CREACIÓN, DISEÑO E INTEGRACIÓN

DE AUDIO Y MÚSICA EN VIDEOJUEGOS

INCLUYE UN PROYECTO COMPLETO DE VIDEOJUEGO DESCARGABLE

EL TERROR ERRANTE

DESARROLLADO CON UNITY Y FMOD

ANTONIO VILLA

E-BOOK
PDF • EPUB

¿Cómo nació este libro?

Si alguien me hubiese dicho cuando tenía siete años, sentado delante de un Amstrad CPC 464 mientras esperaba interminables minutos a que cargara un videojuego desde una cinta de casete, que algún día escribiría un libro sobre la integración de audio y música en videojuegos, probablemente no le habría creído. Aquel ordenador llegó acompañado de un manual que despertó en mí una curiosidad inesperada: descubrir cómo unas pocas líneas de código *BASIC* eran capaces de dar vida a un programa.

Poco tiempo más tarde empezaría mi andadura musical, acudiendo a clases particulares de piano -primer instrumento que tuve entre mis manos- y que, a la postre, tendría un impacto en mi vida muchísimo mayor de lo que yo podría imaginar en aquel momento. Y es que sería la música la que, con el paso del tiempo, triunfase entre mis otras aficiones (*gaming*, programación, deportes, arqueología, etc.), optando por ella como futuro profesional y llevándome a estudiar la carrera de música, la de magisterio musical, dos másteres (en Musicología Histórica y Composición respectivamente) y un ciclo superior de FP de sonido.

No obstante, entre tanta formación musical, y, casi por accidente, un verano me matriculé en un curso de programación web, despertando de nuevo mi interés por este mundo de ceros y unos. Poco a poco fui devorando lenguajes como HTML, JavaScript, PHP, Java, CSS, etc., recordándome a mí mismo que no quería bajarme de este maravilloso barco que permitía crear tantas cosas de tan diversa naturaleza y funcionalidades desde un simple teclado.

El tiempo fue pasando, compatibilizando mis labores de compositor y docente, soñando algún día con poder plasmar en una obra dos de mis grandes amores: la programación y la música.

La luz se encendió cuando tiempo atrás un alumno me preguntó acerca de la integración de música en los videojuegos, a lo que –humildemente- le respondí que no tenía ni idea. Yo sabía cómo era componer para videojuegos, pero no sabía qué procesos se seguían para que esa música acabase finalmente sonando en el videojuego.

Despertada mi curiosidad, empecé a investigar desde diferentes enfoques. No me quise centrar exclusivamente en la integración de audio y música de manera aislada, ya que, si algún día me decidía a escribir sobre esto, era obvio que necesitaría de un conocimiento básico en la creación

de videojuegos. Mi idea era hacerme una idea global de la creación de videojuegos y después centrarme en la integración de la música y el audio.

Dicho y hecho. Poco a poco fui investigando sobre las entrañas de los videojuegos y su creación, los motores de juego, conceptos como *Components* o *Game Objects*, el uso de *assets*, y, como no: la programación. Descubrí el lenguaje C#, el cual me resultó muy atractivo y familiar ya que había trasteado anteriormente con Java.

Después de varios meses de intensa lectura, en los que me pude hacer una idea mucho más precisa de lo que era la creación de videojuegos, escogí el motor de juego para el que quería desarrollar mi proyecto –Unity- y el middleware que quería utilizar para crear los eventos de audio y música: FMOD. Las razones eran claras… y de mucho peso.

Ambos son herramientas potentísimas y, a pequeña escala, gratuitas, con un portfolio impresionante (en *Unity* se han hecho juegos de la talla de Pokémon Go! o Hearthstone), muy populares en el mundo de la creación de videojuegos (FMOD es usado en compañías tan potentes como Activision, Microsoft o Capcom) y muy intuitivas en su manejo, por lo que eran idóneas para esta aventura. Por lo tanto, accesibilidad, versatilidad y calidad fueron los criterios clave en la elección del *software* con el propósito de que tú, querido lector, no sólo puedas obtener la mayor cantidad de conocimientos posible, sino que también puedas ponerla en práctica en la mayor cantidad de juegos posible.

El siguiente paso natural fue formular la pregunta del millón: ¿sobre qué es lo que quieres escribir? Si bien una pregunta de tal magnitud al principio puede ser muy difícil de responder, más sencillo resulta responder acerca de lo que no quería escribir. No quería escribir una disertación teórica, densa y soporífera, sin ninguna o poca aplicación práctica en un entorno real y cuyo destino -tarde o temprano- fuese de pisapapeles o sujetando algún mueble dañado. Estaba claro que el enfoque debía ser práctico. Más pronto que tarde el lector debía ser quien de crear algo real –aunque fuese muy básico- pero algo real. Sin embargo… ¿qué?

Para hallar la solución creé en un escenario práctico: acabas de ser contratado en un estudio de creación de videojuegos y eres el responsable de la integración de audio y música en el juego. No obstante, es la primera vez que te enfrentas a un reto así y careces de la formación fundamental en ello. El tiempo apremia y antes de ahogarte en un océano de ansiedad y desarrollar un

incipiente síndrome del impostor decides buscar una salida y te haces preguntas como: ¿Qué necesidades de audio y música comunes tienen casi todos los videojuegos?, ¿qué eventos necesitan música o audio en un videojuego? ¿qué debo hacer para crearlos e implementarlos en Unity?

Dar respuesta a tales preguntas dentro del supuesto práctico no sólo ayudó a profundizar en mi investigación sino también a perfilar la metodología que quería utilizar. Como dirían los goblins de *World of Warcraft*: "Time is money, friend". Cada bloque de contenido debía ser sencillo, práctico y directo, diseñado específicamente para solucionar un problema o necesidad que le pudisese surgir a ese novato en peligro, y permitiéndole desarrollar una serie de habilidades y conocimientos específicos en su carrera en el mundo de los videojuegos.

Después de un *brainstorming* que duró varios meses, empecé a esbozar una suerte de guión con distintos puntos que actuarían como unidades didácticas en las que identificaba una necesidad (o más) en la integración de música y audio en un videojuego y cómo solucionarla. A medida que indagaba más y más, aquellos puntos fueron creciendo en número y complejidad, hasta llegar a la cifra de catorce unidades didácticas o capítulos.

Cada capítulo/unidad didáctica –a su vez- constaría de la siguiente estructura: Comenzaría con una breve introducción al problema o necesidad que queremos solucionar, continuaría con las herramientas que vamos a utilizar para solucionarla, guiaría al lector en la creación de la solución y –finalmente- en la implementación en el videojuego a través de la programación.

Por ejemplo, poner sonido a las acciones del jugador. Comenzaríamos con una breve introducción al concepto de *oneshot* (nombre popular del tipo de evento empleado para sonorizar las acciones de un jugador, npc, etc.), seguido de la exposición de herramientas de FMOD que vamos a utilizar en la creación de eventos oneshot en FMOD (tipos de eventos, instrumentos, etc.) para continuar mostrando cómo crear eventos oneshot y concluyendo con la integración en Unity a través de la programación.

El horizonte ya estaba mucho más claro y el equipaje estaba listo. No obstante, como en todo proceso creativo, el camino no estuvo exento de problemas y errores. Si programadores profesionales encuentran problemas habitualmente en sus estudios bien sean de código, optimización, etc. ¿qué no iba a encontrar un compositor musical y profesor? Pero -como suele

ocurrir- de aquellos errores surgieron nuevos aprendizajes, por lo que decidí anotarlos (al igual que sus soluciones) para que tú, en caso de tropezar en las mismas piedras que yo, puedas encontrar la solución de manera rápida y efectiva a aquellos inconvenientes que te puedan surgir. Recuerda, en cualquier caso, que puede haber varias soluciones para un mismo problema. Yo mismo, mientras escribía el libro, diseñé diferentes soluciones para un mismo problema, acabando priorizando aquellas que consideré más pedagógicas.

Ya con las ideas mucho más claras y ya un buen trecho del camino recorrido notaba que faltaba algo… Tenía claros los problemas que quería solucionar, los conocimientos que quería transmitir, la metodología que quería emplear… ¿qué faltaba? ¡Por supuesto, un videojuego! ¿Cómo iba a ser una experiencia práctica sin un videojuego en el que poder testar nuestros aprendizajes?

Nace El Terror Errante

Rápidamente me puse manos a la obra y, con el guión de catorce capítulos en mano que quería desarrollar, empecé a diseñar el *documento de juego* del videojuego al que quería poner música y audio. Éste debería servir al lector a poner en práctica los conocimientos del libro de manera sencilla e intuitiva. Cada capítulo debería tener su eco en el videojuego. Recuerda que éste no es un libro sobre Unity o FMOD, es un libro sobre integración de audio y música en videojuegos, por lo que el videojuego es una pieza clave en nuestro proyecto.

Es así como, en colaboración con un programador profesional, nace *El Terror Errante*, un sencillísimo juego de tipo RPG[1] diseñado específicamente para, paso a paso, ir explorando de manera didáctica las necesidades que van surgiendo en el juego desde el punto de vista de la creación e integración de la música y audio, comenzando por lo más básico hasta procedimientos y técnicas más avanzadas. A lo largo del libro iremos construyendo, paso a paso, un sistema completo de audio para este videojuego. Cada nuevo concepto aprendido tendrá una aplicación inmediata dentro del proyecto, permitiéndote comprobar su funcionamiento en un entorno real de desarrollo.

Llegado a este punto es probable que ya te halles medio convencido de comprar este libro (o de leerlo si lo has comprado ya), siendo la duda acerca de si este libro es adecuado para ti la última

[1] *Role Play Game* (Juego de Rol).

barrera que te lo impide. La respuesta es un rotundo sí. Debido al enfoque multidimensional este libro está indicado para diferentes perfiles profesionales. Si eres un programador o creador de videojuegos que quiere adentrarse en el mundo del diseño sonoro, integración de audio o la composición musical, este libro te ofrece una guía amplia y efectiva sobre técnicas y procesos a seguir para la creación y diseño de música y audio de manera profesional y creativa. En cambio, si eres compositor, esta obra te enseñará las entrañas de la creación de videojuegos, así como valiosos trucos de diseño sonoro, la creación e integración en eventos y el código necesario para implementarlos en el juego. Si eres un diseñador de audio, este libro –además de valiosos procedimientos de procesado de audio con fines creativos- te ayudará a entender mejor la composición musical para videojuegos, así como los procesos necesarios para su integración en eventos y su implementación vía código. Finalmente, si no eres ninguno de los perfiles anteriores, pero quieres adentrarte en este maravilloso mundo de la creación de audio y música y su integración en videojuegos, éste es tu libro, ya que su naturaleza pedagógica hace que no necesites de conocimientos previos y -de manera rápida, sencilla y eficaz- desarrollarás las habilidades y conocimientos necesarios para llevar a cabo un gran número de tareas que podrían facilitarte tu integración en la creación profesional de videojuegos.

Espero que este libro no solo te enseñe a integrar audio y música en videojuegos, sino que despierte en ti la misma curiosidad que despertó en mí aquella pregunta formulada por un alumno hace ya algunos años. Si al terminar estas páginas eres capaz de mirar un videojuego desde la perspectiva de un diseñador de audio, un compositor y un desarrollador al mismo tiempo, entonces este libro habrá cumplido su propósito.

Sobre el autor

Antonio Villa es profesor de música y filosofía, compositor, diseñador de audio e intérprete, con más de veinte años de experiencia en los ámbitos de la docencia, la composición musical y la interpretación.

Su formación académica integra disciplinas musicales, pedagógicas y tecnológicas. Es Graduado en Música, Diplomado en Educación Musical, Técnico Superior en Sonido para Audiovisuales y Espectáculos, Máster en Composición Musical con Nuevas Tecnologías y Máster en Musicología Histórica. Actualmente desarrolla su investigación como doctorando en **Música y su Ciencia y Tecnología**, centrando su trabajo en la aplicación de la inteligencia artificial, la semiótica y la psicología de la emoción a la composición musical.

A lo largo de su trayectoria ha compaginado la enseñanza con la composición y el diseño de audio, manteniendo siempre un especial interés por la relación entre la música y la tecnología. Esa combinación de experiencia artística, técnica y docente le ha permitido desarrollar una metodología de aprendizaje basada en la práctica, orientada a facilitar la comprensión de conceptos complejos mediante proyectos reales.

En *Creación, diseño e integración de audio y música en videojuegos con Unity y FMOD*, Antonio reúne dos de sus grandes pasiones: la música y la programación. El resultado es una guía eminentemente práctica que acompaña al lector a lo largo de todo el flujo de trabajo del audio para videojuegos, desde la composición musical y el diseño sonoro hasta la implementación de sistemas de audio interactivo en un videojuego desarrollado específicamente con fines didácticos.

Web: www.antoniovillamusic.com

Contacto: info@antoniovillamusic.com

3.5 Conectando FMOD con Unity

Hasta ahora hemos manejado FMOD y Unity como entes separados. Es la hora de unirlos de tal manera que los eventos de audio y música que creemos en FMOD suenen en Unity.

Para ello debemos ir a: FMOD → *Edit Settings*

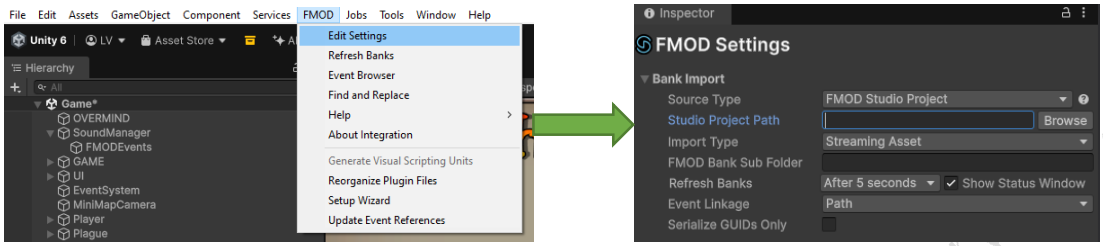

Ilustración 1. Acceso a los settings de FMOD en Unity.

Como vamos a construir el proyecto completo (*FMOD Studio Project*). En *Studio Project Path* seleccionaremos la ruta donde se halla nuestro proyecto y listo. Vamos a comprobar que, efectivamente, nuestro proyecto en Unity está conectado al de FMOD. Vamos a hacerlo a través de la herramienta de FMOD en Unity *Event Browser*.

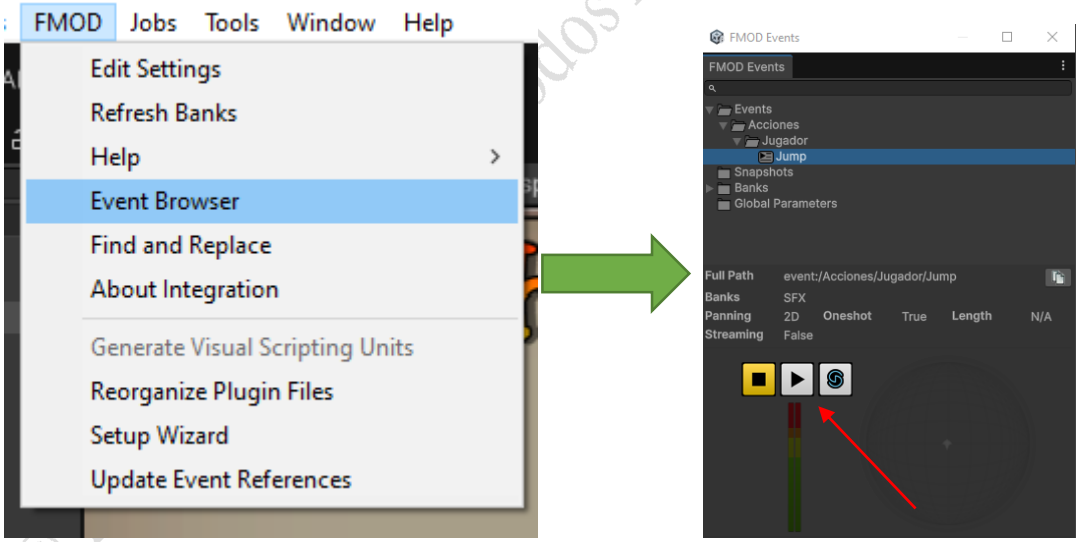

Ilustración 2. Event Browser de FMOD en Unity.

Utilizando el *browser* podemos probar los eventos que hemos ido creando en FMOD mediante su reproducción. ▶

Los *mensajes debug* a la consola ayudan a monitorear y analizar el comportamiento de la aplicación mientras se ejecuta, lo que permite detectar posibles fallos o comportamientos inesperados.

En Unity, existen varias maneras de realizar depuración, y la herramienta principal para esto es la ventana Console (Consola)[2], donde puedes visualizar mensajes, advertencias y errores. Vamos a ver los tipos más comunes:

Debug.Log()

Este es el tipo de depuración más común. Permite imprimir mensajes en la Consola para ver el estado de ciertas variables o el flujo del programa.

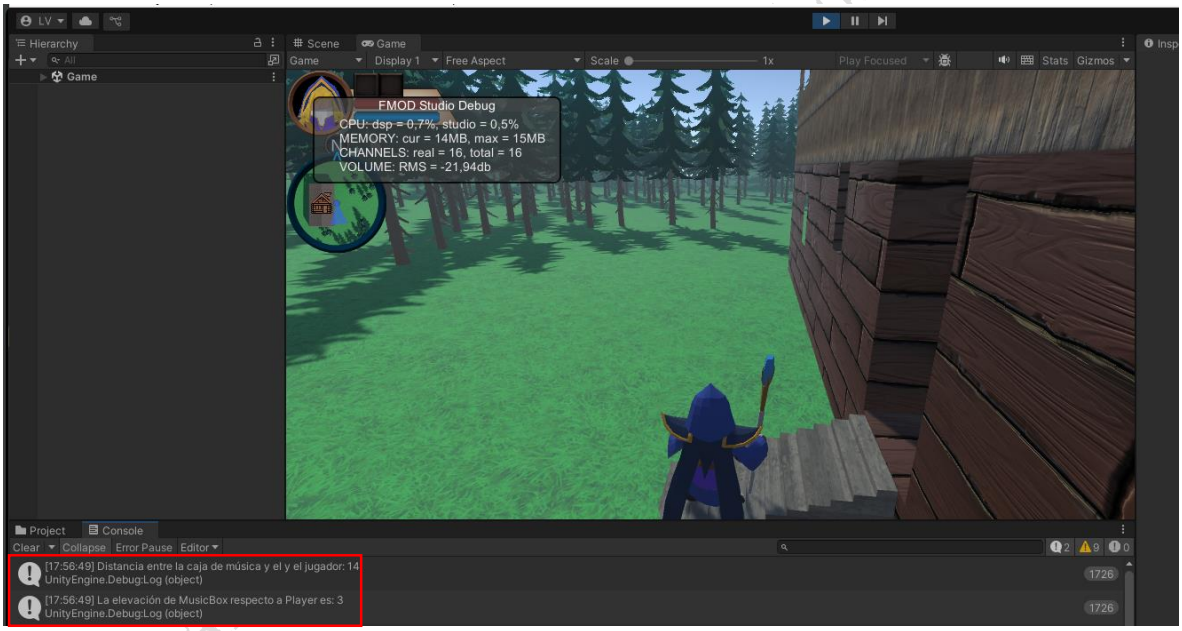

Ilustración 3. Ejemplo de mensajes de tipo Debug.Log en la consola.

Debug.LogWarning()

Similar a Debug.Log(), pero se usa para advertencias. El mensaje aparecerá en color amarillo y te indicará que algo no está mal, pero puede causar problemas más adelante.

[2] Además, Unity ofrece varios métodos y herramientas para ayudar en la depuración, como los logs, breakpoints, y la herramienta Profiler.

10. El método *PlayMusic(Emusic music)* requiere de algunas variables por lo que si no las declararmos antes, produciremos un error. Vamos a declarar la variable *Emusic* (de tipo *Enum*) y la variable *currentMusic* que se identifica con –dentro del *Enum*- la variable que esté en uso.

```
public enum EMusic { Menu, Desierto, Bosque, BossApproach, Boss, }
public EMusic currentMusic;
```

11. Ahora que ya están declaradas vamos a proceder a escribir el método. Asegúrate de que las instancias están bien referenciadas o, de lo contrario, dará un error. ***Visual Studio* suele mostrar los errores con un subrayado en *zig-zag* rojo** debajo de la palabra o carácter que no está correcto. En este ejemplo el código que detiene el ambiente del desierto está mal.


```
public void PlayMusic(EMusic music)
{
    switch (currentMusic)
    {
        case EMusic.Desierto:
            //stop area music

            break;

        case EMusic.Bosque:
            //stop area music
             break;
    }
     currentMusic = music;
    switch (music)
    {
        case EMusic.Desierto:
            //play area music

            //play area ambience
            DesiertoAmbiente.start();
            Debug.Log("Comienza el ambiente del desierto");
            break;
        case EMusic.Bosque:
            //play area music
            DesiertoAmbiente.stop(FMOD.Studio.STOP_MODE.ALLOWFADEOUT);
            Debug.Log("Se detiene el ambiente del desierto");
            //play area ambience

            break;
    }
```

Explicación del código: *LatidosUpdate()*

Este método recibe el parámetro de tipo *int IntHealth* para actualizar el evento.

El evento utiliza el método *setParameterByName* para actualizar el parámetro *Salud* con el valor *IntHealth*. Es muy importante que **el nombre del parámetro coincida con el asignado en el evento de FMOD** o, de lo contrario, no funcionará.

```
public void LatidosUpdate(int IntHealth)
{
    Latidos.setParameterByName("Salud", IntHealth);
}
```

25. Ahora que hemos creado los métodos vamos a **localizar el código que controla la salud de nuestro personaje**. Éste lo encontramos en el *script* ***Player.cs***.

26. Luego, dentro del *script Player.cs*, nos vamos al código que controla la **lógica de la salud del jugador**. Una vez localizada, **añadiremos nuestros métodos** con una lógica condicional.


```
public override void ReceiveDamage(float damage)
{
    if (Game.MusicBox.IsPickedUp && Game.Gem.IsPickedUp)
    {
        damage *= 0.25f;
    }
    if (Game.Enemy.IsEnraged)
    {
        damage *= Game.Enemy.enrageDamageModifier;
    }
    base.ReceiveDamage(damage); Game.UI.RefreshPlayerHealth(currentHealth / health);
    int IntHealthPlayer = Mathf.RoundToInt(currentHealth);
    //cambio
    if (IntHealthPlayer <= 50)
    {
        SoundManager.Instance.LatidosPlay();
        SoundManager.Instance.LatidosUpdate(IntHealthPlayer);
        //Debug.Log("La Salud es" + IntHealthPlayer);
        if (IntHealthPlayer == 0)
        {
            SoundManager.Instance.LatidosStop();
            // Detiene los latidos
        }
    }
    else
    {
        SoundManager.Instance.LatidosStop();
    }
}
```

Modificadores de daño

Conversión a entera de la varable *currentHealth* en la variable ***IntHealth***

Si la salud es menor o igual a 50, la instancia se iniciará y se actualizará. En caso contrario se detendrá con un *fade out*.

Si el jugador muere (salud = 0) la instancia se detendrá inmediatamente.

Es decir, tendríamos cuatro grupos VCA que permitirían a través de cada *fader* controlar el nivel de todos los diálogos, de todos los SFX, de todas las pistas musicales y de todos los ambientes a través de su *fader VCA* respectivo.

Nuestro menú con *sliders* permitirá modificar los niveles de nuestros grupos VCA.

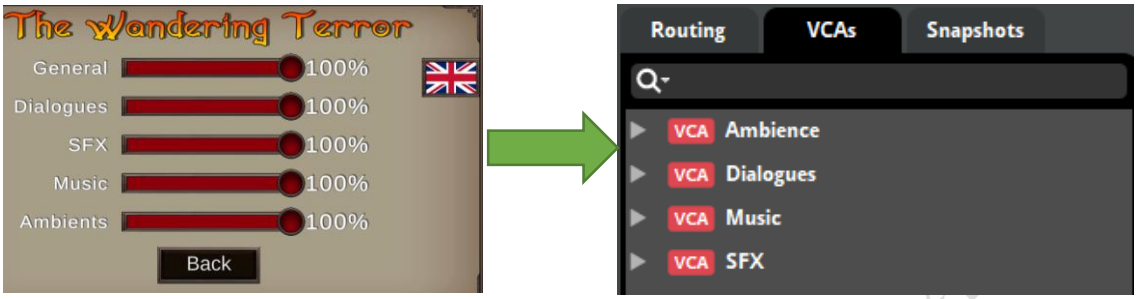

Ilustración 4. Vista del menú en Unity y de la configuración de VCA en FMOD.

1) Ahora que ya tenemos nuestro proyecto configurado, nos dirigiremos al mezclador (*mixer, Ctrl +2*) y realizaremos la mezcla en los grupos y los VCA. Vamos a dejar los VCA a 0 dB.
2) Una vez, asignados, finalmente exportaremos todos los bancos (F7).
3) Nos dirigimos a Unity y ejecutamos el juego. Al mismo tiempo, **sincronizamos** FMOD con Unity para comprobar que el mezclador funciona. Lo haremos mediante *Live Update* y seleccionando *Localhost*.

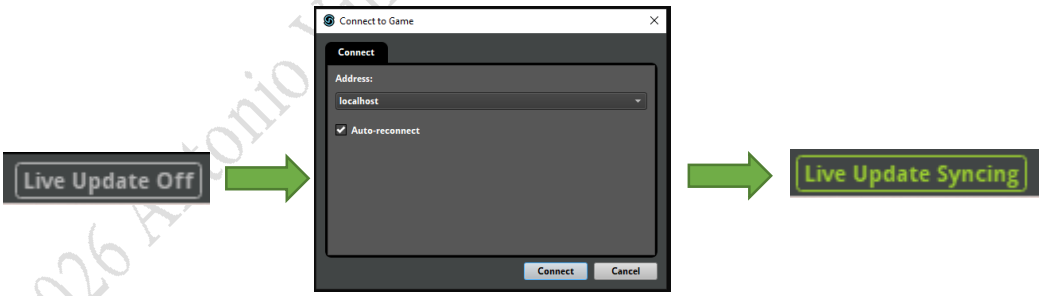

Ilustración 5. Sincronización entre FMOD y Unity.

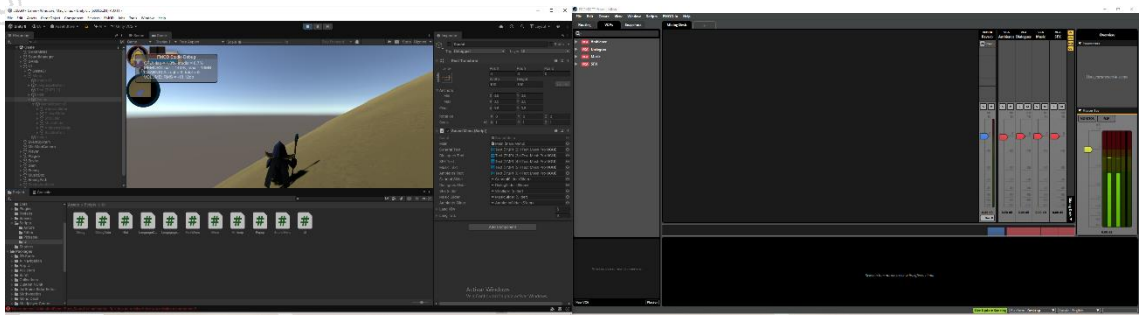

Ilustración 6. Sincronización entre FMOD y Unity.

1. Ahora, en el compás 17, añadiremos el marcador de destino final: *Death* (muerte del *Boss*).

   El aspecto debería ser similar a este:

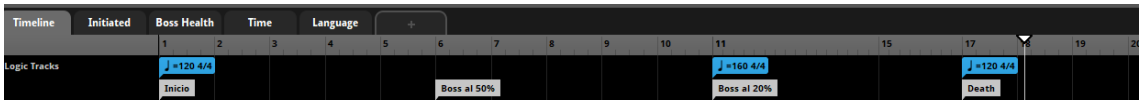

Ilustración 7. Adición de marcadores de tempo.

Recuerda que para evitar que se solapen las regiones y transiciones, podemos seleccionar el marcador o región y arrastrarlas arriba o abajo para crear diferentes niveles de lógica:

Creación de dos niveles de lógica:

Marcadores y *tempo*.

Ilustración 8. Asignación de marcadores en FMOD.

2. Una vez creados los marcadores vamos a **crear la primera región de transición** (***add transition region***). Esta región de transición la colocaremos en el segundo nivel[3] (encima de los marcadores[4] y debajo del *tempo*). La configuraremos de la siguiente manera:
   - Destino (destination): Boss at 50%.
   - El resto de las opciones las dejaremos tal y como están.
3. Para **asignar la región a la que queremos transicionar** basta con seleccionar la región y, click con el botón derecho, seleccionar la opción ***Set Destination to*** > ***Markers*** > *Seleccionar el marcador que nos interese* (en este caso *Boss at 50%*).

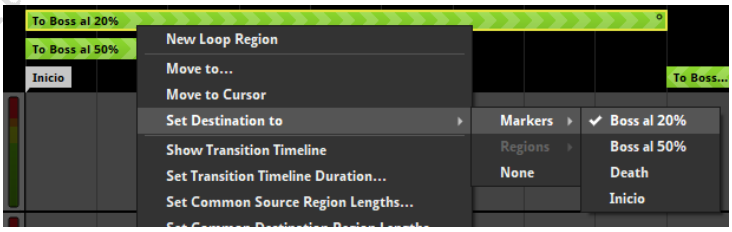

Ilustración 9. Selección del marcador al que dirigir una región de transición.

[3] Recordemos que, cuanto más altos estén los elementos en la línea temporal, mayor será su prioridad.
[4] Para hacer sitio a la región de transición entre el marcador y el tempo, basta con arrastrar ligeramente el marcador hacia abajo, con cuidado de no crear más de un nivel.

Las regiones de control actúan como el típico "portero de discoteca", el cual permite -o no- pasar al siguiente bloque. Por lo tanto, no olvides de cerrarlas sin incluir ningún archivo de audio o, de lo contrario, crearás un silencio incómodo que interrumpirá la reproducción en *loop* de la sección que se esté reproduciendo.

La lógica es simple: **Si el evento no posee las condiciones no podrá pasar de bloque y se reproducirá en bucle hasta que lo haga**.

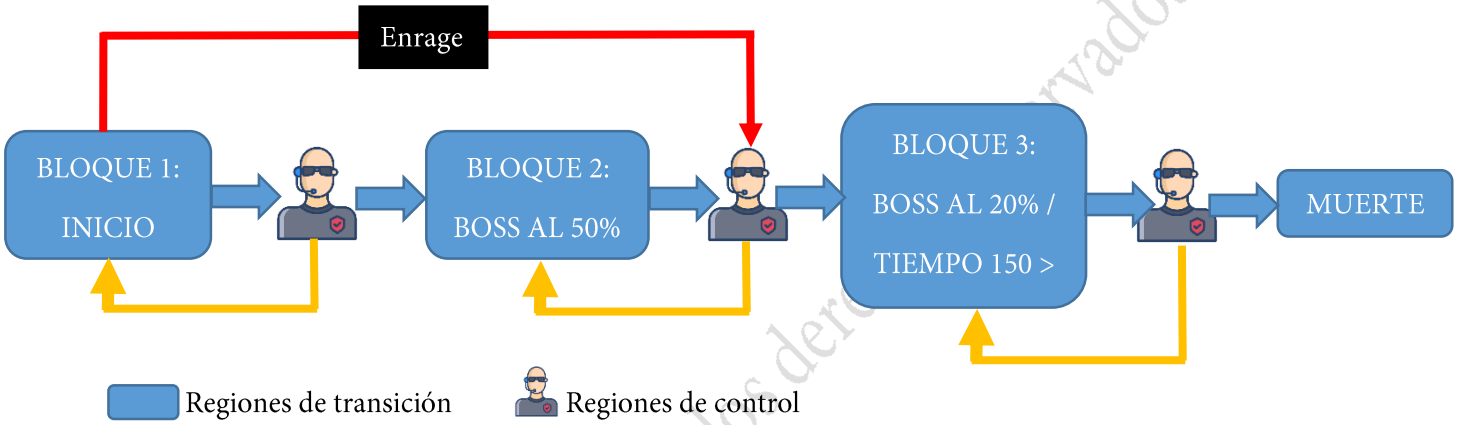

Ilustración 10. Lógica del evento BossEncounter en FMOD.

Añadiendo Transition Timelines

Recuerda que es importante prestar atención a ese pequeño círculo presente en las regiones de transición ya que hace referencia a la presencia de una *transition timeline*.

Ilustración 11. Ejemplo de región de transición.

Eso quiere decir que, en el caso de que haya alguna, FMOD leerá también esa *transition timeline*. Esto es útil si la utilizamos para añadir un *fill* entre sección y sección y que no sea tan abrupto. Pero en las regiones de control son un problema ya que añade compases extra de manera poco orgánica. Es por ello que no vamos a añadir *transition timelines* a las regiones de transición que actúen como control. Si lo haces accidentalmente puedes quitarla haciendo click derecho en ella y seleccionando *Remove Transition Timeline*.

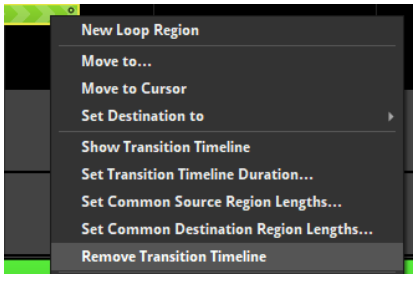

Ilustración 12. Herramienta Remove Transition Timeline.

4. Vamos a añadir las *transition timelines* en las *regiones de transición*. Y, dentro de ellas, vamos a insertar el archivo *Fill.wav*.

O directamente en el **inspector de Unity** en los campos *TalkHealthPoints* y *Spell2HealthPoints*

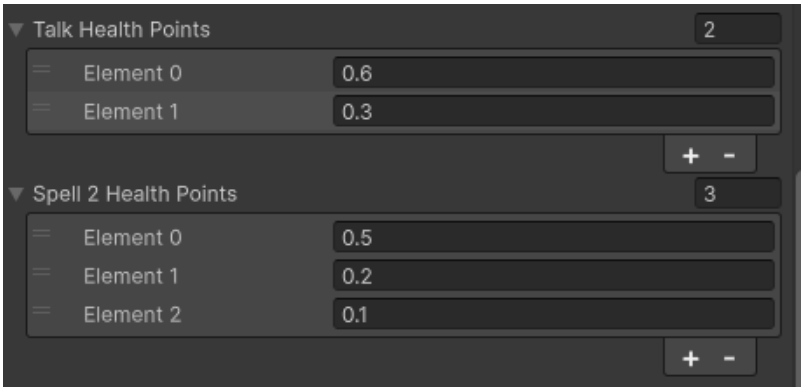

Ilustración 13. Modificación de los valores para la reproducción de voces.

Ya tenemos configuradas las voces. Vamos a insertarlas.

5. Vamos a empezar por las voces que dan inicio al encuentro con el *Boss*. Para ello vamos a dirigirnos a la hoja de parámetros *Initiated* y vamos a colocar en el valor 1 los dos archivos en su pista respectiva. Podemos arrastrar directamente los archivos de audio desde *assets* o crear *single instruments*. Una vez creados debemos añadir la condición de *Language* para que, en función del idioma seleccionado, suene uno u otro.

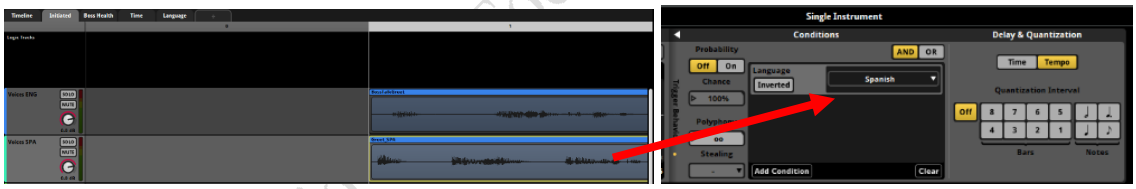

Ilustración 14. Añadiendo pistas en la hoja de parámetros "Initiated" y una condición de idioma en una pista.

6. Continuaremos en la hoja de parámetros *Boss Health* donde colocaremos los *single instruments* en los puntos de vida del *boss* anteriormente especificados.

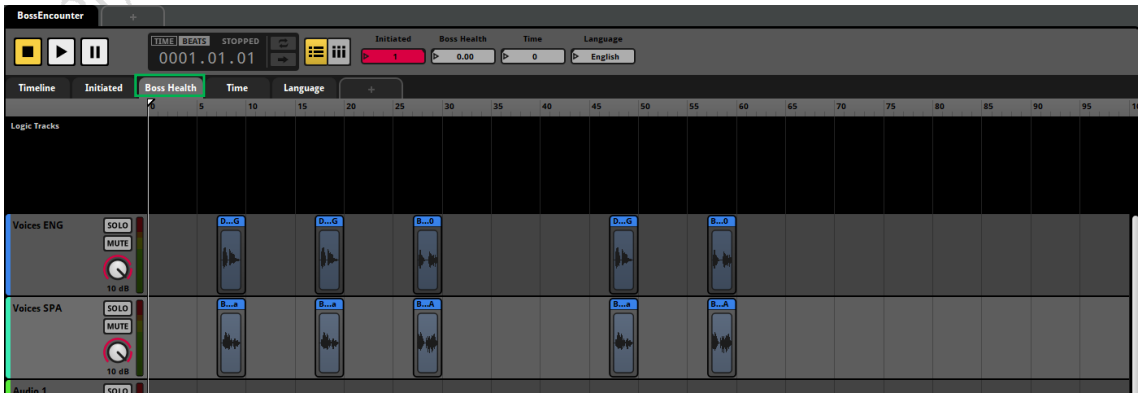

Ilustración 15.Inserción de single instruments en la hoja del parámetro Boss Health del evento BossEncounter.

Y, además, le añadiremos la condición del idioma (*Language*) para que no se reproduzcan a la vez.

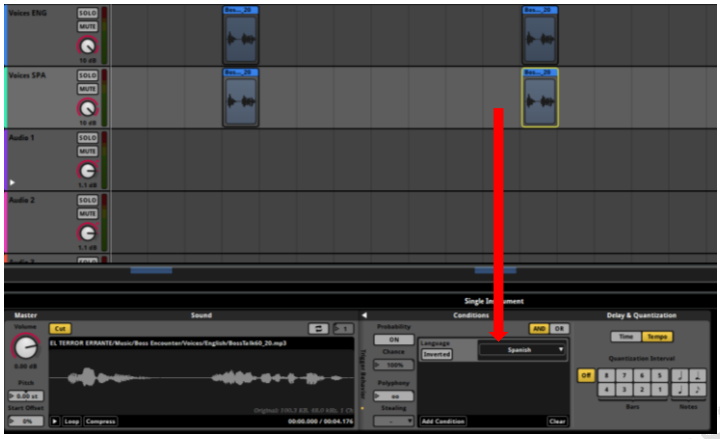

Ilustración 16. Adición de una condición de activación a través del parámetro Language.

De los demás, dos irían en la hoja de parámetros *Initiated* con los que el *boss* saludaría al jugador al inicio del combate.

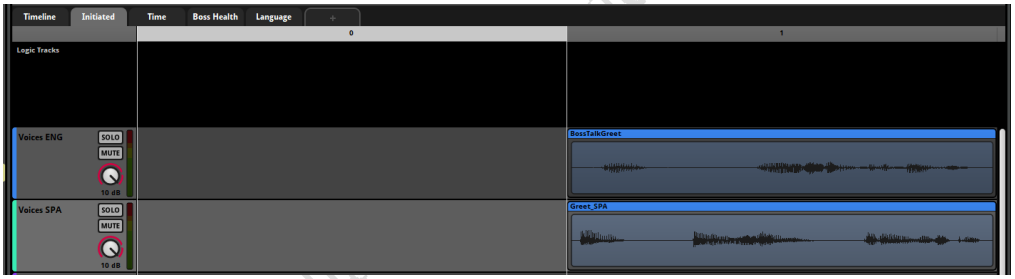

Ilustración 17. Vista de la hoja de parámetros Initiated en el evento BossEncounter.

Y el otro en la hoja de parámetros ***Time*** ya que cuando el encuentro llegue a los 150 segundos el *boss* entrará en *enrage*:

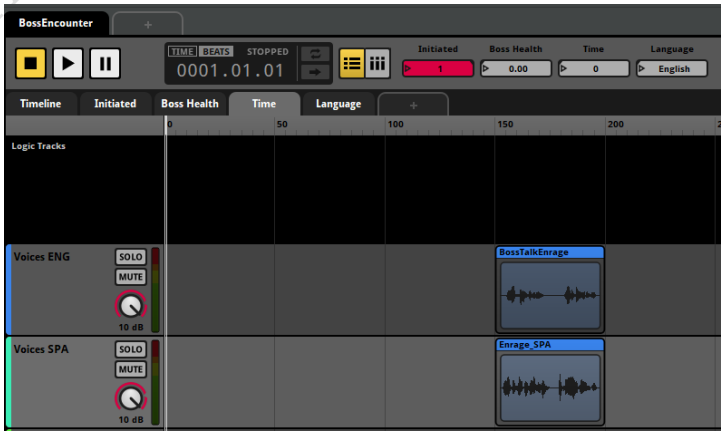

Ilustración 18. Vista de la hoja de parámetros Time en el evento BossEncounter.

DOMINA EL AUDIO Y LA MÚSICA EN VIDEOJUEGOS. DE LA IDEA A LA IMPLEMENTACIÓN.

Esta guía completa te lleva desde los fundamentos del sonido y la música hasta la integración de sistemas avanzados en Unity utilizando FMOD.

A través de explicaciones claras, ejemplos prácticos y un proyecto completo, aprenderás a diseñar música interactiva, implementar audio dinámico, trabajar con diálogos y optimizar el sonido de tu juego para lograr resultados profesionales.

- Composición musical e interacción
- FMOD: de lo básico a la implementación avanzada
- Integración completa con Unity y C#
- Diálogos, localización y soporte multilenguaje
- Optimización, perfilado y rendimiento

INCLUYE UN PROYECTO COMPLETO DE VIDEOJUEGO DESCARGABLE

EL TERROR ERRANTE

Explora un proyecto completo en Unity (The Wandering Terror) y aprende haciendo.

Modifica cada sistema de audio, experimenta y observa los resultados dentro del juego.

ANTONIO VILLA

Compositor, diseñador de sonido y docente con más de 20 años de experiencia en música, audio y educación.

Apasionado por el audio en videojuegos y la música interactiva, combina el rigor académico con la experiencia práctica para ayudarte a convertir tus ideas en paisajes sonoros inmersivos.

DESCARGA EL PROYECTO | PRÁCTICAS EN UNITY | INTEGRACIÓN CON FMOD | TÉCNICAS PROFESIONALES

E-BOOK

PDF • EPUB