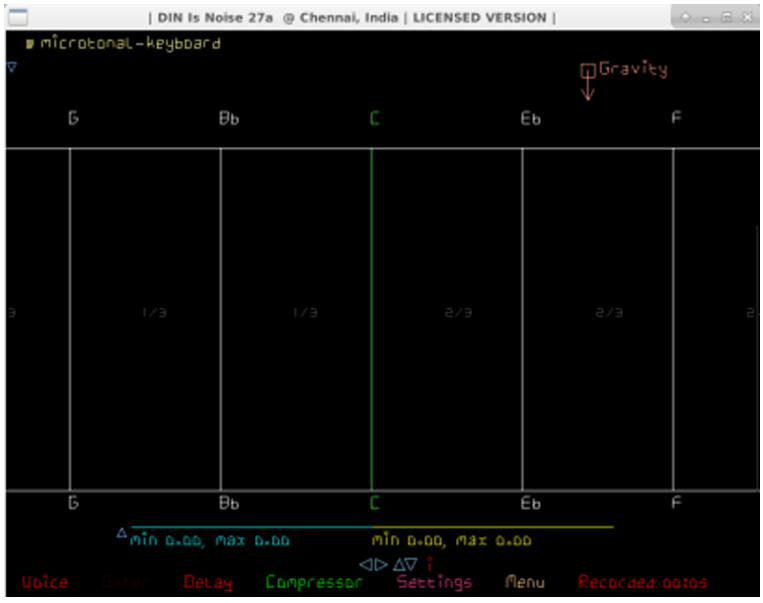
1.3.4. Otros sonidos de ambientación con DIN

Aparte de las melodías, que consisten en secuencias temporales de sonidos, para el diseño sonoro nos interesa también la superposición simultánea de frecuencias para crear sonidos. Ya hemos explicado el ejemplo de los acordes en el apartado anterior. El timbre o forma de onda también se podría considerar una superposición de frecuencias. En composiciones musicales nos interesará que los sonidos superpuestos guarden relaciones matemáticas entre ellos, pero para otros sonidos nos interesará precisamente que no las guarden.

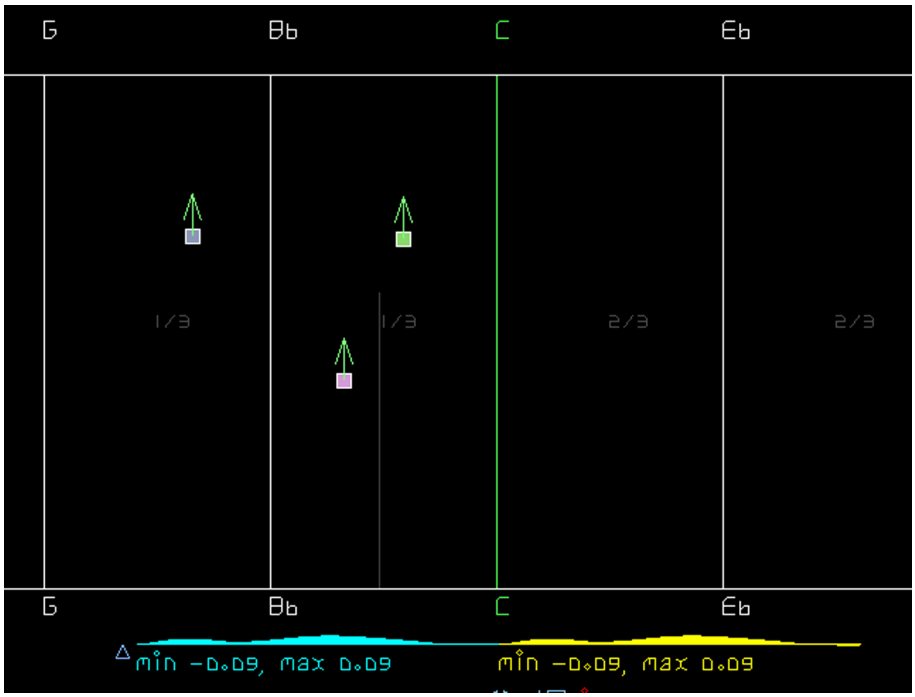
Intérprete tocando un Theremin. La posición de los dos brazos controlan el volumen y la afinación de manera parecida a como se hace en el teclado microtonal de DIN.

Fuente: By What's On the Air Company - What's On the Air, July 1930 (page 45), Public Domain, https://commons.wikimedia.org/w/index.php?curid=56188934

DIN nos permite trabajar visualmente superponiendo generadores de frecuencias llamados drones, que imitan los drones musicales asiáticos, pero se controlan a partir de un sistema de representación cartesiana similar al funcionamiento del Theremin, un instrumento muy utilizado en diseño sonoro durante los primeros tercios del siglo XX. Para utilizar los drones en DIN, accederemos al *microtonal-keyboard* a través de la tecla 1 o a través del menú (botón secundario del ratón o botón correspondiente en la parte inferior derecha de la pantalla). El teclado microtonal tiene el siguiente aspecto:



Una vez accedemos a él por primera vez, si procede, debemos configurar de nuevo la escala en *settings*, ya que la escala de este instrumento es independiente de la del *keyboard-keyboard* y por defecto vuelve a ser la pentatónica de jazz aunque hayamos cambiado la primera. Para el ejemplo utilizaremos la escala diatónica de do mayor que ya conocemos.



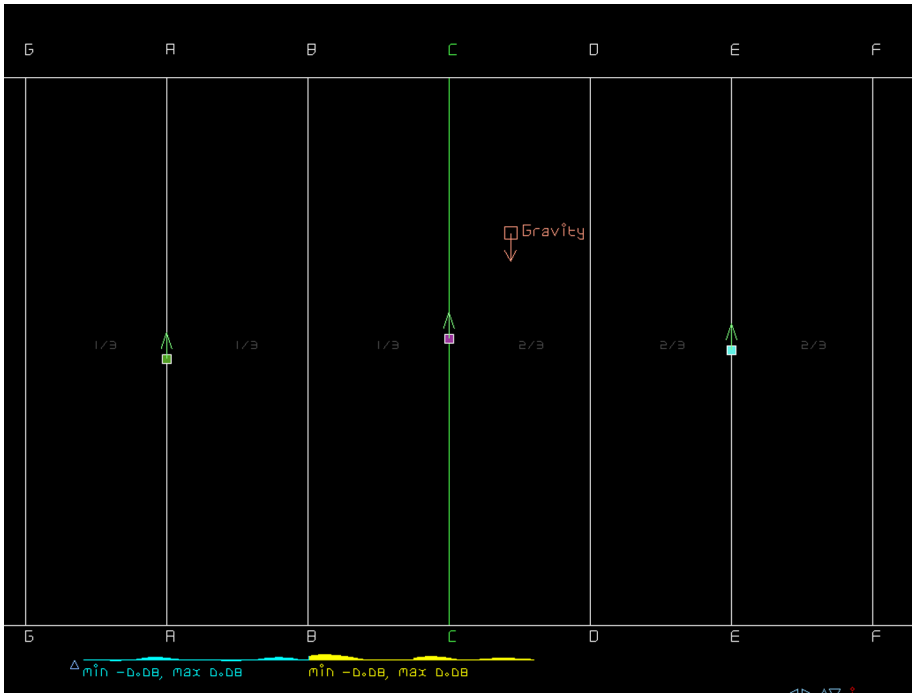
El teclado microtonal funciona como un Theremin o, en otras palabras, como un gráfico logarítmico: eje de las X en Hz (aproximado, según la escala) y eje de las Y en decibelios. Cada vez que se pulsa la tecla generaremos un «dron» en la posición que ocupe el ratón, que a su vez generará una nota enfrecuencia y volumen correspondientes a su posición:

Podemos posicionar los drones libremente en cualquier punto de la pantalla. Si hacemos clic con el botón primario del ratón podremos seleccionarlos. A través del menú o de la tecla E podemos mover los drones seleccionados. También a través del menú o de las teclas M y C, respectivamente,podemos seleccionarlos todos y eliminarlos, cosa muy útil cuando empiezan a ser demasiados.

La forma de onda asociada por defecto a un dron es entre sinusoidal y triangular, pero que se puede editar accediendo al editor con la tecla 8 y editando manualmente (o seleccionando entre las diferentes formas de onda a través de la opción *library* del menú o de las teclas 9 y 0). En DIN hay varios editores y opciones que no entraremos a detallar, ya que en general se trata de funciones muy específicas de ese programa, aunque el estudiante puede acceder a través del menú o del teclado, a modo similar a los que describimos. También es posible programar directamente el comportamiento de los drones para conseguir resultados más específicos.

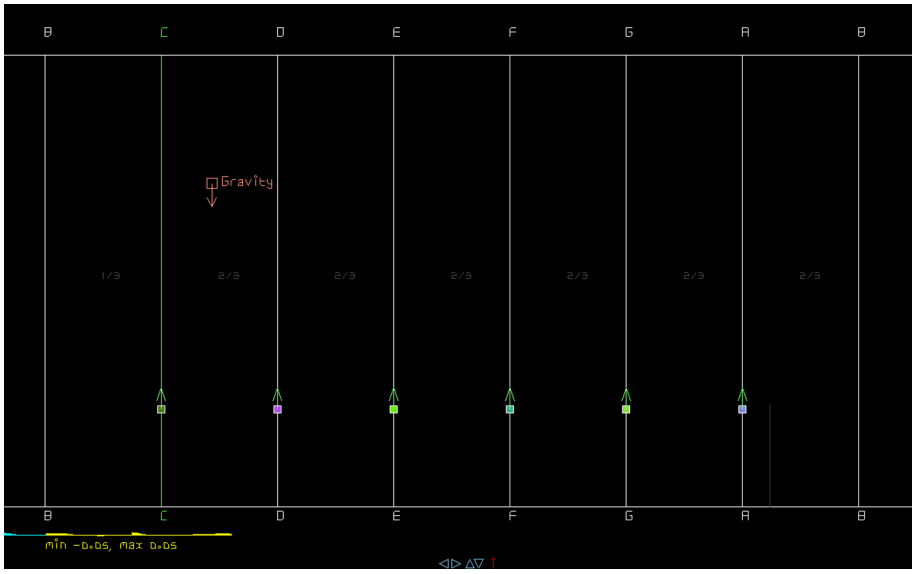
Podemos crear acordes fácilmente en el teclado microtonal si situamos tres o más drones en las posiciones correctas, tal y como hemos comentado al hablar de la escala diatónica. Para facilitar la tarea podemos seleccionar la opción *Snap drones to notes* en el menú o pulsar la tecla *ins*. Un ejemplo de acorde tríada sería el siguiente:

Tres drones posicionados para formar el acorde tríada de la menor

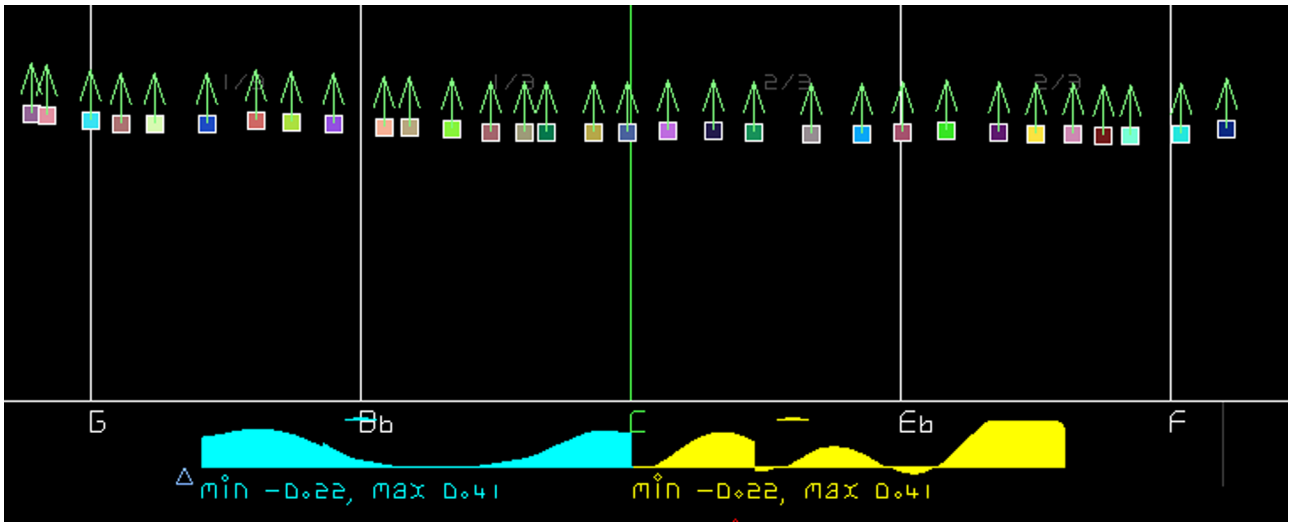


Aunque el instrumento microtonal por sí solo es bastante incómodo para hacer cambios de armonía, podemos introducir fácilmente una nota o acorde, dejarlo sonando, y volver al instrumento *keyboard-keyboard* para tocar una melodía encima. De esta manera podemos desarrollar fácilmente las melodías sobre un acompañamiento armónico (nos faltará todavía una herramienta para el acompañamiento rítmico, que veremos más adelante). Este recurso también nos interesará mucho cuando hablemos de modos.

De todos modos el teclado microtonal de DIN tiene muchas más utilidades que la de crear los acordes musicales que hemos comentado. En nuestro juego *Space Shooter*, por ejemplo, nos puede ayudar creando sonidos que se relacionan con la exploración espacial y las civilizaciones alienígenas. Sonidos de este estilo se consiguen fácilmente si se añaden varios drones en notas consecutivas de la escala diatónica, creando tricordes:



o bien, después de desactivar la opción *snapdronestonotes*, colocando los drones aleatoriamente, especialmente sobre el eje de las x:



Como se puede observar, estos últimos recursos nos ayudan a crear sonidos similares a los que se utilizaban en las antiguas películas de ciencia ficción. Combinando esta técnica con las siguientes que explicaremos podemos generar una gran cantidad de efectos sonoros.

1.3.5. Drones en movimiento

Tal y como también se puede intuir por su nombre (y aunque la palabra musical no guarda relación directa con los vehículos no tripulados), los drones de DIN se pueden mover, en este caso sobre el espacio de trabajo. Las posibilidades de movimiento del sintetizador son muy grandes, incluyendo lanza-

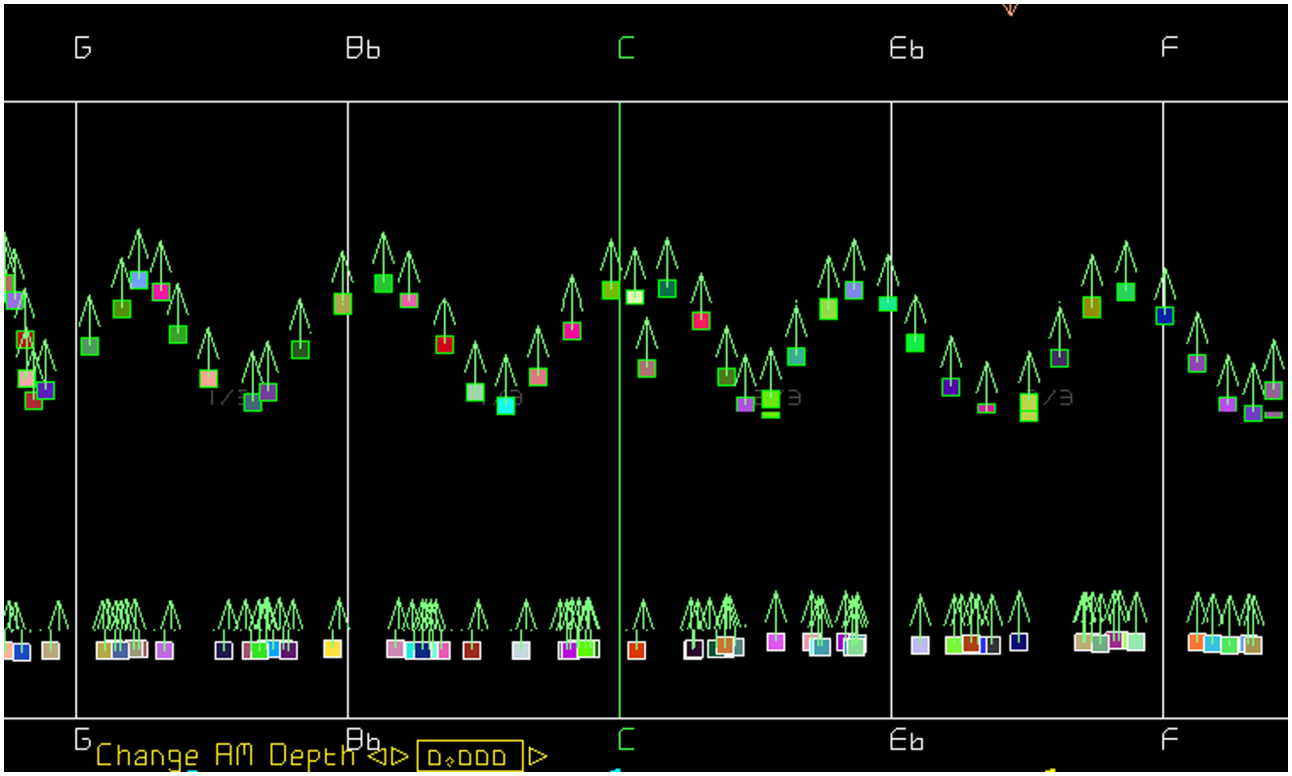
mientos, órbitas y efectos gravitatorios, de todos modos sólo describiremos los dos movimientos básicos, que tienen especial relevancia por sus consecuencias sobre el sonido.

Por un lado, tal y como hemos dicho, la posición vertical del dron afecta el volumen. Por lo tanto, si hacemos oscilar el dron en esta posición podemos conseguir una intermitencia en el volumen. Este efecto, que en música se conoce como *tremolo*, se puede conseguir seleccionando los drones y modificando el parámetro *AM Depth* a través del menú o de las teclas r y t.

Por otro lado, la posición horizontal determina la frecuencia. Aplicando una vibración horizontal, pues, se consigue el efecto conocido como *vibrato*, que nos puede ser útil por ejemplo para recrear ambulancias. El efecto se aplica modificando el parámetro *FM Depth* o con las teclas y y u.

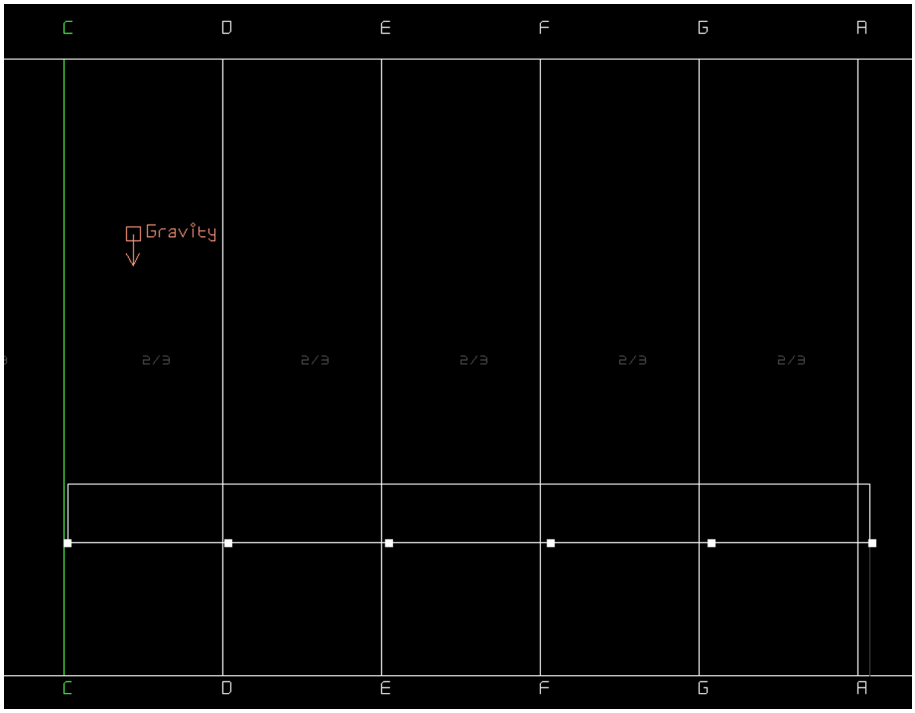
Si se selecciona un conjunto de drones independientes y se les aplica un movimiento vertical u horizontal, se podrá observar que no se mueven unánimemente sino a destiempo, en un patrón que dependiendo de la distancia entre drones puede parecer aleatorio pero que, en realidad, es sinusoidal:

Dos filas uniformemente distribuidas de drones independientes a los que se ha aplicado respectivamente un movimiento AM (fila superior) y FM (fila inferior).



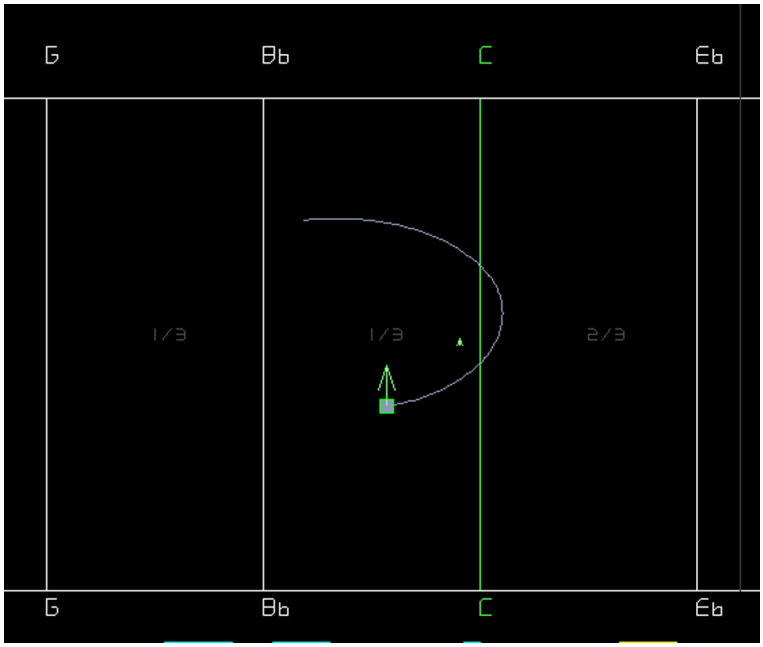
Se puede editar el movimiento sincronizado fácilmente a través del editor *drone modulation* (tecla 3). Para sincronizarlos, sin embargo, la mejor manera es crear juntos en una matriz a través de *create drone mesh*; luego se pueden recolocar libremente y serán afectados por la modulación sin el patrón sinusoidal que modula a destiempo los drones independientes.

Creación simultánea de varios drones con la opción *mesh*



También se pueden crear efectos sonoros interesantes si se combinan las dos vibraciones anteriores, con lo que los drones describen un movimiento circular. Este recurso es especialmente útil a la hora de recrear zumbido de insectos:

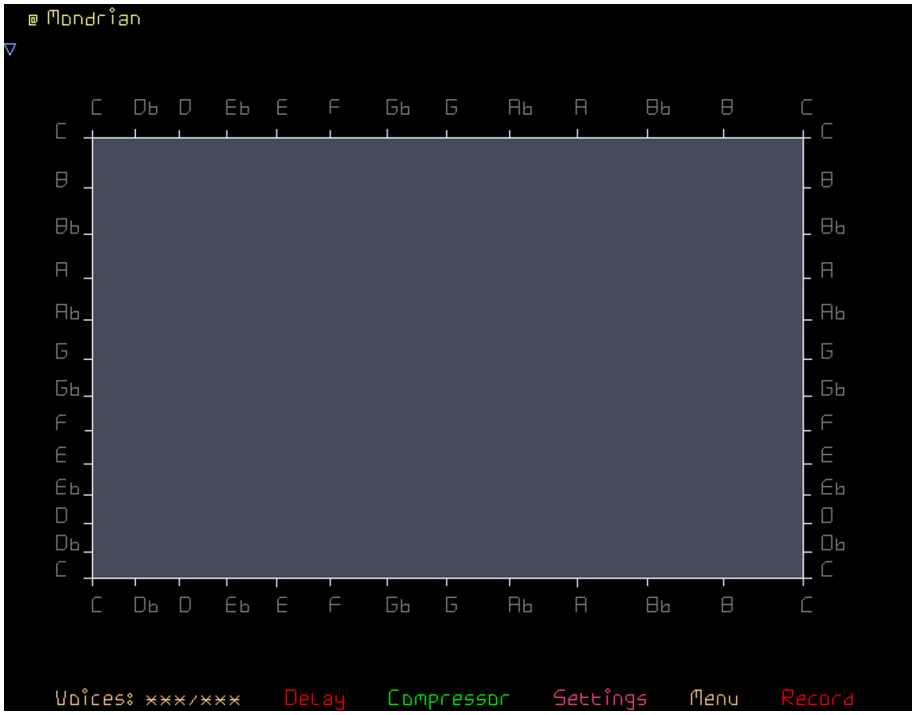
Dron en movimiento circular, combinación de FM Depth y AM Depth. Se le ha activado la opción de cola para mostrar el movimiento en al imagen.



1.3.6. Secuencias temporales pseudoaelatorias con DIN

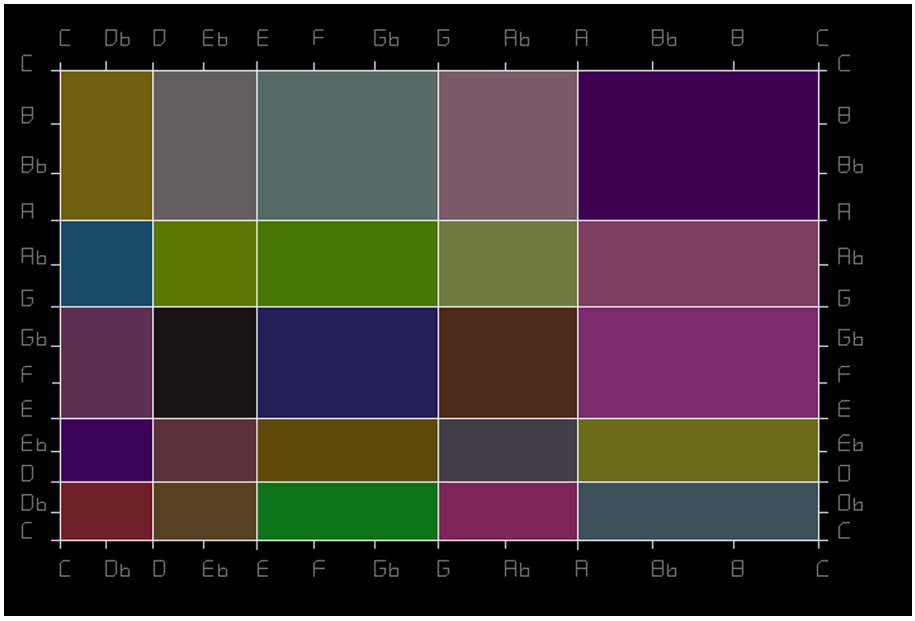
Hasta ahora hemos trabajado sobre todo aspectos relacionados con la dimensión de las frecuencias del sonido. De todos modos, la otra dimensión fundamental del sonido tiene que ver con la distribución y cambio de los sonidos en el tiempo, es decir, con la estructura y con el ritmo.

DIN hace difícil el diseño sonoro en su dimensión temporal si no se complementa de otras herramientas, como son los secuenciadores o los controladores midi. De todos modos, tiene un generador secuencial llamado *Mondrian*, en homenaje a los cuadros del holandés Piet Mondrian, que permite generar secuencias temporales interesantes. Encontraremos el instrumento, como siempre, buscando en el menú o pulsando la tecla 1, después del *microtonal-keyboard*. Por defecto, presenta el siguiente aspecto:



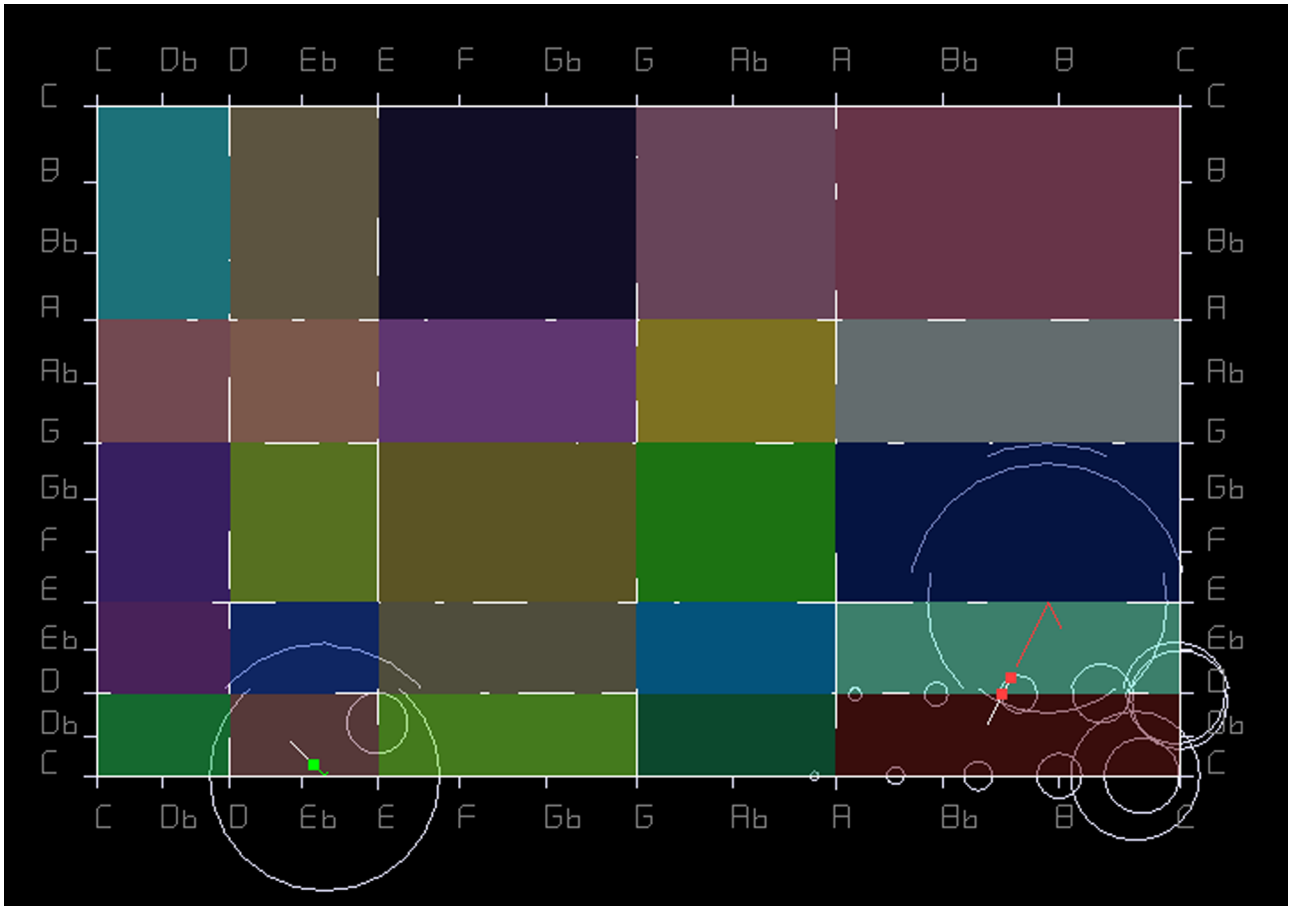
El instrumento *Mondrian* consiste en una caja dividida en uno o más compartimentos rectangulares, donde se mueven pelotas (*balls*). Funciona de manera similar a un juego de pinball, pero generando notas cada vez que las pelotas rebotan en alguna de las paredes. La posición de las paredes determina la nota que se genera al golpearla. Es un instrumento bastante heterodoxo, difícil de utilizar rítmicamente, pero capaz como veremos de generar fácilmente atmósferas sonoras.

En primer lugar, vamos a crear las cajas. Lo podemos hacer con las teclas f o r, que crearán divisiones horizontales o verticales respectivamente en el punto donde esté el cursor. También las podemos borrar con la tecla *v*. De todos modos es recomendable elegir una escala simple en *settings*, como una pentatónica, y crear directamente una retícula a través del menú con la opción *Make note grid*, borrando algunas de las cajas posteriormente si lo queremos. El resultado final puede ser cualquiera, por ejemplo con los parámetros propuestos sería similar al siguiente:



Una vez creada la retícula, se pueden añadir las pelotas. Es recomendable hacerlo a través del menú, con la opción *Add*; para crearlas, podemos pulsar en el punto donde queramos insertarla o bien hacer un clic largo mientras movemos el ratón. Esta segunda opción «lanzará» el balón en la dirección y velocidad calculada según la trayectoria del ratón. Como el resto de instrumentos de DIN, también hay otras opciones que se pueden explorar a través del menú o programar directamente, aunque no entraremos a detallarlas.

Podemos añadir tres tipos de pelotas: normales (*bouncers*), destructoras de la retícula (*Wreckers*) o reconstructores de la retícula (*healers*). Una opción interesante, por ejemplo, consiste en crear un Wrecker rápido y un healer lento, por lo que entre los dos vayan cambiando la configuración de las paredes y por tanto generando melodías diversas:



El volumen, timbre, trayectoria etc... se pueden regular a través de los menús, en las secciones *Editors*, *Parameters*, *Tools* y *Misc*, de manera similar a lo que hemos ido viendo con los demás instrumentos. Combinando los instrumentos *Mondrian*, *microtonal-keyboard* y *keyboard-keyboard*, y los conocimientos básicos que hemos ido exponiendo, podemos crear atmósferas musicales experimentales pero completas, aunque no tengamos conocimientos musicales previos. Para crear canciones normales, sin embargo, tendremos que utilizar otras herramientas y conocimientos que superan el alcance de este módulo, como trackers, secuenciadores, o programas de grabación multipista.

1.3.7. Sonoridades musicales adicionales: modos

Tal y como hemos ido comentando en varios puntos, hay una cuestión importante a la hora de conseguir sonoridades diferentes y que hemos reservado para el final: los modos.

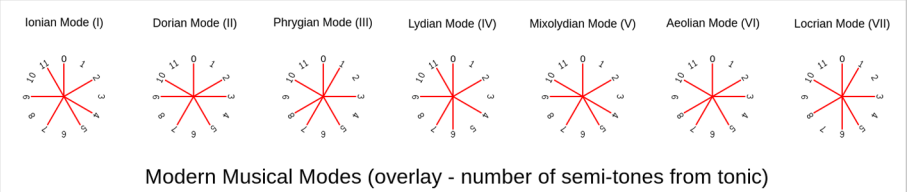
Como hemos visto, muchas de las escalas son asimétricas dentro de la octava. Como las octavas se encadenan entre ellas, de una forma que podríamos considerar circular, nos es posible comenzar a utilizar la escala desde cualquiera de sus notas. Por ejemplo, podemos empezar una pentatónica menor en un do, o en un fa#, y aunque la afinación es parecida, si las dos escalas guardan la misma relación entre ellas tendrán sonoridades similares.

Sin embargo, si en vez de empezar a buscar las relaciones de las notas a partir de la primera nota de la escala empezamos por otra cualquiera de ellas, las distancias de semitonos entre la primera y las siguientes serán diferentes (lo que no ocurre con las escalas simétricas, como hemos avanzado). En algunos casos concretos dos escalas "diferentes" consisten precisamente en eso, como por ejemplo las escalas diatónicas de la menor y de do mayor, que usan ambas todas las notas blancas del piano. Estas dos escalas, como veremos mejor a continuación, son en realidad dos modos de la misma escala diatónica.

En música popular las escalas se suelen dividir entre mayores (descritos general como «alegres» o decididos) y menores (descritos como «tristes»), basándose en la distancia en semitonos que guardan entre ellas la primera nota (fundamental) y la tercera. Una escala diatónica de do mayor, como hemos visto, contiene las mismas notas que una la escala diatónica de la menor. Pero el acuerdo tríada basado en la primera (do-mi-sol) es mayor, mientras que el acuerdo tríada basado en la segunda (la-do-mi) es menor. Es por ello que, entre ellas, se consideran respectivamente como relativas mayor y menor.

Existe excepciones a la clasificación general entre escalas mayores y menores. Una escala se considera disminuida cuando la distancia en semitonos desde la primera nota hasta la quinta es un semitono menor de lo habitual. Por contraste, una escala se considera aumentada cuando la distancia de la cuarta es un semitono mayor de lo habitual.

Relaciones entre las notas de los diferentes modos modernos de la escala diatónica. La numeración indica la distancia en semitonos respecto la nota fundamental.



Fuente: By User:Tcolgan001 - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=10705942

Cuando basamos el acorde en cada una de las notas de la escala, pues, logramos acordes mayores o menores. El resto de las notas, sin embargo, guardan relaciones diferentes porque la escala es asimétrica. La escala diatónica pues tiene siete combinaciones posibles, otras escalas como la octatónica (también conocida como Korsakoviana o escala disminuida de jazz) a pesar de tener más notas tienen sólo dos. Cada combinación de la escala según la nota fundamental es precisamente lo que se conoce como modo, y cada modo tiene una sonoridad propia que aporta un matiz propio a la escala.

Aunque el orden de los modos es circular, se suele designar uno de ellos como «primero» y numerar consecuentemente el resto para saber de qué modo se está hablando. Por ejemplo, ordenando la escala pentatónica que se crea con las notas negras de un piano, que ya conocemos, damos lugar a cinco modos que se llaman, entre otros, con los siguientes nombres:

1) Pentatónica menor (sonoridad menor, triste)

2) Pentatónica mayor (sonoridad mayor, alegre)

3) Escala egipcia (sonoridad norteafricana o asiática occidental)

4) Escala Man Gong (sonoridad asiática oriental)

5) Escala Yo (sonoridad asiática oriental, especialmente japonesa).

A pesar de los diferentes nombres y sonoridades de cada modo, repetimos, las notas que la componen son exactamente las mismas. Así pues, la sonoridad de una escala con modos depende del contexto, y no de las notas de la escala en sí.

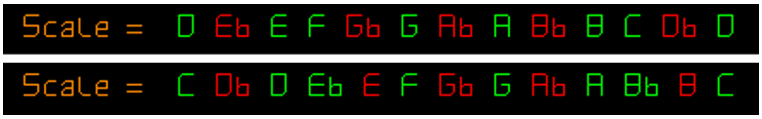
El concepto de modo suele crear mucha confusión, especialmente entre los músicos autodidactas, ya que a menudo cuesta ver la utilidad del concepto. Hay que tener en cuenta que el «modo» aparece en una escala concreta, por ejemplo la pentatónica de la menor, cuando ésta se acompaña con una armonía (por ejemplo un acorde) que da protagonismo a la primera nota del modo. Si cambiamos el acorde por otro, o si cambiamos la escala sobre el mismo acorde, el modo cambia. Es muy aconsejable que los alumnos que quieran entender el concepto hagan sonar menos la nota fundamental de un acorde mientras experimentan con el modo escogido (por ejemplo con un dron en el *microtonal-keyboard* mientras se experimenta con una escala en el *keyboard-keyboard*).

Una confusión extra a la hora de hablar de modos es que diferentes períodos históricos y escuelas utilizan los mismos nombres para referirse a diferentes modos de diferentes escalas. Según los postulados de la música occidental actual, los modos diatónicos modernos son los siguientes:

- **Iónico**: el más utilizado de los mayores. Tiene una sonoridad que podríamos describir como «decidida», en contraposición al resto de modos. Es el modo mayor más utilizado en música popular occidental.

- **Dórico**: modo menor, aunque relativamente más decidido que los otros modos menores, como por ejemplo el eólico, ya que más de las relaciones entre sus notas son parecidas a las de los modos mayores.

Dos ejemplos de dórico. En la parte superior, re dórico (con do como nota fundamental). En la parte inferior, do dórico (con si^b como nota fundamental). Es importante prestar atención al hecho que, aunque las relaciones entre las notas de las dos son las mismas, las notas utilizadas en cada una de ellas corresponden a las de la escala iónica de la respectiva fundamental.



- **Frigio**: modo menor, a medio camino entre el eólico y el locrio, que transmite cierta desesperación o rabia. Es muy utilizado en músicas como el flamenco.

- **Lidio**: modo mayor, con un sonido un poco fantástico debido a la cuarta aumentada (algo que lo aproxima a la sonoridad de la escala de tonos enteros). Como curiosidad, es el modo utilizado en la conocida sintonía de entrada de la serie The Simpsons, y el que le da su peculiar sonoridad.

- **Mixolidio**: modo mayor, característico por la séptima disminuida, algo que lo sitúa en sonoridad entre el jónico y el dórico. Es muy utilizado en músicas como el funk.

- **Eólico**: es el másestable entre los modos «tristes», y por consecuencia el más utilizado de los menores diatónicos.La música popular suele escogerlo como modo principal junto al jónico.

- **Locrio**: su sonoridad es opuesta a la del modo lidio. Se trata de un modo menos utilizado en música occidental, ya que es el único modo disminuido de la escala diatónica y en consecuencia su sonoridad es mucho más inestable que la del resto.

Aunque el uso consciente de los modos a la hora de componer requiere de cierto conocimiento de causa, se utilizan de manera intuitiva cada vez que se hacen cambios armónicos en la estructura de una canción simple. Por ejemplo, una estructura armónica muy simple pero muy utilizada consistiría en do mayor (jónico) seguido de fa mayor (lidio), sol mayor (mixolidio), y finalmente regreso a do mayor (jónico). También se pueden utilizar por ejemplo para entender las estructuras con dominantes secundarias, como por ejemplo do mayor (jónico) seguido de re mayor (segunda dominante, mixolidio), sol (tónica provisional que se transforma otra vez en dominante, mixolidio) y finalmente resolviendo a do (jónico).

Reto 2:

Elige un proyecto de Unity y planifica para él algunos efectos sonoros complejos similares a los que hemos utilizado. Como alternativa fácil a los tres instrumentos de DIN puedes usar por ejemplo Multiplayer Piano (http://www.multiplayerpiano.com), la recreación Theremin de Stuart Memo (https://stuartmemo.com/smashing-magazine/theremin/) y Otomata (http://earslap.com/page/otomata.html)

2. Diseño sonoro en entornos 3D

Cada vez más, la experiencia de juego tiende a ser inmersiva. En el caso del audio, tradicionalmente, la tendencia había sido aumentando el número de altavoces. Seleccionar qué altavoz reproducirá el sonido ayuda a posicionar los sonidos en el espacio. Con algunas técnicas básicas de mezcla, además, es posible crear sonido altamente envolvente. De todos modos, en los juegos, el factor de interactividad es lo suficientemente grande como para hacer difícil un trabajo acústico espacial satisfactorio sólo con las herramientas tradicionales del cine. Esto hace que el diseño sonoro en videojuegos sea bastante particular.

Actualmente, la potencia de la mayoría de CPU (a menudo ayudada por DSP) es suficientes como para hacer mezclas y aplicar filtros en tiempo real. De esta manera el volumen, posición y otras características del sonido se pueden alterar en función de lo que nos interese en cada momento. En especial, estas posibilidades abren la puerta a lo que se llama sonido 3D, es decir, la simulación del espacio sonoro *in situ*. Hacer un diseño sonoro en un videojuego moderno, pues, presenta ciertas características específicas que hay que conocer.

Unity integra nativamente el sonido 3D, de hecho lo hace en tal grado que hasta trata el sonido 2D como caso particular del mismo. Entender cómo funciona el sistema antes de conseguir las muestras de audio es clave para potenciar al máximo las características del sonido. Iremos viendo las peculiaridades de Unity transversalmente a lo largo del módulo.

2.1. Obtención de material sonoro: diálogos y atmósfera sonora

Si estamos trabajando en producciones pequeñas, muy probablemente intentaremos buscar librerías de sonidos en internet para solucionar la mayoría de situaciones con el mínimo esfuerzo (y, probablemente, con mejores resultados que si lo intentáramos nosotros). Es muy fácil encontrar la mayoría de sonidos que necesitaremos, a veces gratuitamente y a veces comprando entre inmensas librerías sonoras, que se utilizan sobre todo para cine y televisión.

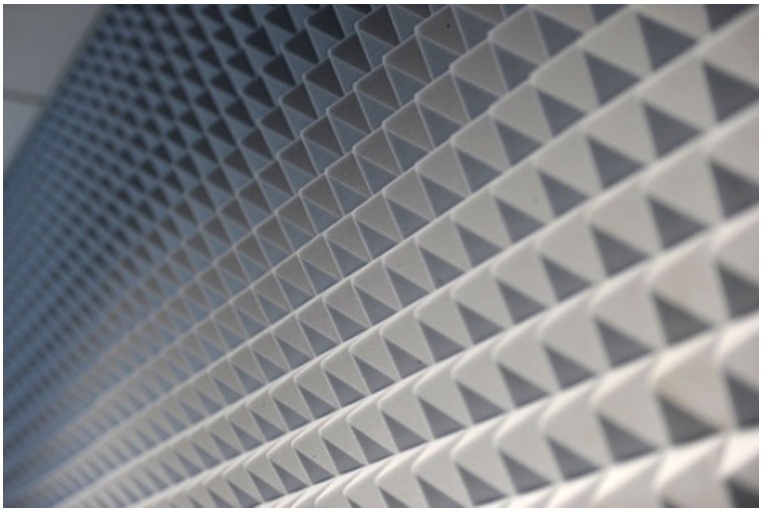
De todos modos hay algunos casos donde nos será imposible encontrar el sonidos que necesitamos. El caso más habitual es el de los diálogos. En este caso, si no se dispone de un equipo de grabación en condiciones, es recomendable ir a un estudio de locución o de sonido.

En caso de grabar los sonidos nosotros mismos, hay que tener en cuenta los siguientes aspectos:

- **Espacio acústico**: si se utiliza un buen equipo de grabación, las condiciones acústicas del espacio pueden ser las limitantes. Es importante que el

espacio de grabación no tenga interferencias sonoras, como chirridos de parqué, ventiladores internos del ordenador, o incluso zumbido eléctrico debido a interferencias con la corriente alterna. El material de las paredes y otros objetos presentes en la sala puede resonar e interferir en el audio que grabamos; por norma general, es preferible que el espacio de grabación sea "seco", por ejemplo revistiendo las paredes con espuma acústica.

Espuma acústica piramidal típica de un estudio de sonido.



Fuente: By Guillaume Paumier - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=16533839

- **Micrófono**: el micrófono que utilizamos determinará muchas características de la grabación. Más allá de la calidad del mismo, cada micrófono tiene diferentes características que lo hacen adecuado a uno u otro uso. Los micrófonos dinámicos son muy sensibles a la distancia, por lo que filtran los sonidos lejanos fácilmente. Por el contrario, micrófonos como los de cinta dan una coloración particular al sonido que nos puede interesar, pero son mucho más sensibles a la velocidad que a la presión. Por otro lado los micrófonos subcardioides y omnidireccionales nos permiten captar mejor el sonido ambiente, mientras que los direccionales (*shotgun*, supercardioides, bidireccionales) permiten aislar los sonidos que nos interesan de las interferencias que provienen de otras direcciones. Para voces humanas, por ejemplo, es recomendable utilizar micrófonos de condensador de patrón cardioide.

Micrófono de condensador Neumann, muy utilizado para registrar voces. A la izquierda se aprecia un filtro pop, que ayuda a grabar mejor sonidos como los de las letras *s* y *p*.

Fuente: By Andi mueller 77 - Own work, CC BY-SA 3.0,https://commons.wikimedia.org/w/index.php?curid=19967745

- **Preamplificador**: a parte del micrófono, es el elemento que más influye en la calidad de sonido. En general el preamplificador se encuentra integrado dentro de los convertidores analógico-digital (DAC) o incluso dentro del mismo micrófono si funciona con conector usb, por lo que pasa desapercibido. En estudios de sonido, sin embargo, suele ser un módulo separado dada su importancia.

- **DAC**: en caso de ser un elemento independiente del preamplificador, suele ser mucho menos importante, ya que se limita a convertir en formato digital la frecuencia y profundidad de las ondas analógicas. En general nos interesará grabar a por lo menos 44,1KHz y 16 bit. 48kHz y 24bit es una resolución habitual para generar los másters finales de audio, aunque algunos DAC llegan a digitalizar a 192kHz y 32bit *floating point*.

- **Software de grabación**: aunque en general no tiene un impacto importante sobre el sonido, algunos programas permiten grabar y tratar varias pistas simultáneamente (DAW, de *Digital Audio Workstation*) o monitorizar el clip de vídeo que nos interese para sincronizar el audio. Más allá de utilizar o no un sistema de grabación multipista, es importante grabar los sonidos en stereo siempre que sea posible, en especial el sonido ambiente, por ejemplo posicionando dos micrófonos iguales en posición perpendi-

cular (formando una X). Esto nos facilitará mucho la recreación de sonido 3D en Unity.

En general, nos interesa grabar los sonidos lo más limpios y naturales posible. Hay que prestar siempre atención en que el volumen de entrada sea alto, para aprovechar al máximo la ratio entre sonido útil y ruido de fondo, pero que en ningún momento los volúmenes máximos nos saturen el rango dinámico del que disponemos, porque esto crearía artefactos de distorsión.

Una vez tengamos los clips grabados, muy probablemente nos interesará editarlos antes de importarlos en Unity. Algunos de los programas más utilizados en producción son ProTools, Logic y, en otros niveles más modestos, Audacity. De todas maneras hay muchas alternativas de gran calidad, como Reaper, Ardour y Audition. Las modificaciones posibles a los sonidos originales son muchas Nos puede interesar, por ejemplo, eliminar ruidos indeseados. Audition es una herramienta muy potente en cuanto a la limpieza sonora.

También podemos manipular el sonido para conseguir nuevos sonidos. Por ejemplo, partiendo de un simple chirrido de madera, si reducimos la velocidad a aproximadamente un 10% y sobreponemos el mismo sonido varias veces conseguiremos un sonido de terremoto bastante bueno. En estos casos, cuando no podamos conseguir exactamente el sonido que queremos de una librería, la creatividad es esencial. Aunque veremos detalles de trabajo con Audition en este módulo, es casi esencial conocer una herramienta equivalente para hacer un buen trabajo de audio en cualquier producto audiovisual, incluidos evidentemente los videojuegos.

2.2. Integración en Unity

Una vez hemos generado todos los clips y los hemos importado en nuestro juego, es el momento de comprobar que todos los efectos sonoros realmente se integran con nuestro juego, que tienen una coherencia entre ellos, y que interaccionan correctamente sin crear efectos indeseados. En general, si hemos hecho un buen trabajo, sólo tendremos que controlar unos pocos aspectos sencillos. Hacer un trabajo bien hecho en este sentido, sin embargo, no siempre es fácil. Como ya hemos dicho desde el principio, si no se es ya un experto en la materia es siempre preferible dejar en manos de especialistas todo el diseño sonoro.

2.2.1. Jerarquia espacial

Como ya sabemos, Unity gestiona el sonido a través de *AudioSources* y del *AudioListener*. Este hecho no tiene mucha importancia cuando el sonido es 2D, pero es esencial en juegos 3D. Cada esfera sonora presenta sus características a la hora de distribuirlas espacialmente:

- **Efectos sonoros**: para todos los efectos sonoros "reales", nos interesará vincular el *audiosource* directamente con la fuente, es decir, con el objeto o individuo que los emite. Esto nos permitirá una localización precisa y dinámica del sonido en un entorno 3D. Los sonidos subjetivos, en cambio, suelen ser independientes de la escena y por lo tanto pueden ser 2D. Por norma general, los efectos sonoros objetivos serán mono, mientras que los subjetivos pueden sacar más ventaja del *stereo*.

- **Sonido ambiental**: el sonido ambiental no está ligado directamente a ningún objeto, pero no por ello pierde la posibilidad de ser 3D. El *audiosource* debería de tener una distancia mínima que abarcase toda la zona ambiental, por lo tanto la acción se desarrolla en su interior hasta que no se sale de él. Si tenemos la suerte de poder grabar un sonido ambiente con dos micrófonos bidireccionales, como hemos visto en el apartado anterior (o si por lo menos disponemos de clips en stereo real) es interesante modificar los parámetros spread y pan para que los dos ejes sonoros, fijos en el espacio, se muevan respecto al *audiolistener* y el sonido sea completamente envolvente. Si lo hacemos correctamente, conseguiremos que haya sutiles diferencias en rotar el *audiolistener* en la escena, como pasa en la realidad con todos los sonidos aunque no sepamos de donde vengan.

- **Diálogo**: Dependiendo de si el diálogo proviene de los personajes presentes en la escena o no, los podemos implementar igual que el resto de efectos sonoros o como sonido 2D. De todos modos a menudo es interesante que el volumen mínimo sea siempre alto, para que el jugador no se pierda ninguna parte importante del diálogo. Si son diálogos fuera de campo, también nos puede interesar modificar ligeramente *pan* para dar más espacio al sonido 2D.

- **Música**: en general la música es un sonido 2D completamente independiente de la escena. Seguramente nos interesará activar también las opciones de *bypass* para que ni los filtros ni las zonas de *reverb* la afecten.

2.2.2. Validación y modificación de sonidos

Hasta ahora hemos generado clips individuales, pero una vez los integramos en el motor de juego pasarán a interaccionar entre ellos. Es habitual que durante las fases de integración se detecten discrepancias respecto al comportamiento que nos habríamos esperado de ellos. Muy a menudo, durante fases más avanzadas, es necesario volver atrás y modificar los clips de origen o incluso descartarlos y crear una versión alternativa.

Por ejemplo en el ejemplo de trabajo, *Space Shooter Mobile*, hemos partido de un diseño gráfico 2D concreto, pero primero hemos propuesto sonidos retro y luego hemos desarrollado efectos complejos más cercanos a la ciencia ficción. En este caso la elección es deliberada y sirve el objetivo de entender el dise-

ño sonoro a través de herramientas muy diferentes. Muy probablemente, si estuviéramos haciendo un diseño sonoro real para este juego sustituiríamos la mayoría de sonidos creados con DIN por otros creados con un *tracker*, secuenciador, o modificando sonidos reales con un editor como Audacity o Audition.

En ciertos casos, evidentemente, también podemos decidir mantener los sonidos de origen y hacer modificaciones para armonizar el conjunto, por ejemplo sobre los gráficos. Si en el caso de ejemplo decidiéramos mantener los sonidos realizados con DIN, seguramente decidiríamos cambiar las *sprites* actuales por imágenes más realistas, a fin de transformar la ambientación de la aplicación entera en un juego de ciencia ficción.

Algunas de las modificaciones que podemos aplicar a los clips para integrar mejor los sonidos, aparte de las que ya hemos ido viendo, son las siguientes:

- **Ecualización**: modifica el timbre del clip potenciando ciertas bandas de frecuencias. Es útil tanto para uniformizar sonidos de orígenes diferentes como para hacer destacar los unos sobre los otros, potenciando en cada caso bandas diferentes (por ejemplo, recortando la franja de 2 a 4KHz de los otros clips para dejar estas frecuencias libres para los diálogos)

- **Compresión**: reduce el rango dinámico del clip, normalmente aplicando una reducción de volumen progresiva (*ratio*) sólo a partir de un cierto volumen establecido (treshold) para no aumentar el ruido de fondo. Las herramientas de compresión normalmente permiten seleccionar los tiempos de reacción en respuesta (*attack*) y retroceso a la normalidad (*release*). Aunque de entrada ese procedimiento reduce el volumen máximo del clip, el rango dinámico extra que se gana se puede aprovechar para subir el volumen final del clip (*gain*).

- **Compresión multibanda**: es una herramienta que combina técnicas de EQ y de compresión simultáneamente. En concreto, comprime las bandas de frecuencias por separado. Es muy útil para dar cuerpo a sonidos que, pese a tener un volumen alto en alguna de sus bandas, es pobre en otras frecuencias (cosa que sucede a menudo con la voz humana).

- ***Pitch Switch***: estrictamente, consiste en acelerar o ralentizar el clip, lo que prueba respectivamente aumento o disminución de las frecuencias. A través de métodos como la transformada discreta de Fourier es posible cambiar la frecuencia sin cambiar la velocidad y viceversa con una pérdida de calidad relativamente baja.

- ***Reverb***: simula el rebote natural del sonido sobre diferentes espacios y materiales, añadiendo una "cola" al sonido original con cierto retraso variable. Colas más largas simularán materiales más resonantes, como las paredes de una cueva. Retrasos mayores servirán también para simular espacios más grandes, donde el sonido tarda más a rebotar y volver atrás. La mez-

cla entre sonido directo y sonido reverberado (*dry / wet*, respectivamente) también ayuda a posicionar los objetos en el espacio, ya que en caso de encontrarse cerca del oyente llegarán a este sobre todo por vía directa (*dry*), mientras que a mayor distancia más peso relativo tendrá la reverberación sobre el sonido directo (*wet*).

- **Delay**: genera efectos de eco a través de repetir una copia del sonido con retraso y a menudo con modificaciones. A parte de los evidentes efectos de eco que se pueden conseguir con ello, también ayudan a dar más cuerpo a los sonidos. Si las repeticiones son muy cortas o con oscilaciones temporales puede producir modificación del timbre, dando lugar a lo que se consideran efectos por sí solos, como el *phaser* que ya conocemos, el *chorus* o el *flanger*. En realidad la reverberación también es un caso particular de *delay*, aunque a efectos prácticos se suelen tratar todos como efectos separados.

A la hora de aplicar cualquiera de estos filtros se notará una gran diferencia entre archivos de origen de alta calidad, (por ejemplo, WAV a 96kHz) y archivos de menor calidad (especialmente comprimidos con pérdida, como mp3, aunque sean a la máxima calidad de 320Kbps). Es un buen momento para recordar que, siempre que se trabaja con media no vectorial, la posibilidad de manipular el material de origen viene limitado por la cantidad de información del mismo, independientemente de si somos capaces de percibirla en su totalidad o no. La calidad de las transformaciones depende también de la calidad de los filtros utilizados, que a veces también pueden imponer pérdidas de calidad inesperadas y difíciles de detectar.

Como se puede observar, muchos de estos efectos se pueden aplicar directamente en Unity. De todos modos, siempre que sea posible, es recomendable reservar esta opción sólo para los clips que queremos modificar durante el juego, ya que el uso de estos efectos hará uso intensivo de la DSP o, peor aún, de la CPU, lo que puede llegar a tener un impacto importante sobre el rendimiento de nuestra aplicación, en especial en plataformas poco potentes como dispositivos móviles.

2.2.3. Volumen

A diferencia de lo que ocurre con la mayoría de *assets*, los volúmenes del proyecto sí se pueden modificar en *Game Mode*. Para hacerlo, hay que activar la opción *Edit in Game Mode* en el inspector. Esta opción resulta muy útil ya que sería muy difícil determinar los volúmenes por adelantado.

Cuando el proyecto es complejo, sin embargo, nivelar los volúmenes puede ser difícil. Es por eso que es muy recomendable organizar los sonidos jerárquicamente. Los volúmenes a Unity se pueden modificar a través de 5 puntos de la cadena, que en general hay que ajustar en orden, del primero al último:

- **Volumen de origen**: el volumen del archivo original, que puede variar de un archivo a otro. En general es útil mantener activada la opción *normalize* del inspector para hacer coincidir el volumen máximo del clip con el máximo del rango dinámico (aunque en sonidos que se deben mantener siempre flojos, tales como ruido ambiente, nos puede interesar desactivarlo).

- **Volumen del AudioSource**: permite dar un volumen entre 0 y 1 (el volumen máximo pues coincide con el volumen original). Será útil para nivelar los volúmenes entre diferentes *AudioSource* de una misma categoría, como veremos después.

- **Posicionamiento 3D**: en el caso de sonidos 3D, la distancia del *AudioSource* respecto al *AudioListener* afecta el volumen. Podemos modificar el perfil de respuesta a través de la opción 3D *Sound Settings* del inspector. En especial, hay que indicar la distancia mínima a partir del cual el volumen será máximo (1) y la distancia máxima a partir de la cual el mismo se dejará de oír totalmente (0). Hay especificar también la curva de respuesta; en general nos interesará la curva original (¡hay que recordar que la percepción del sonido es logarítmica!).

- **AudioGroup**: nos permite regular el volumen de varios AudioSource simultáneamente o, por extensión, de otros *AudioGroups* contenidos en el mismo. Aunque en proyectos muy pequeños no son necesarios, es recomendable estructurar siempre una jerarquía clara y detallada de *Audiogroups*, empezando por los de las esferas sonoras. A la hora de mezclar proyectos complejos nos será mucho más fácil modificar los volúmenes si empezamos nivelando cada grupo individualmente y luego los nivelamos entre ellos. Hay que tener en cuenta que el volumen de los *Audiogroups* no trabaja como los anteriores, de 0 a 1, sino que lo hace en decibelios. Aunque podemos llegar a dar a cada grupo una ganancia de + 20dB, hay que vigilar porque cualquier aumento de volumen puede llevar fácilmente a distorsionar la mezcla final.

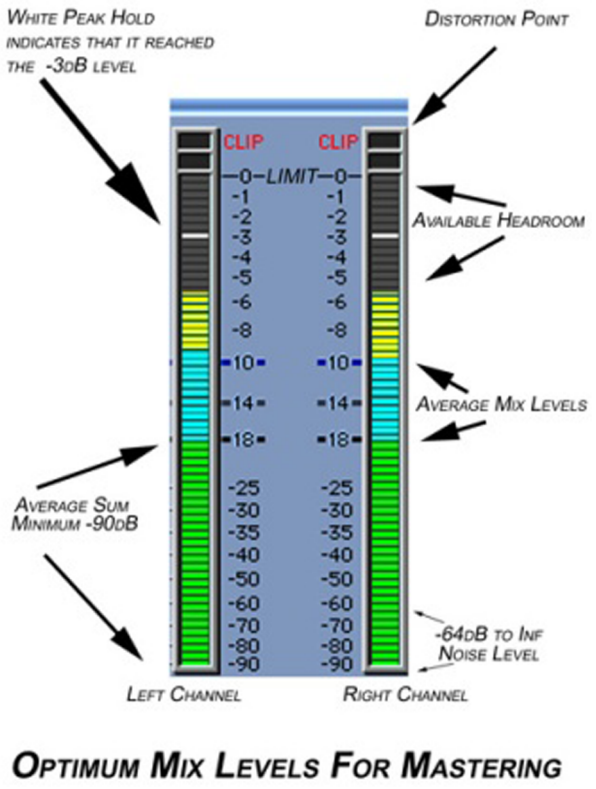
- **AudioMixer**: es la mezcla principal, de la que dependen todos los demás *Audiogroups* y *Audiosources*. Es lo que tenemos que controlar durante todo el proceso de mezcla y testeo para controlar que el volumen final no sature el rango dinámico del que disponemos.

Como es evidente, los clips deben guardar un equilibrio de volúmenes entre ellos. Además, a la hora de buscar este equilibrio es necesario forzar todas las situaciones del juego, por ejemplo donde el máximo de sonidos activan si-

multáneamente, con el fin de asegurarse de que los volúmenes combinados no distorsionan. Unity dispone de un indicador de tipo VU con indicador de *peak* (esto es, las finas líneas que aparecen momentáneamente en el extremo superior) que nos permite monitorizar tanto el volumen medio como el máximo. En general es suficiente nivelar los sonidos guiándose por el propio oído, procurando siempre que el indicador *Master* del *AudioMixer* deje un margen prudencial de al menos 6 dB (*Headroom*).

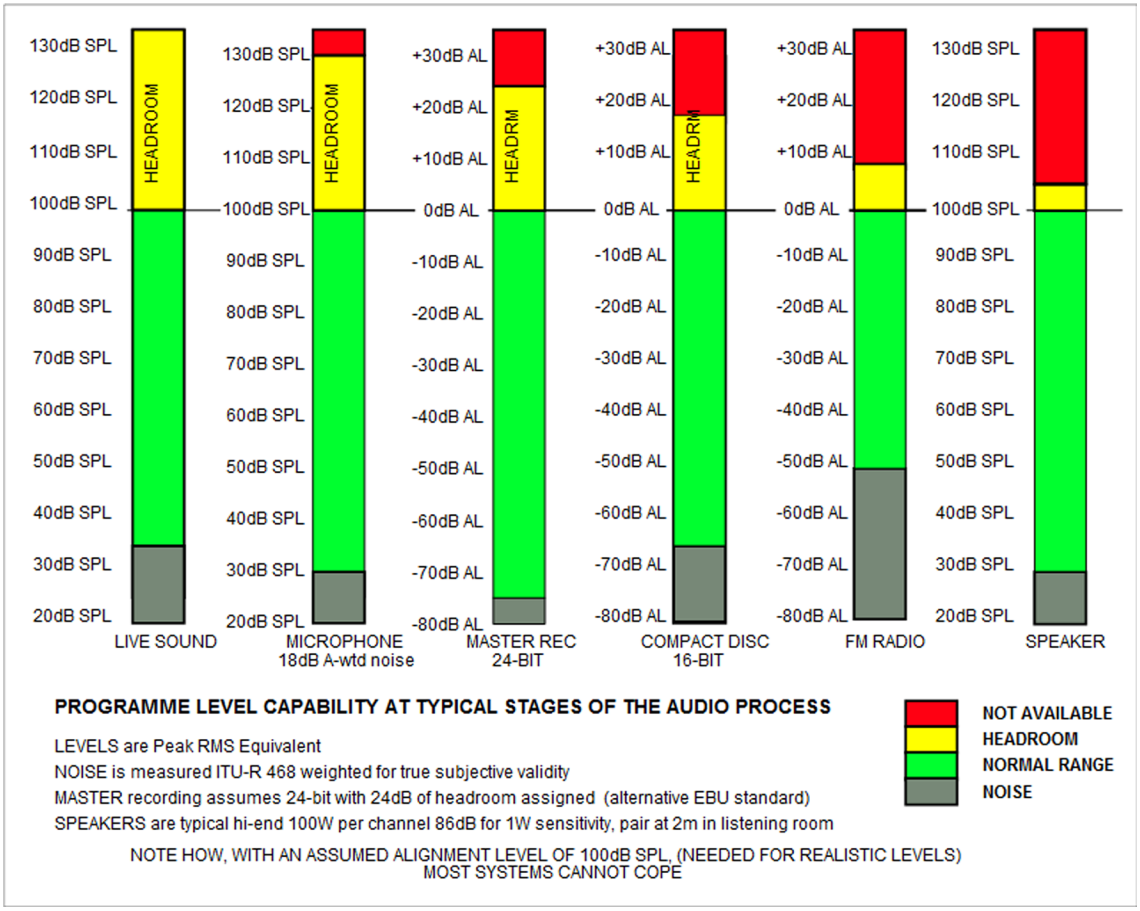
La relación entre el volumen y la percepción del mismo es una cuestión compleja y no trivial. Los sistemas digitales deben basarse estrictamente en la máxima amplitud de onda ya que si ésta sobrepasa en cualquier momento el umbral máximo que se puede representar (96dB para 16bit, 144dB para 24bit) el sonido final será necesariamente distorsionado. De todos modos la percepción del volumen, tanto para los humanos como para los altavoces que reproducirán el sonido, no depende tanto de la onda individual como de las características del fragmento sonoro. Una onda cuadrada a la máxima amplitud, por ejemplo, se percibe con un volumen mucho más alto que una onda sinusoidal de la misma amplitud y frecuencia. Diferentes frecuencias también se perciben diferentemente. Se puede dar el caso, pues, que sonidos dentro del rango dinámico del apoyo produzcan sonidos saturados en los altavoces (*peak*), o que sonidos saturados se perciban con un volumen bajo (VU). Los indicadores VU y peak de Unity intentan ilustrar estos dos fenómenos por separado. A veces para resolver ciertas situaciones donde VU y *peak* no van de acuerdo habrá que aplicar técnicas adicionales, tales como compresión, ecualización, o compresión multibanda.

Ejemplo de volúmenes utilizados en masters de 16bit.



Fuente: By Edward Vinatea - www.supermastering.com/mastering/, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2559275

Ejemplo de rango dinámico aconsejado en diferentes estadios de la producción sonora.



Fuente: Copyrighted free use, https://commons.wikimedia.org/w/index.php?curid=2260497

2.2.4. Testing

Una vez el conjunto de los sonidos ya está validado y mezclado, aunque es recomendable hacer una fase de testeado para detectar problemáticas específicas. En especial, es recomendable asegurarse de que los sonidos no crean un efecto negativo claro sobre el usuario en las sesiones de juego más largas que podemos esperar para nuestra aplicación.

La fase final de testeado debe hacerse siempre sobre las plataformas reales para las que estamos programando. Si estamos diseñando un juego para dispositivos móviles, con altavoces relativamente malos, no tiene sentido que hagamos la validación en un estudio con monitores de alta calidad, que reproducen los sonidos de una manera bastante diferente. Incluso en caso de una misma plataforma, como una videoconsola, nos encontraremos con jugadores que utilizan altavoces mejores o peores, o incluso que juegan con el volumen relativamente bajo en un espacio con mucho ruido, o con otra música de fondo. Tenemos que intentar que el diseño sonoro sea impecable cuando se juega en condiciones óptimas, pero también que sea eficiente en las condiciones más adversas.

Aunque los videojuegos en sí no compiten entre sí a nivel de volúmenes, hay que tener en cuenta que medios como la música digital o la parrilla de programación televisiva viven desde hace décadas lo que se llama la guerra de decibelios. Esta "guerra" consiste en la tendencia de sacrificar rango dinámico para conseguir el máximo volumen posible y destacar sobre la competencia. Si la plataforma de destino puede reproducir también otros audiovisuales (televisión en el caso de videoconsolas, notificaciones de escritorio en caso de ordenadores personales, etc...) deberíamos tener en cuenta si pueden interferir con nuestro juego, y en tal caso adaptar los volúmenes de nuestra aplicación por tal que no interfieran con la experiencia del usuario (ya sea por notificaciones demasiado fuertes simultáneas al juego o bien por cambios de volumen repentinos en cambiar la televisión de la videoconsola a un canal de televisión).

2.3. Optimización

2.3.1. Formatos de archivo

La optimización de los ficheros de audio dependerá de qué motor de juego estemos utilizando. Unity, por ejemplo, acepta 8 formatos de audio que podemos distribuir en tres categorías:

a) Formatos PCM no comprimidos

- .WAV → el formato PCM no comprimido de Windows por excelencia. Prácticamente idéntico al AIFF.

- AIFF → el formato PCM no comprimido de Mac. Prácticamente idéntico al WAV.

b) Formatos PCM comprimidos

- .MP3 → formato propietario con pérdida.

- .ogg → formato libre con pérdida.

c) Ficheros de módulo (*Module Files*)

- .mod → El archivo utilizado originalmente por el *Ultimate Soundtrack* (soft propietario). Está diseñado específicamente para el chipset original de *Amiga*.

- .S3M → Utilizado originalmente en *ScreamTracker* (propietario). Es una modificación extendida del MOD original.

- .XM → Utilizado originalmente por *FastTrack 2* (propietario, aunque a la hora de cerrar los materiales se está desarrollando el clon *FastTrackerII* bajo una licencia indefinida). Otra modificación popular del MOD original.

- .IT → El archivo utilizado originalmente por *Impulse Tracker* (bajo licencia BSD). Todavía es relativamente utilizado en algunos de los juegos actuales de más renombre.

A la hora de importar el audio en Unity, debemos tener en cuenta que no todas las esferas sonoras presentan las mismas características, por lo que nos puede interesar utilizar uno u otro formato en cada caso.

Algunos, como la música o el sonido ambiente, suelen ser archivos largos que no requieren una sincronización exacta con las imágenes. En estos casos no será crítico que se activen inmediatamente y, al mismo tiempo, nos interesará que sean ligeros, para que consuman poca RAM. Es recomendable, pues, que en la versión final del juego se compilen en formatos comprimidos, como ogg, y a poder ser que se llamen a través de métodos de *streaming* para optimizar los tiempos de carga y el uso de memoria. En caso de haber creado creado la música con un *tracker* es recomendable importar directamente el formato respectivo, para utilizarlo en las plataformas que lo permitan, aunque es posible renderizarlo en Unity mismo a través del *Asset Import Inspector*.

Otros, como la mayoría de efectos sonoros y parte de los diálogos, suelen ser cortos pero deben reaccionar con precisión al desarrollo de la acción. En este caso nos interesará cargarlos por adelantado en un formato PCM como wav o AIFF, que a pesar de que ocupan más no están comprimidos y por lo tanto pueden invocarse inmediatamente cuando el juego lo requiera. En caso de considerar que el ahorro de memoria es un factor importante, la opción ADPCM de Unity es un buen compromiso ya que permite comprimir ligeramente los clips sin que su descompresión requiera tanta CPU como la descompresión de los archivos ogg.

De todos modos Unity ya aplica por defecto una optimización de los sonidos bastante buena, y en general no será necesario preocuparnos de hacerlo manualmente. Es recomendable pues utilizar clips de origen con el máximo de calidad cuando sea posible, como por ejemplo archivos WAV o AIFF a 48kHz y 24bit.

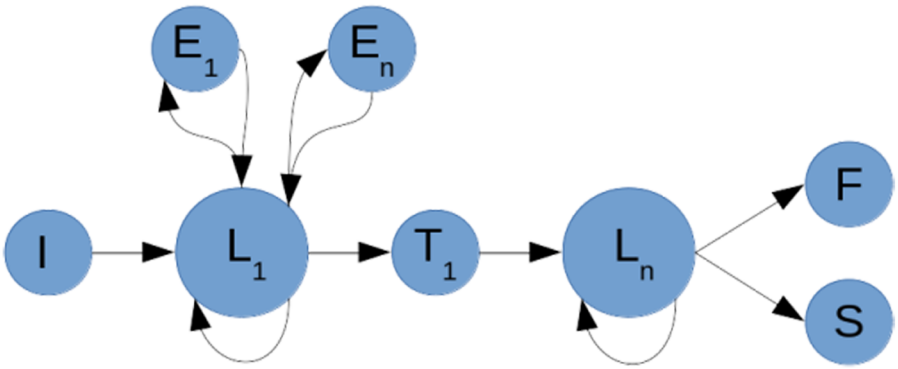
2.3.2. Archivos musicales

La música que acompañará el juego, en algunos casos, puede ser una pieza lineal. En especial pasará cuando haya vídeos introductorios o transiciones predefinidas donde el tiempo es lineal y en las que el usuario no puede inter-

actuar. También puede darse el caso de que un juego o nivel se pueda cronometrar con precisión, como ocurre a veces en *Kid Icarus: Uprising*. En caso de que se pueda predecir con precisión cada evento, se puede crear la música de manera que se adapte perfectamente a la escena, de manera similar a lo que se puede hacer en una película o clip de vídeo tradicional.

En el resto de casos, sin embargo, las estructuras rígidas de la música contrastan con la flexibilidad de la acción del juego. A menudo nos interesará que la música tenga un comienzo, que luego pueda repetirse indefinidamente mientras no se acabe el nivel, que reaccione a los eventos aleatorios, y que pueda terminar enseguida cuando nos interese. Todos estos aspectos, a no ser que se utilice música modificada en tiempo real, se deben prever en el momento de crearla. En general, pues, los archivos de música requieren de un paso extra para su optimización.

La solución más fácil pasa por concebir la música misma en bloques, de manera similar a como hemos trabajado el *scripting* de *AudioSource*, y generar un fichero diferenciado para cada uno de ellos:



- **Introducción** (I): es el primer archivo que invocaremos, y en general sonará una sola vez por nivel. La mayoría de las veces nos interesa que sea corto y conduzca rápidamente hacia el *loop* principal.

- ***Loop* principal** (L): es la parte principal de la música, que se repite indefinidamente en un *loop*. Puede ser corta, pero en general interesa que sea relativamente larga y variada, en especial si prevemos que el usuario la escuche durante mucho rato. También podemos programar triggers para hacer saltar la música de un *loop* a otro a partir de *checkpoints* específicos, consiguiendo que la atmósfera musical se adapte mejor a cada parte del nivel.

- **Eventos** (E) y **Transiciones** (T): son clips que interrumpen el loop principal. En el caso de los eventos, devolverán al *loop* principal (el punto donde

la habían dejado o reiniciándolo). En el caso de las transiciones, en cambio, llevarán a un nuevo *loop* o a la finalización.

- **Finalizaciones** (*Failure*, *Success*): son los clips que interrumpirán el *loop* y finalizarán el nivel. Normalmente tendremos uno en caso de éxito y otro en caso de fracaso.
  A la hora de invocar los archivos, sólo tendremos que indicar a Unity por qué archivo empezar y qué hacer cuando termina de reproducir un bloque o cuando se desencadena un evento concreto. De esta manera conseguiremos un alto grado de dinamismo en la música sin tener que recurrir a otras técnicas más complejas, como las que veremos a continuación.

2.3.3. Música adaptativa

La máxima expresión de dinamismo en el diseño sonoro se consigue cuando, añadido encima de todo lo que hemos visto hasta ahora, la música del juego se adapta dinámicamente a la acción. Si se hace un buen trabajo, se puede conseguir que la banda sonora parezca creada especialmente para cada partida individual.

En el *Asset Store* podemos encontrar varios paquetes de música adaptativa. En caso de que estemos interesados en crearla nosotros mismos, hay varias técnicas que nos permiten generarla efectivamente:

- **Seleccionar las canciones de la *playlist* según *tags***: a pesar de ser bastante rudimentario, si se utilizan canciones externas, especialmente en un *open world*, se pueden categorizar los ficheros en sí y crear una *playlist* que elija la siguiente canción relacionando marcadores de la zona o nivel del juego con *tags* asociados a cada canción. También se puede utilizar la misma técnica para ajustar la afinación de los efectos sonoros según la afinación de la canción que esté sonando en cada momento, en tal caso los metadatos de la canción tienen que contener el tono y a través de un *script* deberemos modificar el *pitch* de los efectos sonoros.

- **Vincular el tempo o la afinación de los sonidos a la velocidad o posición del protagonista**: ayuda a intensificar fácilmente la experiencia de juego según la intensidad de cada momento. en caso de trabajar con audio en formato PCM, la solución más fácil es vincular directamente la velocidad de la cámara principal con el parámetro de *pitchshift*. En caso de utilizar módulos *tracker* es posible modificar la velocidad y la afinación por separado, vinculante por ejemplo la primera con la velocidad horizontal y la segunda con la posición vertical.

- **Alternar entre *loops* similares**: si se crean dos o más clips con estructura y ritmo similar, se pueden sincronizar para cambiar de uno a otro según convenga. Los usos son muchos, en especial ayuda a acentuar la adquisi-

ción de poderes especiales, los estado de salud críticos, etc. Para hacer la transición entre uno y otro, en general, basta con hacer que los volúmenes se releven en un *crossfade* (*fade in* y *fade out* simultáneos). Las diferencias entre las distintas versiones de los *loops* pueden incluir cambios en la intensidad, en la orquestación, en el modo musical, o incluso consistir en músicas aparentemente diferentes unidas entre ellas por pequeñas transiciones. *Monkey Island 2* es un buen ejemplo clásico de implementación de esta técnica.

- **Activar y desactivar capas sonoras**: llevando al límite el caso anterior, se puede diseñar la música por capas que funcionen independientemente y se añadan o desaparezcan según nos convenga en función de diversos *triggers*. Por ejemplo, podemos crear un *loop* musical donde sólo haya las capas de bajo y batería en caso de pausar el juego, aparezca un arreglo normal cuando la acción es tranquila, se añadan capas adicionales según ciertos *triggers* de intensidad, y una orquestación completa cuando la tensión es máxima, como cuando la pantalla está llena de enemigos.

- **Vincular diferentes espacios con diferentes instrumentos**: otra alternativa del arreglo por capas consiste en sustituir una a una las capas por otras equivalentes y compatibles. Por ejemplo, hacer que la capa rítmica cambie de percusión caribeña a secuencias de *drum'n'bass*, que el acompañamiento rítmico cambio de piano a guitarra eléctrica, que la melodía pase a estar interpretada de un violín a una marimba, etc.

- **Modificación de parámetros de los instrumentos**: esta técnica, para aplicarla como tal, sólo se puede hacer en caso de utilizar instrumentos virtuales, aunque se puede emular generando los clips por adelantado y aplicando la técnica que hemos comentado antes. Consiste en modificar en tiempo real los parámetros de los sintetizadores, a fin de modificar el sonido fácilmente al activar ciertos *triggers*. *Portal 2* es un buen ejemplo de aplicación de esta técnica.

- **Programar música pseudo-aleatoria**: con suficiente conocimiento y paciencia, la música se puede deconstruir en unidades mínimas de cada instrumento (*riffs*, *liks*) y programar la canción a través de métodos de generación por procedimientos, para que la música de cada partida sea siempre única. Un ejemplo magnífico de esta técnica combinada con muchas de las anteriores la podemos encontrar en *Legend of Dungeon* de Calvin Goble.

- **Adaptar la animación a la música**: como el sonido suele tener un patrón temporal bastante rígido, a veces la mejor solución será programar las animaciones para que se sincronicen con estos y no al revés. Sólo necesitamos mantener un contador de la estructura rítmica de la canción (probablemente por muestras) y sincronizar los puntos claves de las animaciones con las mismas. Así podemos hacer por ejemplo que los pasos de los personajes sigan el tempo de la música, lo que en general será una solución

más natural que no modificar el ritmo de la canción para que se adapte a animaciones que pueden empezar en cualquier momento.

Las características de los videojuegos, sumados con la potencia de computación actual, hacen que los retos sean muchos y que las posibilidades de la banda sonora sean ilimitadas, sólo comparables quizás a los pianistas que improvisaban mientras musicaban en directo las primeras películas de cine mudo. Nuestro límite, pues, sólo se encuentra en buscar la solución óptima según los recursos que podemos invertir en nuestro diseño sonoro y el retorno que podemos esperar una vez distribuimos el juego. Virtualmente, podemos crear música adaptativa original para cualquier minijuego, pero probablemente sólo nos será rentable implementarla en grandes producciones o en proyectos personales que no estén ligados por costes de plantilla ni por calendario de producción estricto.

2.4. Diseño sonoro dinámico

El diseño sonoro es ya de por sí una actividad creativa. Los departamentos de *foley*, que son los encargados de la recreación sonora en estudio para radio y vídeo, suelen generar los sonidos a partir de objetos que no guardan ninguna relación con las imágenes o ideas que acompañarán. Por ejemplo, es conocido el uso de cáscaras de coco para recrear el galope de caballos. A veces esta elección se debe porque capturar el sonido real es difícil o costoso, pero a veces el sonido recreado funciona mejor que captar el sonido original o incluso sustituye la carencia del mismo. Por ejemplo, muchos juegos y películas crean sonidos para escenas espaciales cuando ningún tipo de sonido se propaga en el vacío, ya menudo se trituran cáscaras de nuez para evocar mejor huesos rotos en escenas violentas.

Los sonidos por un motor de juego con audio 3D no son una excepción, y es interesante utilizar las herramientas que nos ofrece para recrear situaciones sonoras que intensifiquen la experiencia de juego. Dependiendo de las preferencias, los cambios se pueden aplicar sólo a sonidos concretos o también a las esferas subjetivas, como previó por ejemplo Grant Kirkhope en componer la música de *Banjo Kazooie*. A continuación detallamos algunas situaciones habituales fáciles de planificar:

- **Cuevas y otros espacios con reverberación**: Unity dispone de zonas de reverb que se pueden ajustar para simular las condiciones de acústicas de cada espacio. Ciertos espacios, como iglesias o mazmorras, son evidentes. Pero se pueden aplicar más sutilmente a cambios de espacio. Si el juego se desarrolla en diferentes espacios es interesante que todos los interiores tengan un punto de reverberación propio, en general más agresivo como más grande y vacío sea el espacio.

- **Efecto *Doppler***: como ya hemos ido comentando a lo largo del módulo, Unity permite modificar los sonidos según el efecto *doppler*, es decir, au-

mentando la frecuencia de los sonidos que se acercan al *audiolistener* y disminuyendo la de los que se alejan de él. Es evidentemente interesante para sonidos que percibimos naturalmente con este efecto, tales como ambulancias, trenes, o proyectiles. Pero también puede tener un efecto interesante aplicado como efecto subjetivo en objetos que normalmente no lo tendrían, como otros personajes u objetos que simplemente cambian de velocidad respecto a la cámara principal, en especial para recrear estados oníricos o de percepción alterada.

- **Ambientes subacuáticos**: si parte de la acción pasa bajo agua, es interesante aplicar a todos los sonidos un filtro de paso bajo y un *chorus* acompañado de ruido marrón, que combinados recrean bastante bien el efecto acústico de inmersión.

- **Radios y megáfonos**: si nos interesa simular por ejemplo un megáfono en tiempo real, sólo le tenemos que aplicar distorsión al clip de la voz. De todas maneras es recomendable añadir filtros de paso, ecualizaciones y posiblemente *delay* para hacer una recreación más cuidada.

- **Explosiones cercanas y golpes en la cabeza**: especialmente en los FPS, es habitual que algunas armas generen efectos negativos sobre la percepción del personaje. En el caso de la vista, suelen cegar y dejar la pantalla en blanco por unos segundos. En el caso del sonido, lo más efectivo es aplicar un filtro de paso bajo acompañado de un sonido fuera de rango, en forma de onda sinusoidal muy aguda (unos 10KHz) para simular *tinnitus*.

- **Espacios abiertos**: de la misma manera que los espacios cerrados presentan reverberación, los espacios abiertos no acostumbran a presentarla. Se pueden utilizar *delays* para generar ecos o reverberaciones cortas de frecuencias altas para modificar el ruido del motor en los laterales de la carretera. En caso de espacios nevados o de terrenos muy planos nos puede interesar aplicar un filtro de paso alto para crear un contraste aún mayor con el resto de espacios.

Las posibilidades del sonido 3D son muy grandes, y sólo con los filtros de Unity podemos recrear muchísimas más situaciones dinámicamente. En general es recomendable fijarse en otros juegos para ver posibilidades. Pero, sobre todo, por el diseño sonoro, es muy importante investigar los profesionales del cine y otras disciplinas audiovisuales, que acumulan décadas de experiencia y pueden inspirar soluciones muy creativas que quizás, en caso de haberse implementado en algún juego, nos sería difícil de detectar por nosotros mismos. Por suerte, el diseño sonoro de videojuegos nos ofrece todas las posibilidades de una disciplina nueva con la ventaja de podernos servir de décadas de experiencia en ámbitos análogos de diseño sonoro.

Bibliografía

Altman, Rick (1992). *Sound Theory, Sound Practice*. Routledge. ISBN: 0415904579

Ament, Vanessa (2014). *The Foley Grail: The Art of Performing Sound for Film, Games, and Animation* (2.ª ed.) Focal Press ISBN:0415840856

Cipriani, Alessandro (2016). *Electronic Music and Sound Design - Theory and Practice with Max 7* (3.ª ed., vol. 1) Contemponet ASIN:B01FGOMJKI; vol. 2 (2014) Contemponet ISBN:8890548444

Collins, Karen (2008). *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*. Mit University Press Group Ltd ISBN:026203378X

Collins, Karen (2013). *Playing with Sound*. Mit University Press Group Ltd ISBN:0262018675

Farnell, Andy (2010). *Designing Sound*. Mit University Press Group Ltd. ISBN:0262014416

Fritsch, Melanie (2016). *Ludomusicology: Approaches to Video Game Music*. Equinox Publishing Ltd ISBN:1781791988

Marks, Aaron (2008). *The Complete Guide to Game Audio: For Composers, Musicians, Sound Designers, Game Developers* (2.ª ed.). A K Peters/CRC Press ISBN:0240810740

Phillips, Winifred (2017). *A Composer's Guide to Game Music*. REPRINT. Mit University Press Group Ltd ISBN:0262534495

Purcell, John (2007). *Dialogue Editing for Motion Pictures: A Guide to the Invisible Art*. Focal Press ISBN:0240809181

Sonnenschein, David (2001). *Sound Design: The Expressive Power of Music, Voice and Sound Effects in Cinema*. Michael Wiese Productions ISBN:0941188264

Sweet, Michael (2014). *Writing Interactive Music for Video Games* Addison Wesley ISBN:0321961587

Viers, Ric (2008). *The Sound Effects Bible: How to Create and Record Hollywood Style Sound Effects*. Michael Wiese Productions ISBN:1932907483

Wyatt, Hillary (2004). *Audio Post Production for Television and Film: An Introduction to Technology and Techniques* (ed. revisada). Focal Press ISBN:0240519477