Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

Sonido 3D mediante realidad aumentada en interpretación polifónica

Alumno: **Juan Carlos Muñoz Castellano**

Tutor: Prof. D. Raúl Mata Campos
Depto.: Ingeniería de Telecomunicación

Septiembre, 2021

The mind is like a muscle.

And if you want to be really powerful you have to work it out.

Each new fact brings you another choice.

Each new idea build another muscle.

And those muscles are going to make you really strong, those are your weapons.

AGRADECIMIENTOS

A mi familia, mis padres y mi hermana,

que siempre me han apoyado en mis estudios

y en cada una de las decisiones que he tomado.

Os estaré eternamente agradecido.

A mis amigos y mi novia,

por tantas risas y por mostrarme

que era capaz de conseguir

todo lo que me propusiera.

A los que ya no están.

Sé que estarían muy orgullosos de mí.

A los compañeros que he conocido en este viaje

y a los profesores que me han motivado

a seguir aprendiendo cada día

y a no poner límites a la creatividad.

RESUMEN

Del mismo modo que una persona sabe, con los ojos cerrados, si está tumbada o está de pie, esa misma persona sabe de dónde viene un sonido incluso con los ojos cerrados. Nuestro cerebro realiza complejos cálculos matemáticos usando química para transformar señales en el sistema auditivo y determinar la posición de un altavoz. En otras palabras, nuestro cerebro escucha en 3D, por lo que sería lógico tener la capacidad de reproducir esa sensación auditiva cuando se desee a través de, por ejemplo, unos auriculares.

En este proyecto, se combinan técnicas de sonido 3D y realidad aumentada (AR) para la realización de una aplicación, válida tanto para smartphone como para PC, capaz de crear esta sensación espacial de sonido 3D.

El funcionamiento de esta app se basa en el reconocimiento espacial de marcadores u objetos a través de una cámara AR, que se corresponderán con ciertos instrumentos musicales, y que emitirán una fuente sonora, dependiendo de la localización u origen de los mismos, para poder crear esa sensación de espacialidad o sonido 3D mediante el uso de auriculares. Además de invocar la fuente sonora tras la detección del marcador, se reproducirá un objeto 3D sobre el mismo para identificar con más claridad que instrumento está sonando.

Como resultado, esta aplicación será capaz de reproducir partituras completas con sonido 3D, creando un entorno interactivo que sea atractivo, tanto para el usuario que busque entretenimiento, como para el uso de la aplicación con fines comerciales y/o artísticos. Se presenta con una interfaz sencilla e intuitiva para facilitar el manejo de la misma.

ABSTRACT

Just as a person knows, with their eyes closed, whether they are lying down or standing up, that same person knows where a sound is coming from even with their eyes closed. Our brain performs complex mathematical calculations using chemistry to transform signals in the auditory system and determine the position of a speaker. In other words, our brain listens in 3D, so it would be logical to have the ability to reproduce that hearing sensation whenever desired through, for example, headphones.

In this project, 3D sound techniques and augmented reality (AR) are combined to create an application, valid for both smartphones and PCs, capable of creating this spatial sensation of 3D sound.

The operation of this app is based on the spatial recognition of markers or objects through an AR camera, which will correspond to certain musical instruments, and that will emit a sound source, depending on the location or origin of the same, in order to create that sense of spatiality or 3D sound through the use of headphones. In addition to invoking the sound source after the marker is detected, a 3D object will be played over it to identify more clearly which instrument is playing.

As a result, this application will be able to reproduce complete scores with 3D sound, creating an interactive environment that is attractive, both for the user looking for entertainment, and for the use of the application for commercial and/or artistic purposes. It is presented with a simple and intuitive interface to make it easy to use.

ÍNDICE DE FIGURAS E ILUSTRACIONES

ÍNDICE

1. INTRODUCCIÓN

1.1. SONIDO 3D

Usando altavoces estéreo de alta calidad, la fuente de sonido se puede ubicar con precisión, pero siempre está frente a nuestros ojos. Este atributo se llama escena sonora. El sonido tridimensional emitido por los auriculares también puede localizar la fuente de sonido detrás y encima de nosotros. A diferencia del estéreo, donde el sonido que se reproduce va de izquierda a derecha y viceversa, el sonido binaural agrega altura y permite un sonido que parece provenir de múltiples direcciones. Queda preguntarse cómo funciona y qué técnicas se emplean, lo cual se explica básicamente atendiendo a la distancia entre las orejas, es por ello que el sonido 3D se capta esencialmente con auriculares.

El uso del mismo no se ha extendido más rápidamente por el hecho de tener que utilizarlo con auriculares. Cuando se reproduce música, siempre se busca que el sonido suene con la mayor fidelidad posible en cualquier dispositivo, pero para el caso del sonido 3D, si no se utilizan cascos, sonaría distinto. La razón de esto es que lo que hace el audio 3D es modificar diferentes parámetros, como la frecuencia o el tiempo de espera. En auriculares puedes ajustar bien la posición, pero con altavoces es más complicado.

Para lograr este efecto, se intenta “confundir” al cerebro con el fin de que sitúe en el espacio ciertas frecuencias. En otras palabras, se copia el espectro de frecuencias y otros parámetros para modificar el archivo de audio, y así poder percibir en los oídos las fuentes de sonido como se desee, aplicable, por ejemplo, con los instrumentos.

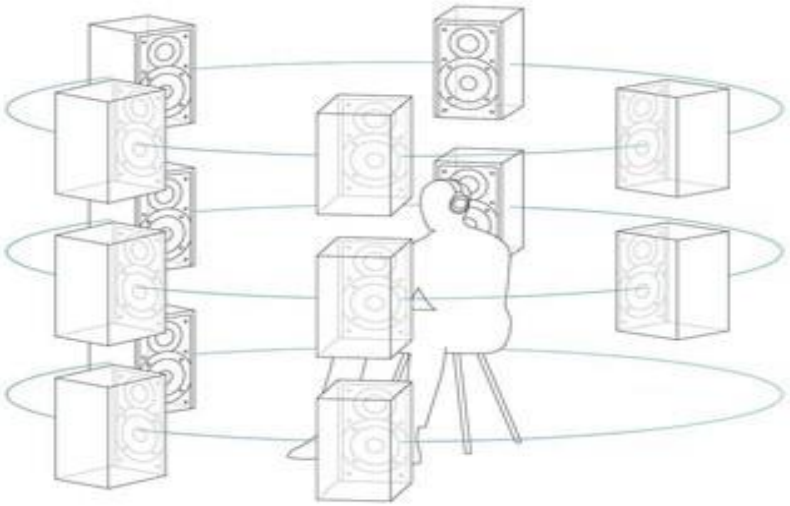


Ilustración 1: Ejemplo de percepción de audio 3D a través de auriculares

El término ‘binaural’ proviene del hecho de utilizar dos pabellones auditivos, que son lo que son las orejas. El cerebro está preparado para interpretar estos sonidos a la perfección y reconoce de donde proviene el sonido con bastante precisión.

Este es un comportamiento inconsciente, pero se basa en tres parámetros: desfase temporal, cambios en el nivel de presión sonora (SPL) y cambios en la frecuencia. Con estos datos, el cerebro puede identificar la ubicación de la fuente de sonido. HRTF (‘Head-related transfer function’) se define como esta respuesta, que caracteriza cómo el oído recibe el sonido desde un cierto punto en el espacio.

El cerebro interpreta que las fuentes de sonido están en diferentes localizaciones con el sonido 3D, aunque lo que de verdad está pasando es que éste ha sido configurado con los parámetros anteriores para hacerle creer que así es. Gracias a esto, se consigue crear una experiencia en 360º totalmente inmersiva, imposible de comparar siquiera con los resultados que puede ofrecer el 5.1 (‘sistemas de audio multicanales con seis canales de sonido surround’).

A través de algoritmos, se puede lograr que la persona piense que el sonido proviene de su alrededor, ateniéndonos al caso de la música, se escucharía como si estuviesen tocando alrededor de dicha persona, por lo que se consigue una sensación hiper realista y una experiencia envolvente. Y esto es lo que buscan ciertamente los fabricantes con la experiencia de sonido 3D, crear ese efecto llamativo para los usuarios que busquen un sonido más sofisticado.

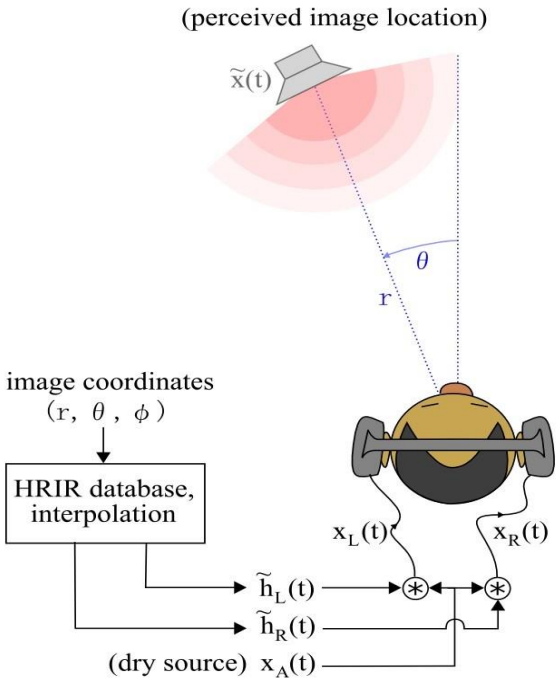


Ilustración 2: Aplicación HRIR para "engañar" al cerebro sobre la posición del sonido

Fuente: https://upload.wikimedia.org/wikipedia/commons/4/42/Hrir_binaural_synthesis.png

1.2. REALIDAD AUMENTADA (RA)/AUGMENTED REALITY (AR)

La realidad aumentada (RA) es una de las mayores tendencias tecnológicas en este momento, e irá creciendo a medida que los teléfonos inteligentes preparados para RA y otros dispositivos sean más accesibles en todo el mundo. La RA permite ver el entorno de la vida real justo delante del usuario - árboles que se balancean en el parque, perros que persiguen pelotas, niños que juegan al fútbol - con un aumento digital superpuesto. Por ejemplo, un pterodáctilo podría ser visto aterrizando en los árboles, los perros podrían estar mezclándose con sus homólogos de dibujos animados, y los niños podrían ser vistos pateando una nave espacial alienígena en su camino para marcar un gol.

Con los avances en la tecnología de RA, estos ejemplos no son tan diferentes de lo que ya podría estar disponible para cualquier smartphone. De hecho, la realidad aumentada está disponible y se utiliza de muchas maneras, como, por ejemplo, como lentes de Snapchat, en aplicaciones que te ayudan a encontrar tu coche en un aparcamiento lleno de gente y en una variedad de aplicaciones de compras que te permiten probarte la ropa sin siquiera salir de casa.

Tal vez el ejemplo más famoso de la tecnología de RA sea la aplicación móvil Pokemon Go, que se lanzó en 2016 y se convirtió rápidamente en una sensación ineludible. En el juego, los jugadores localizan y capturan los personajes Pokemon que aparecen en el mundo real, en la acera, en una fuente, incluso en el propio baño.

Ilustración 3: Ejemplo cámara AR Pokemon GO

1.3. LA TECNOLOGÍA RV/RA

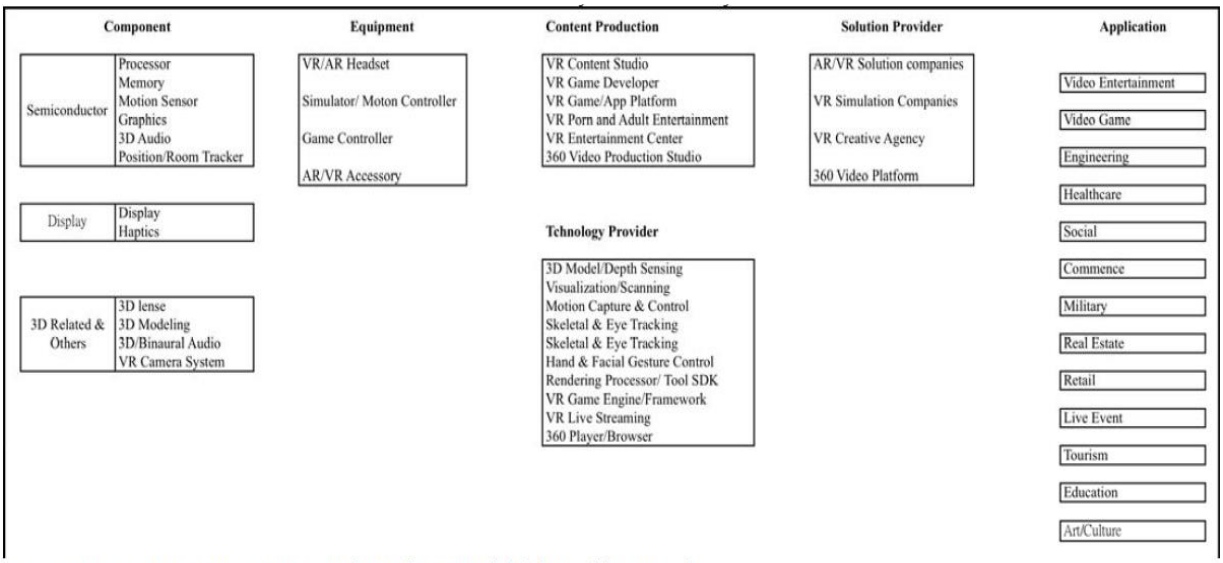
La tecnología RV/RA (Realidad Virtual/Realidad Aumentada) tiene el potencial de cambiar los modelos de negocio y las formas en las que se realizan las transacciones. Mientras que el concepto de realidad virtual ha sido discutido desde los años 90, la tecnología de entonces no alcanzó a la imaginación de las personas. Sin embargo, en estos días, la era de la RV/AR está llegando con potentes ordenadores y smartphones, y un rendimiento significativo de las pantallas y sensores.

Empresas tecnológicas clave, como Google, Facebook, Microsoft, HTC, Samsung y Apple han estado participando en el desarrollo del mundo de la realidad virtual que las personas han estado imaginando (Bellini et al., 2016; Okazaki, et al.,2015). La experiencia del usuario es el factor más importante para ampliar los modelos comerciales aplicables y la omnipresencia de la RV/RA.

Se puede observar que la RA tiene obstáculos más significativos que superar, incluyendo desafíos en la tecnología de visualización y el procesamiento y calibración en tiempo real del entorno físico del mundo real. Dicho esto, a medida que la tecnología de RA madura, también se puede observar que están surgiendo aplicaciones empresariales más fuertes, especialmente teniendo en cuenta que la RA le permite ver su entorno físico, mientras que la RV lo bloquea completamente.

Esta tecnología ha creado un sinfín de posibilidades en el mercado actual y un ecosistema propio con un volumen de negocio, en el caso de RA, que está en camino de generar más de 30.000 millones de dólares en ingresos en 2021, el doble que RV.

Tabla 1: Dinámica del modelo de negocio de la industria VR/AR



Source: VR & AR Ecomap System and Database Worldwide; Goldman Sachs

1.4. UNITY/VUFORIA

Existen una gran cantidad de alternativas para realizar un proyecto de estas características. Hay empresas bastante fuertes en el sector que intentan hacerse con su cuota de mercado en lo que a desarrollo de aplicaciones o juegos de realidad aumentada se refiere.

Aunque existen alternativas con gran popularidad como *Unreal Engine* (motor de juegos creado por Epic Games), en este proyecto, se ha optado por utilizar la plataforma de desarrollo Unity, ya que presenta unas características más que suficientes para el desarrollo del mismo, ofreciendo un comportamiento adecuado y de prestigio en el campo de desarrollo de videojuegos.

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies. Está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS y Linux, y tiene soporte de compilación con diferentes tipos de plataformas como Android. Además, posee una 'tienda de recursos' denominada Assets Store, en la que los desarrolladores incluyen una inmensidad de recursos de todo tipo (gratuitos y de pago), que se pueden incorporar a nuestro proyecto. Estos recursos incluyen modelos 2D y 3D, música y efectos de sonido, texturas, materiales, paquetes de scripts y multitud de extensiones.

Por otro lado, Vuforia es una plataforma o SDK (kit de desarrollo de software) creada para que desarrolladores puedan integrar fácilmente tecnologías de realidad aumentada a sus aplicaciones, de manera que cualquiera pueda hacer uso de ella. Es una de las herramientas más sólidas que existen para integrar realidad aumentada y se puede incluir de manera sencilla en Unity en forma de paquete o package, por lo que se consideró la tecnología apropiada para el desarrollo de este proyecto.

Vuforia utiliza la pantalla de la cámara del dispositivo combinada con datos del acelerómetro y del giroscopio para examinar el mundo, entender lo que 've' y crear un modelo de entorno. Existen múltiples posibilidades entre las que destacan el reconocimiento de targets o marcadores y el reconocimiento del terreno del mundo, que permite al usuario escanear su ambiente y realizar procesos para interpretar el mundo y crear una visión en el ordenador o smartphone en 3D del mundo real.

2. OBJETIVOS

1. Adquirir nuevos conocimientos en técnicas avanzadas de procesamiento de imagen/video y audio, así como de realidad aumentada AR. En concreto, en este trabajo se pretende desarrollar una herramienta para la recreación de sonidos polifónicos 3D a partir de la identificación y análisis de distribución de marcadores para realidad aumentada.

2. Diseñar un entorno de realidad aumentada, así como desarrollar una aplicación capaz de reproducir diferentes interpretaciones adaptadas a la posición virtualizada que se haga de los mismos, siendo capaz de realizar esta reproducción de manera sincronizada.

3. Explorar las diferentes posibilidades para el desarrollo y posterior mejora de la aplicación, realizando un estudio exhaustivo de las herramientas actuales para implementar realidad aumentada y sonido espacial, con el fin de elegir la mejor opción posible.

4. Estudio de la situación actual del mercado con respecto a la realidad aumentada y el sonido 3D, y de las aplicaciones más innovadores en este sector, así como analizarlas para entender el funcionamiento y poder aplicar técnicas diferentes a la aplicación.

5. Facilitar la compresión de la aplicación para usuarios no experimentados en el entorno de realidad aumentada y sonidos polifónicos. Para ello, se presentará una aplicación que se pueda manejar de manera sencilla, y que facilite la incorporación de nuevas partituras e instrumentos musicales, ofreciendo una solución versátil y cómoda.

6. Deberá proporcionar un interfaz cómodo, intuitivo y sencillo de utilizar para facilitar su uso con finalidad de un producto cerrado respecto a su utilización. Se pretende que sea visualmente atractiva para el usuario. Para cumplir esta finalidad, se incluirá un menú simple y expresivo al principio.

7. Comprobar el correcto funcionamiento de la aplicación, tras finalizar la implementación de la misma, para cumplir con los objetivos marcados en la propuesta inicial.

3. METODOLOGÍA

3.1. ESTADO DEL ARTE

Antes de comenzar un diseño en cualquier ámbito, más aún si cabe en el tecnológico, la metodología a seguir en cualquier proyecto que se quiera lograr con éxito tiene su comienzo observando que es lo que ya está en el mercado o que se está desarrollando paralelamente a dicha idea, es decir, se debe observar la magnitud y alcance de proyectos similares a los propios si no se quiere fracasar en el mismo.

En esta era de revolución tecnológica, se está sometido a constantes cambios debido a la vertiginosa rapidez con la que evoluciona la tecnología, en parte a causa de la globalización en la que se vive en estos días. Se dice, incluso, que el mundo se está adentrando en la cuarta revolución industrial, gracias al avance de campos como la inteligencia artificial, el aprendizaje automático o los ordenadores cuánticos. Es por eso que resulta imprescindible conocer las herramientas más actuales que se disponen para el desarrollo de cualquier proyecto.

Aunque la innovación de este proyecto se centra principalmente en la creación de una app que reproduzca sonido 3D, se hablará también de lo último que se puede encontrar en aplicaciones de realidad aumentada y se expondrán los ejemplos más recientes, tanto de la tecnología 3D y RA por separada, como de aplicaciones en su conjunto tal y como en lo que se basa la realización de este proyecto.

3.1.1. Tecnología sonido 8D

El sonido 8D, como se le conoce como nombre comercial, no es más que el uso de la tecnología binaural o sonido 3D de manera sofisticada. De hecho, es exactamente lo mismo, solamente que se ha viralizado con el nombre de sonido 8D porque suena más atractivo.

La realidad es que no hay una nueva tecnología, sino una técnica que permite oír los sonidos como si estuvieras en medio de ella. Esa sensación de música que pasa por tu cabeza, gracias al uso de los auriculares, es el resultado de las técnicas de ecualización, panorámica y efectos combinados.

El resultado es el llamado audio 8D (que no tiene sentido ya que el espacio físico donde vivimos está mapeado en 3 dimensiones), y es lo que te permite apreciar que los sonidos provienen de diferentes direcciones. Se puede encontrar en YouTube o Soundcloud este tipo de grabaciones de audio desde la calidad más baja, donde la grabación simplemente se mueve de izquierda a derecha, hasta grabaciones de alta calidad, donde el sonido se

localiza correctamente en una región específica del espacio tridimensional y se puede escuchar claramente a través de unos auriculares. De hecho, el resultado es espectacular en algunas ocasiones, y dejan al espectador boquiabierto, como es el caso de los siguientes enlaces (uso obligatorio de auriculares):

https://www.youtube.com/watch?v=IUDTlvagjJA&ab_channel=LovelyVirus

https://www.youtube.com/watch?v=Gi-WJHsIoKE&ab_channel=Sletter

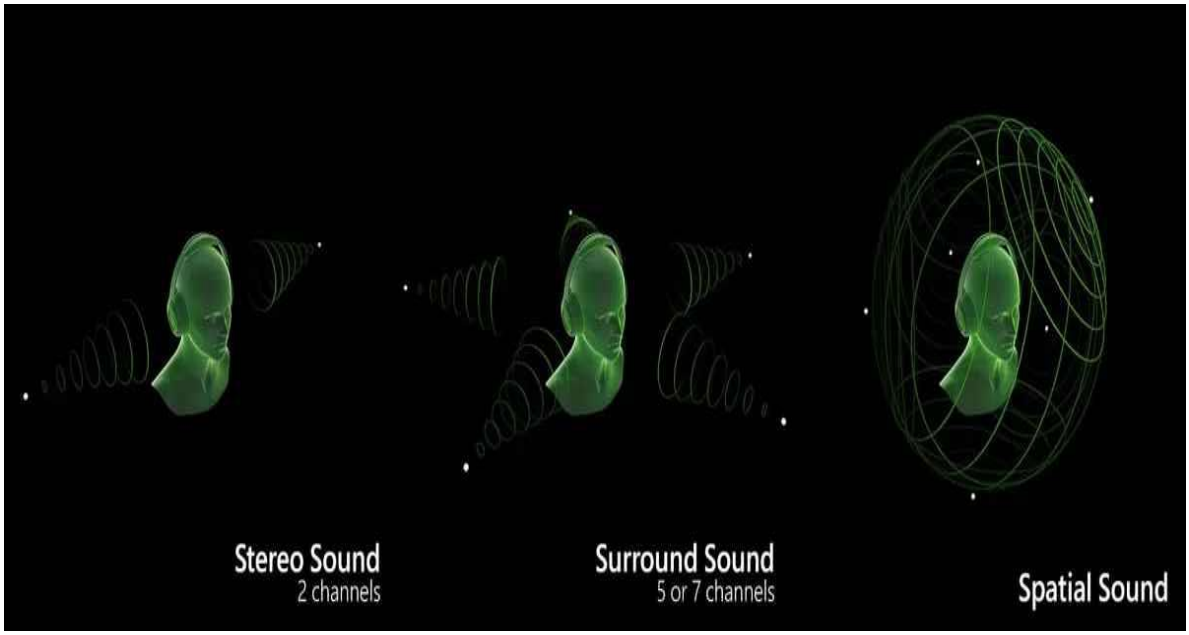


Ilustración 4: Visualización de cómo percibimos las diferentes tecnologías de sonido

En cuanto a la aplicación de esta tecnología más allá de un audio que 'suene bien', se pueden encontrar una inmensa cantidad de empresas que utilizan esta tecnología para el desarrollo de sus aplicaciones, como, por ejemplo, Ubisoft para crear el sonido envolvente en sus videojuegos, o numerosos artistas que la usan para crear canciones que contengan un sonido espacial. Empresas como Audioalter permiten transformar cualquier tipo de audio en 3D.

En el caso particular para este proyecto, el entorno de desarrollo Unity también ha dado facilidades a través de funciones que contiene de manera nativa, para crear el sonido 3D en cualquier pista de audio que se añada previamente, y, aun siendo una plataforma no recomendada para principiantes, se puede lograr fielmente el objetivo del proyecto a través del uso de Unity, es por eso, que finalmente se decidió desarrollar la aplicación utilizando su motor.

3.1.2. Aplicaciones recientes que se basan o utilizan Realidad Aumentada.

a) **Tik Tok**: Esta famosa red social implementa una serie de efectos de RA para la creación de vídeos que se han hecho tendencia en varias ocasiones, como es el caso del efecto pincel, que permite dibujar lo que se quiera sobre la cámara del móvil, quedando este dibujo grabado en la pantalla de forma tridimensional en el espacio, de manera que al moverse sobre él se puede ir viendo el objeto 3D creado.

Ilustración 5: Derecha, ejemplo efecto pincel de la app Tik Tok. Izquierda, logo de Tik Tok

b) **Skyview:** Esta app, imprescindible para los amantes del espacio, sean astrónomos o simples espectadores permite con solo mirar al cielo a través del smartphone poder ver como los puntos en el cielo se convierten en constelaciones, planetas y satélites. A través de la tecnología RA consigue lograr una experiencia única en el usuario.

Ilustración 6: Ejemplo de uso de Skyview

c) **Google**: Conocida por todo el mundo, la principal filial de la empresa multinacional Alphabet, ha desarrollado un sinfín de productos implementando RA, como es el caso de su traductor (Google Translate), cámara (Google Lens) e incluso su buscador. Permite traducir texto en cualquier idioma o conseguir información en la cámara solamente enfocando el texto u objeto. Incluso recientemente en este verano, ha desarrollado prototipos 3D para colocar dinosaurios en escala real en el entorno que se esté observando a través de la pantalla. Es sin duda, una de las grandes promotoras de la tecnología RA en estos días.

Ilustración 7: Dinosaurios en tu entorno con la RA de Google

3.1.3. Aplicaciones musicales que implementan realidad aumentada y/o sonido 3D

1. **Firstage**: es una aplicación que hace uso de la realidad aumentada para disfrutar de conciertos pregrabados en vivo. Sus creadores tuvieron la idea al observar la pobre calidad musical de algunas ciudades, y esto les hizo plantearse cómo podría un fan consumir conciertos en directo si vive en uno de estos lugares. Con esta aplicación, desde la pantalla de un móvil se podría ver a cualquier artista sin importar el lugar donde se encuentre.

Ilustración 8: Ejemplo de uso app Firstage

2. **SoundStage VR**: En este caso, se habla de una app de realidad virtual, pero cabe la pena mencionarla ya que es parecida a lo que se pretende lograr con este proyecto. Esta app permite crear un estudio musical en cualquier lugar a través de unas gafas de realidad virtual, permitiendo incorporar teclado, altavoces, mesa de mezclas y un sinfín de instrumentos musicales. Como resultado, se tendría a disposición del usuario un estudio musical sin tener que gastar demasiado, con un sonido binaural, y con la única necesidad de utilizar ciertos gadgets para manejarla, además de gafas VR y auriculares.

Ilustración 9: Modelo de uso de la app SoundStage VR

3. **Sleep Orbit**: Esta aplicación, dedicada a combatir el estrés, problemas de insomnio o simplemente a relajarnos, destaca sobre las demás en este ámbito gracias al uso de la tecnología de sonido 3D. Permite crear propias mezclas de sonidos naturales con un más que logrado sonido binaural. La app permite configurar cada uno de los sonidos para dejarlos estáticos en una posición concreta o configurarla para que "orbiten" alrededor del usuario de manera controlada o aleatoria. Es fácil identificar de qué parte nos llega el sonido, como si la fuente original se tuviese detrás, en frente o a cada uno de los lados de la cabeza.

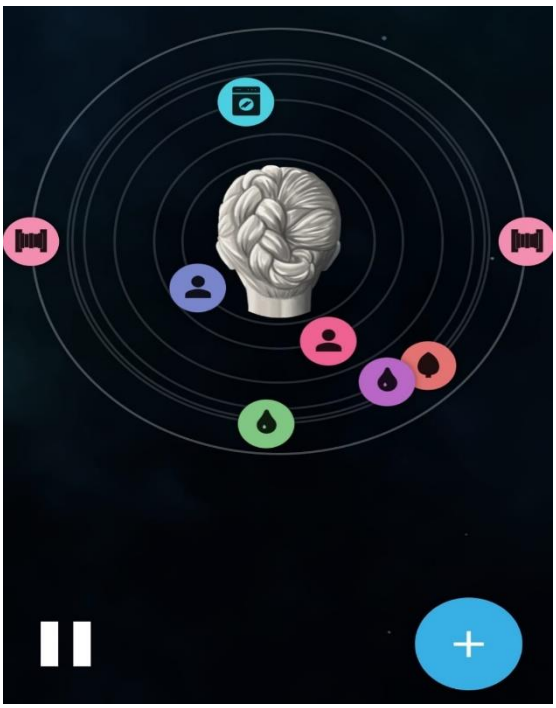


Ilustración 10: Captura de uso de la aplicación Sleep Orbit

3.2. METODOLOGÍA ORQUESTA RA CON SONIDO 3D

La metodología que se ha seguido para la realización de este proyecto ha sido probando diferentes posibilidades de configuración, de opciones a incluir o no dependiendo de la utilidad que pudieran tener en el proyecto y también, a base de *prueba y error*. La mayoría de horas han sido invertidas en búsqueda de información acerca del mismo para encontrar una solución final que fuese práctica, intuitiva y con capacidad de amoldarse para poder continuar en el futuro con líneas de investigación semejantes.

Finalmente, se ha apostado por utilizar la plataforma de desarrollo Unity, como base para crear la aplicación, debido a la gran base de datos, librerías e información que se dispone acerca de la misma. En Unity, se pueden integrar con facilidad SDKs (Software Development Kit/ Kit de desarrollo de software) tales como Vuforia, que permite la creación de aplicaciones de realidad aumentada en dispositivos móviles, y, Android SDK, que nos ha permitido configurar todos los parámetros necesarios para que la aplicación sea funcional en este dispositivo.

Sin tener conocimientos previos sobre Unity o Vuforia, y dada la lenta curva de aprendizaje de los mismos, fue difícil trazar una ruta que fuese capaz de resolver todos los objetivos que se planteaban en el proyecto. Si bien es cierto que se dispone de mucha información y tutoriales sobre el uso de estas plataformas, dicha información se encuentra bastante anticuada y la mayoría de funcionalidades han cambiado o se han integrado de manera diferente en el programa. Es por eso, que, una vez conseguido el proyecto, podría repetirse y rehacerlo de cero en cuestión de horas, pero el progreso de aprendizaje hasta llegar y conseguir el resultado final ha sido intenso.

El desarrollo de este proyecto se constituye por 5 bloques principales, los cuáles se detallarán en este capítulo. Para tener en mente y poder visualizar el proyecto con mayor agilidad, se presenta un diagrama de flujo que se compone de estos principales módulos que integran el resultado final, es decir, la apk para Android. Estos bloques son los siguientes:

1. *Creación y estructura de la escena en Unity:* Este bloque, en rojo, desarrollará todo lo relativo a la programación necesaria para la creación de la aplicación en Unity, a excepción de la configuración del sonido 3D, incluida en el bloque 3.
2. *Procesamiento del audio:* Este bloque, en azul, se encargará de todos los preparativos previos para introducir el audio en la escena.
3. *Configuración del sonido 3D:* Este bloque, en amarillo, se encarga como su nombre indica de todo lo relativo a la configuración del sonido 3D.

4. *Diseño del menú*: Este bloque, en morado, indicará los pasos a seguir para la configuración del menú en la escena, y, posteriormente, en la aplicación.
5. *Construcción APK para Android:* Finalmente, este bloque, en negro, indica los pasos finales para la construcción de la APK.

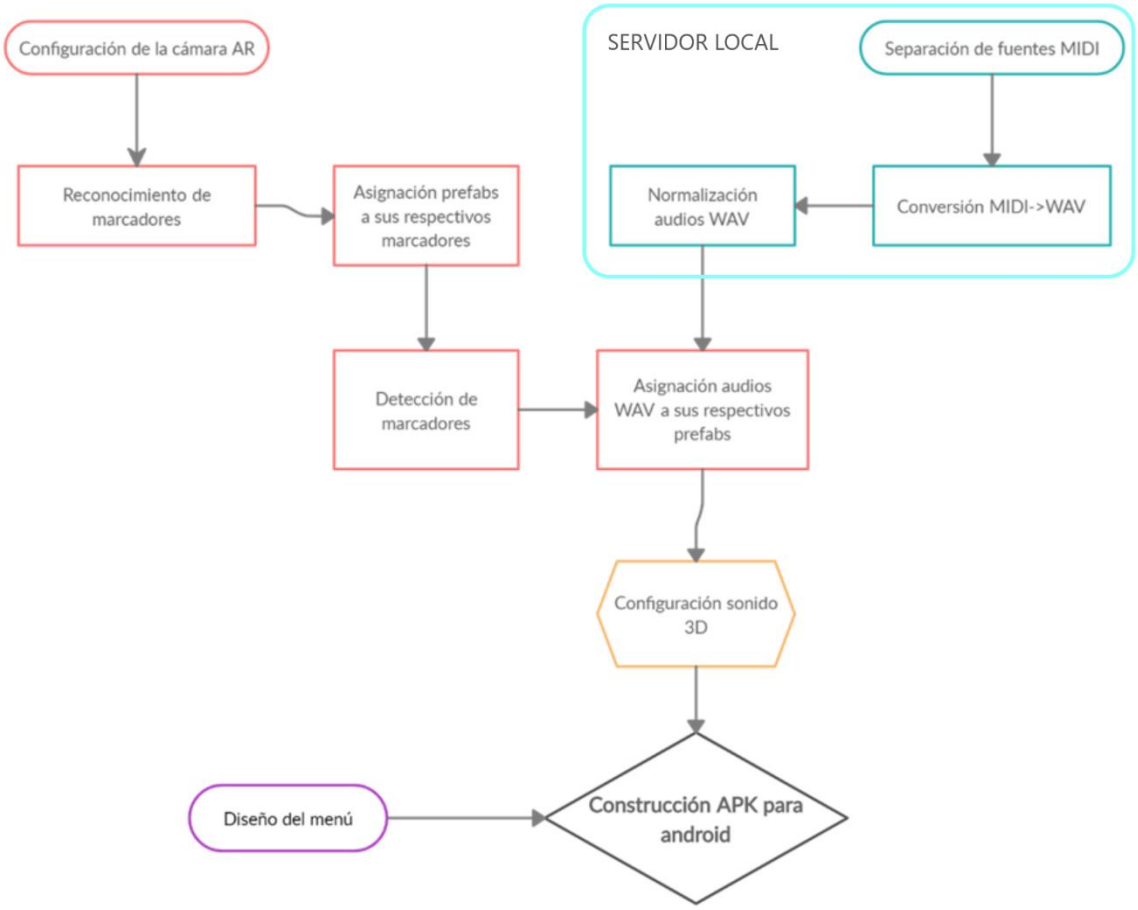


Ilustración 11: Diagrama de flujo para el desarrollo del proyecto

3.2.1. Creación y estructura de la escena en Unity

Previamente, antes de comenzar con la configuración de la cámara AR, se debe de instalar Unity, en cualquiera de sus versiones de 2019 o 2020, para el correcto desarrollo de este proyecto. Para ello, se puede descargar Unity Hub, y conseguir una licencia gratuita para estudiantes o desarrollo personal de la plataforma Unity. En la instalación, se deberá incluir los módulos que se detallan en la siguiente figura.

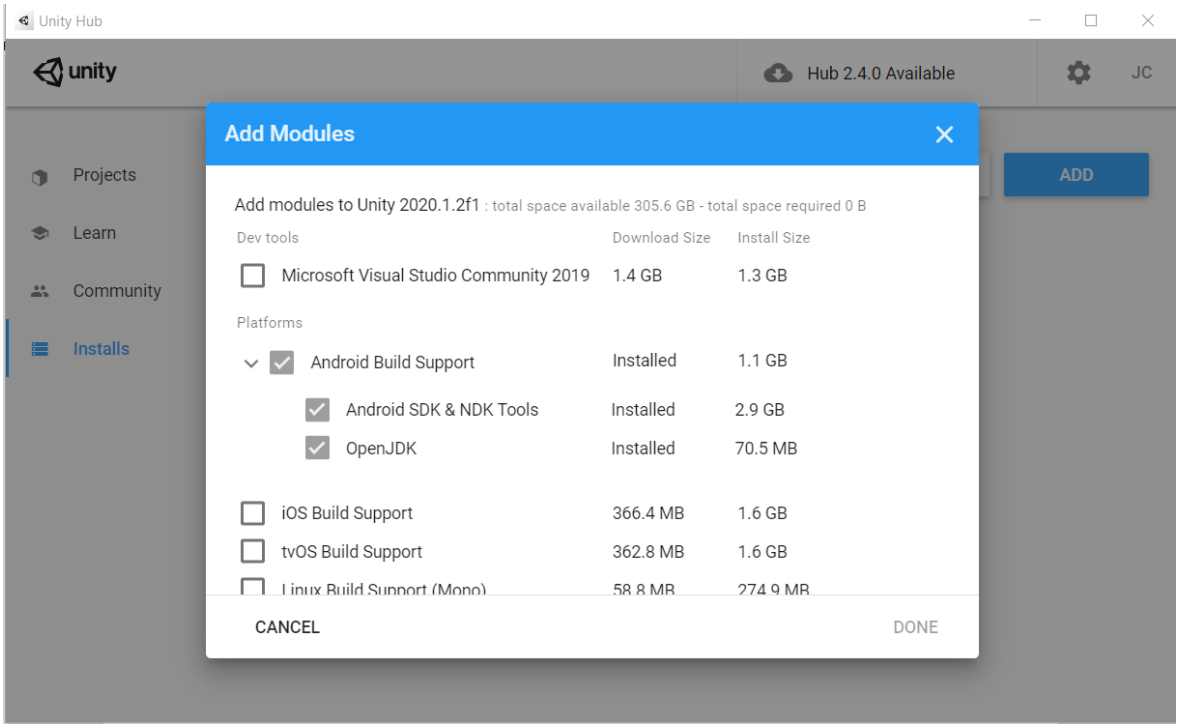


Ilustración 12: Instalación plataforma de Unity

También sería conveniente, aunque opcional, instalar el módulo de *Microsoft Visual Studio Community 2019*, para modificar con mayor facilidad los scripts incluídos en Unity, y el módulo *Universal Windows Platform Build Support*, si se quiere que la aplicación funcione también para cualquier plataforma de Windows.

Posteriormente, se deberá importar el SDK de Vuforia en Unity, para lo que previamente se creará una cuenta en su página oficial y se procederá a descargar su versión más actualizada para Unity.

*Si se utiliza una versión de Unity anterior a 2019, es necesario acceder a la configuración de proyecto (*Project Settings*), y en "XR Settings", asegurarse que la casilla "*Vuforia Augmented Reality Supported*" está activada.

Una vez realizados estos primeros pasos, se deberá configurar la escena para la plataforma Android, esto se podría hacer al principio, durante, o cuando se realice toda la configuración final. Para ello, se accede a los ajustes de la escena (*Build settings*), se selecciona Android, y se hace clic en construir (*Build*).

- **Configuración de la cámara AR:**

Para poder comenzar el desarrollo de la escena, primero se deberá obtener una licencia para la AR Camera. Para ello, se abrirá el portal de *Vuforia* desde el siguiente enlace

https://vuforia.developer.com, y se accederá al apartado de desarrollo, desde el que se obtendrá una clave para el funcionamiento de la cámara.

Una vez obtenida, se puede comenzar la configuración de la escena. Primeramente, se eliminará la cámara por defecto incluida al comienzo de la escena, y se añadirá la AR Camera. Se deberá entrar en *Vuforia Configuration* y pegar la clave que ha sido proporcionada. También es esencial cambiar el número de objetos e imágenes de las que se puede hacer seguimiento simultáneamente para que sea capaz de detectar tantas como sea necesario.

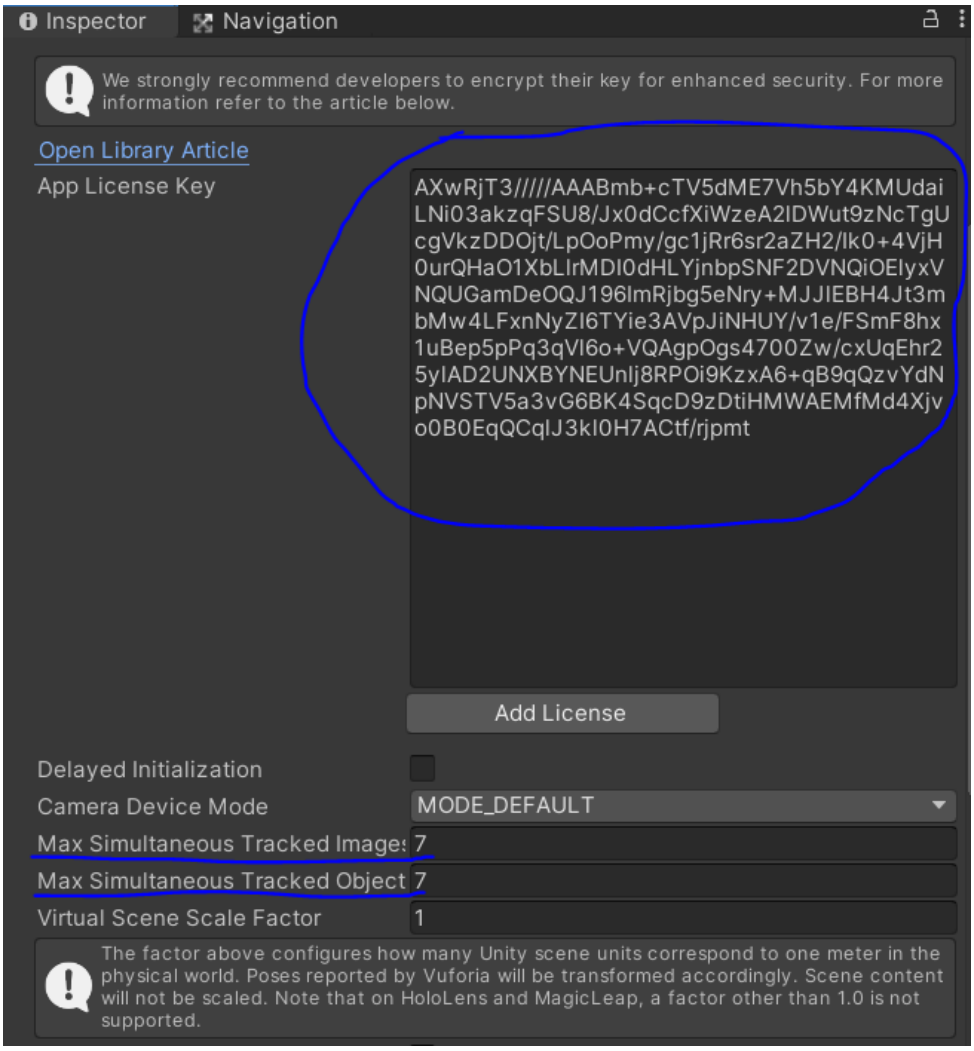


Ilustración 13: Configuración AR Camera

La correcta configuración de la cámara AR es imprescindible para el desarrollo del proyecto, ya que será la encargada de proporcionar el soporte de realidad aumentada que se utilizará después para poder visualizar la escena desde cualquier cámara, ya sea en un dispositivo móvil o en cualquier cámara externa o interna de un ordenador.

Una vez realizados estos pasos y para comprobar que todo funciona correctamente, se puede hacer clic sobre el botón *play* de la escena y observar que la cámara configurada arranca sin problemas.

- **Reconocimiento de marcadores:**

Es necesario que Vuforia contenga una base de datos con marcadores, que son los elementos encargados de indicarle a nuestra AR Camera que actividad RA reproducir sobre los mismos.

Para ello, en la sección de desarrollo del portal de Vuforia, se encuentra un apartado dedicado a crear una base de datos de marcadores (*Target Manager*). Se disponen de diferentes formatos para añadir marcadores como se observa en la *Ilustración 13*.

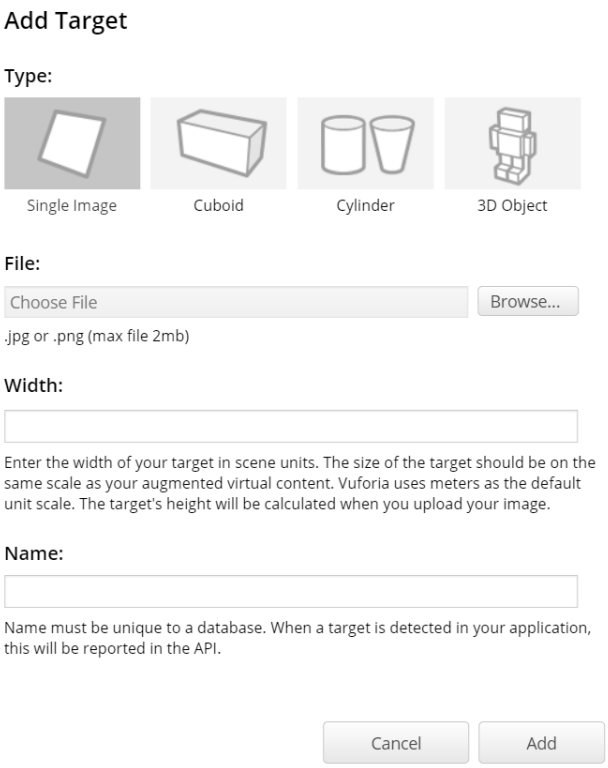


Ilustración 14: Formatos para añadir marcadores a nuestra base de datos en Vuforia

Para el desarrollo del proyecto, se ha elegido la primera opción, correspondiente a imágenes simples, en este caso, instrumentos. La propia aplicación dota con una puntuación del 1 al 5 en forma de estrellas, llamada *rating* (puntuación), la capacidad de esa imagen para ser reconocida como marcador. Es recomendado que, al menos, la imagen posea 3 estrellas si se quiere evitar problemas en el reconocimiento de

marcadores. Para conseguir una mayor puntuación, se deben incluir imágenes con un alto contraste.

Una vez se incluyan los marcadores deseados, se deberá importar la base de datos con el formato Unity Editor, para posteriormente agregarlos a la escena.

En la *Ilustración 14*, se puede observar algunos ejemplos de marcadores con puntuación 5 estrellas.

Ilustración 15: Ejemplo de marcadores que se corresponden con instrumentos musicales

- **Asignación prefabs a sus respectivos marcadores:**

Los prefabs se corresponden con los elementos multimedia que Unity es capaz de reproducir en la escena de la aplicación sobre los marcadores. Se pueden incluir formatos tan diversos como textos, imágenes 2D, vídeos, figuras 3D sin movimiento, figuras 3D con animaciones o sonidos.

En el caso de este proyecto, se han incluido prefabs en formato figuras 3D, a las que posteriormente se les ha añadido un audio en formato wav para que se reproduzca en conjunto con este, como se detallará más adelante.

La obtención de estos modelos o figuras 3D puede ser desde la plataforma "*Asset Store*" de Unity, o bien desde diversas páginas web, en las que, con un poco de suerte, se podría encontrar el modelo 3D asociado al instrumento musical que se requiera de manera gratuita.

En este proyecto, se ha optado por las páginas web https://free3d.com/3d-models/ y https://www.turbosquid.com/es/Search/3D-Models/ para obtener las figuras 3D de los instrumentos musicales requeridos de forma gratuita. Si se quiere añadir nuevos

instrumentos musicales a la escena sin coste y no es posible encontrarlos en las direcciones propuestas, se podría optar por buscar en Google algo parecido a "*instrumentomusical .obj free download*", o bien, crear por si mismo el objeto 3D, pero requeriría de un gasto de tiempo innecesario. Es esencial que el objeto 3D se encuentre con la extensión ".obj" para que Unity lo reconozca.

Una vez se obtengan los objetos 3D que se van a utilizar, se debe crear en la escena los diferentes "Image Target" o marcadores a través de GameObject-> Vuforia Engine-> Image Target y una vez creados, elegir el marcador deseado de la base de datos que se importó previamente.

Lo último que quedaría por hacer es arrastrar el objeto 3D hasta el marcador con el que se quiere asociar y configurar su posición en la escena con respecto a la AR Camera para que al reproducirlo se observe de manera correcta y no tenga un tamaño desproporcionado o sea visualmente desagradable.

A modo de ejemplo, en la *Ilustración 15*, se expone la figura 3D del piano y el archivo utilizado en la aplicación que se corresponde con la misma. En la *Ilustración 16*, se puede observar la configuración de la posición de este objeto 3D para una correcta visualización del mismo sobre el marcador asociado a través de la cámara AR.

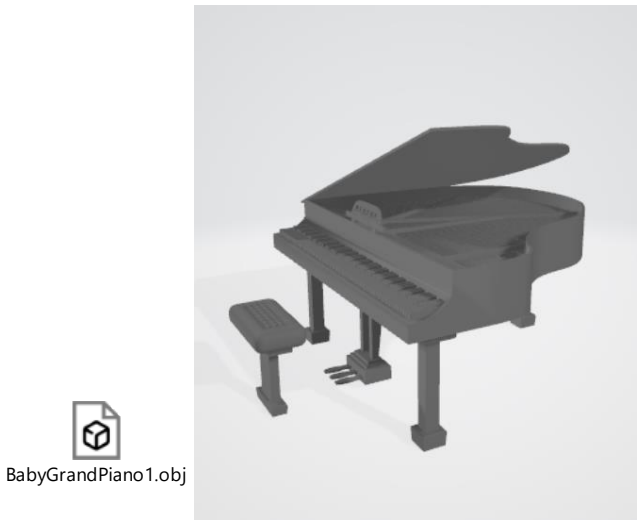


Ilustración 16: Ejemplo figura 3D piano

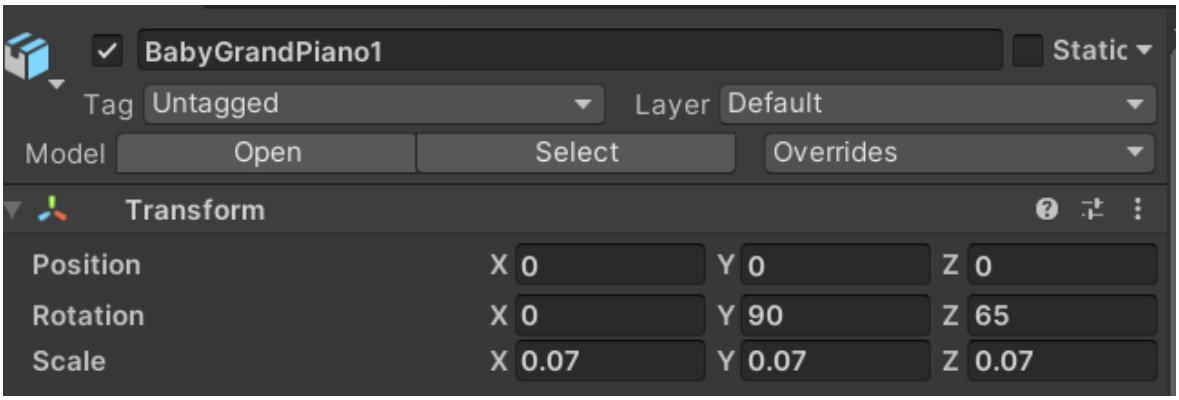


Ilustración 17: Configuración objeto 3D piano para su correcta visualización

- **Detección de marcadores:**

Este apartado, ha tenido un grado de complejidad alto en cuanto a configuración se refiere, ya que existían múltiples vías posibles para abordarlo, pero se requería conseguir un resultado óptimo. Con lo que se refiere a la aparición de las figuras 3D al detectar los marcadores, no existió gran complejidad, pero con respecto a la reproducción y correcta sincronización del sonido 3D, sí que la hubo. Todo esto quedará detallado más adelante en el apartado 3.2.3. Configuración Sonido 3D.

El comienzo para encontrar una solución se basó en crear una función propia de detección de marcadores para optimizar el resultado en función de las necesidades. Tras varios intentos de configuración, presentaba problemas a la hora de programarlo, ya que como se mencionó anteriormente, la mayoría de código de ayuda que se puede encontrar en la web se encuentra bastante desactualizado con respecto a las nuevas versiones de Unity.

Finalmente, se decidió que era una opción viable utilizar el script "DefaultTrackableEventHandler", que viene integrado en Unity por defecto, como detección de marcadores, y que realmente cumple con la función esperada.

En la *Ilustración 17*, se puede observar el correcto funcionamiento en la detección de un marcador, en este caso, el que representa al instrumento musical del clarinete.

Ilustración 18: Detección de marcador (clarinete)

- **Asignación audios WAV a sus respectivos prefabs:**

Una vez se ha realizado el tratamiento de los archivos de audio tal y como se explica en el apartado 3.2.2. Procesamiento del audio, se puede proceder a insertar el 'audioclip' dentro de Unity, concretamente se tienen dos opciones, ya que se puede asignar tanto al marcador, como a la figura 3D, ya que esta última se encuentra asociada igualmente al marcador.

En el apartado creado por defecto al comenzar cualquier proyecto en Unity, llamado "Assets", se puede incluir una subcarpeta que contenga todos los archivos de audio para tener una mayor ordenación de los mismos. Bastará con arrastrar el clip de audio sobre la figura 3D correspondiente para que ésta comience a reproducirse en conjunto con el objeto 3D asociado al activarse el marcador por pantalla.

3.2.2. Procesamiento del audio

Aunque no se planteó como un objetivo, sí que fue recomendado por el tutor de este proyecto que la reproducción del sonido 3D se hiciera a través de un archivo de audio en formato MIDI. Es por eso, que se exploró esta posibilidad y se realizaron diferentes pruebas para la reproducción de archivos MIDI en Unity, y, pese a que se logró el objetivo de integrar archivos MIDI mediante el complemento MIDI Tool Kit Free (MPTK) para Unity, se pudo apreciar que con esta herramienta difícilmente podía lograrse conseguir una espacialidad para los diferentes sonidos, y crear, por tanto, el sonido 3D.

Existe una versión de pago de este 'plug-in' o complemento, llamado MIDI Tool Kit Pro, que se puede adquirir en Asset Store de Unity por el precio de 65$, y que según expone en su descripción, posee un script llamado "SpatializerFly", que permitiría posicionar los instrumentos en un escenario 3D.

Quizás con este complemento, podría haberse logrado también el objetivo de crear el sonido espacial o 3D para la orquesta virtual de realidad aumentada, pese a que la herramienta no está pensada con este propósito (no soporta VuforiaEngine), pero se optó por explorar otras vías que pudiesen ser igualmente válidas y en las que se pudiera conseguir el objetivo sin gasto alguno, primeramente por el ahorro que supone en lo personal, y también, de cara a que si en un futuro la aplicación tuviera algún uso comercial o individual, y se quisieran reproducir otra partituras diferentes a las que se exponen en este proyecto como ejemplo, no le supusiera un impedimento el hecho de tener que desembolsar dinero, pues es totalmente gratuita la vía que se ha tomado para su realización. No obstante, podría ser una línea futura a explorar en este proyecto tal y como se expone en el apartado 7. LÍNEAS FUTURAS.

- **Separación de fuentes MIDI:**

Supone de un grado de dificultad mayor y no es objetivo de este proyecto conseguir una separación de los diferentes instrumentos que suenan en un archivo de audio con formatos tales como .mp3 o .wav.

Para conseguir un producto final en el que se consiga reproducir una partitura con diversos instrumentos con sonido 3D, es más sencillo partir desde un formato en el que los diferentes canales o fuentes de sonido estén perfectamente definidos, como es el caso del formato MIDI. Es por ello que, finalmente, se decidió utilizar este formato para que fuese más práctico.

Utilizando la herramienta MIDI Editor, se pueden separar los diferentes canales y fuentes de un audio en formato MIDI, para obtener un archivo diferente por cada instrumento, de forma manual.

En la *Ilustración 18*, se puede observar cómo seleccionar una única fuente en Midi Editor, para posteriormente guardarla en un nuevo archivo de audio.

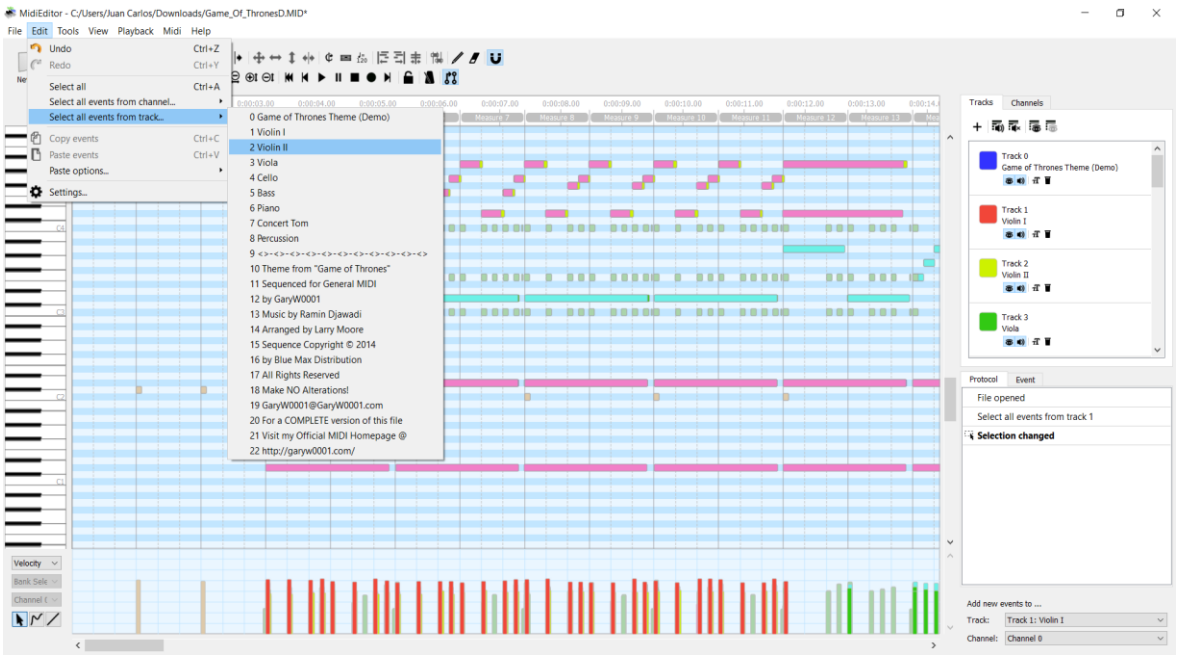


Ilustración 19: Selección de una sola fuente (Violín II)

Para la realización de este proyecto se ha optado por crear un script, en lenguaje Python, que se encargue de la separación de los diferentes instrumentos de manera automatizada.

En MIDI, el sonido de instrumento o "programa" para cada uno de los 16 canales MIDI posibles se selecciona con el mensaje de cambio de programa o program change, que sigue un parámetro de números que muestra que sonido de instrumento corresponde a cada uno de los 128 números de programa posibles para General MIDI exclusivamente.

Para el correcto funcionamiento del script, el archivo midi a procesar deberá seguir la especificación General MIDI Level 1, ya que para el desarrollo del mismo se ha utilizado el orden de esta especificación.

Piano

1 - Piano 1
2 - Piano 2
3 - Piano 3
4 - Honky-Tonk Piano
5 - Electric Piano 1
6 - Electric Piano 2
7 - Harpsichord
8 - Clav.

Chromatic percussion

9 - Celesta
10 - Glockenspiel
11 - Music Box
12 - Vibraphone
13 - Marimba
14 - Xylophone
15 - Tubolar Bell
16 - Santur

Organ

17 - Organ 1
18 - Organ 2
19 - Organ 3
20 - Church Organ
21 - Reed Organ
22 - Accordion
23 - Harmonica
24 - Bandoneon

Guitar

25 - Nylon-str. Guitar
26 - Steel-str. Guitar
27 - Jazz Guitar
28 - Clean Guitar
29 - Muted Guitar
30 - Overdrive Guitar
31 - Distortion Guitar
32 - Guitar Harmonics

Bass

33 - Acoustic Bass
34 - Fingered Bass
35 - Picked Bass
36 - Fretless Bass
37 - Slap Bass 1
38 - Slap Bass 2
39 - Synth Bass 1
40 - Synth Bass 2

Strings/Orchestra

41 - Violin
42 - Viola
43 - Cello
44 - Contrabass
45 - Tremolo Strings
46 - Pizzicato
47 - Harp
48 - Timpani

Ensemble

49 - Strings
50 - Slow Strings
51 - Synth Strings 1
52 - Synth Strings 2
53 - Choir Aahs
54 - Voice Oohs
55 - Synth Voice
56 - Orchestra Hit

Brass

57 - Trumpet
58 - Trombone
59 - Tuba
60 - Muted trumpet
61 - French Horns
62 - Brass 1
63 - Synth Brass 1
64 - Synth Brass 2

Lead

65 - Soprano Sax
66 - Alto Sax
67 - Tenor Sax
68 - Baritone Sax
69 - Oboe
70 - English Horn
71 - Bassoon
72 - Clarinet

Pipe

73 - Piccolo
74 - Flute
75 - Recorder
76 - Pan Flute
77 - Bottle Blow
78 - Shakuhachi
79 - Whistle
80 - Ocarina

Synth Lead

81 - Square Wave
82 - Saw Wave
83 - Synth Calliope
84 - Chiffer Lead
85 - Charang
86 - Solo Vox
87 - 5th Saw Wave
88 - Bass & Lead

Synth Pad

89 - Fantasia
90 - Warm Pad
91 - Polysynth
92 - Space Voice
93 - Bowed Glass
94 - Metal Pad
95 - Halo Pad
96 - Sweep Pad

Synth FX

97 - Ice Rain
98 - Soundtrack
99 - Crystal
100 - Atmosphere
101 - Brightness
102 - Goblin
103 - Echo Drops
104 - Star Theme

Ethnic

105 - Sitar
106 - Banjo
107 - Shamisen
108 - Koto
109 - Kalimba
110 - Bagpipe
111 - Fiddle
112 - Shanai

Percussive

113 - Tinkle Bell
114 - Agogo
115 - Steel Drums
116 - Woodblock
117 - Taiko
118 - Melo Tom
119 - Synth Drum
120 - Reverse Cymbal

Sound FX

121 - Guitar FretNoise
122 - BreathNoise
123 - Seashore
124 - Bird
125 - Telephone
126 - Helicopter
127 - Applause
128 - GunShot

Ilustración 20: Lista de instrumentos General MIDI Level 1

Fuente: https://alchetron.com/cdn/general-midi-2ff96187-f3f9-44be-9342-8e41e3995f7-resize-750.gif

El script, al que se ha llamado como "**script.py**", funciona de la siguiente forma: Primero, se debe importar la herramienta pretty-midi, que puede encontrarse en GitHub, para poder leer, modificar y extraer información de un archivo midi cualquiera. Después, se realiza una copia del archivo midi original por cada conjunto de instrumentos, y, finalmente, se modifican las propiedades de los instrumentos, bajándole la velocidad de la nota, para que solo suene el instrumento correspondiente en cada archivo midi y guardando únicamente los archivos que contengan algún instrumento.

Como se considera un trabajo tedioso y poco práctico crear un código que reconozca y separe los 128 instrumentos posibles uno a uno, se ha optado por separar los instrumentos en agrupaciones de 8, tal y como se indica en la especificación General MIDI Level 1, de manera que, por ejemplo, el piano, entraría dentro del grupo de valores del 1-8.

Si se encontrase una partitura en la que suenan, por ejemplo, varios pianos, o una trompeta y un trombón, que coinciden dentro de una misma agrupación, si el deseo es escuchar estos instrumentos por separado, se debería de optar por la separación manual, o por realizar un cambio en el script tal y como se indica a continuación:

```
for instrument in trompeta.instruments:
    if instrument.program != 56:
        for note in instrument.notes:
            note.velocity = 0
for instrument in trombon.instruments:
    if instrument.program != 57:
        for note in instrument.notes:
            note.velocity = 0
```

De esta forma, se consigue aislar el sonido de un solo instrumento para su posterior uso.

El script completo se muestra a continuación:

```
import pretty_midi
import copy
from os import remove
import os

pitch_cutoff = 65

## Variables Flag para generar o no los archivos MIDI:
pianosFlag = False
percusioncromaticaFlag = False
organosFlag = False
guitarrasFlag = False
bajosFlag = False
cuerdasFlag = False
ensembleFlag = False
metalesFlag = False
maderasFlag = False
sintetizadoresFlag = False
etnicosFlag = False
percusionFlag = False
efectosespecialesFlag = False

mid = pretty_midi.PrettyMIDI('pretty-midi/midi.mid')
```

```
pianos= copy.deepcopy(mid)
percusioncromatica= copy.deepcopy(mid)
organos= copy.deepcopy(mid)
guitarras= copy.deepcopy(mid)
bajos= copy.deepcopy(mid)
cuerdas= copy.deepcopy(mid)
ensemble= copy.deepcopy(mid)
metales= copy.deepcopy(mid)
maderas= copy.deepcopy(mid)
sintetizadores= copy.deepcopy(mid)
etnicos= copy.deepcopy(mid)
percusion= copy.deepcopy(mid)
efectosespeciales= copy.deepcopy(mid)

## Bloque Pianos
for instrument in pianos.instruments:
    if not (instrument.program>=0 and instrument.program<=7):
        for note in instrument.notes:
            note.velocity = 0
    else:
        for note in instrument.notes:
            if note.velocity > 0:
                pianosFlag = True

## Bloque Percusion Cromática
for instrument in percusioncromatica.instruments:
    if not (instrument.program>=8 and instrument.program<=15):
        for note in instrument.notes:
            note.velocity = 0
    else:
        for note in instrument.notes:
            if note.velocity > 0:
                percusioncromaticaFlag = True

## Bloque Organos
for instrument in organos.instruments:
    if not (instrument.program>=16 and instrument.program<=23):
        for note in instrument.notes:
            note.velocity = 0
    else:
        for note in instrument.notes:
            if note.velocity > 0:
                organosFlag = True

## Bloque Guitarras
for instrument in guitarras.instruments:
    if not (instrument.program>=24 and instrument.program<=31):
        for note in instrument.notes:
            note.velocity = 0
    else:
        for note in instrument.notes:
            if note.velocity > 0:
                guitarrasFlag = True

## Bloque Bajos
for instrument in bajos.instruments:
    if not (instrument.program>=32 and instrument.program<=39):
        for note in instrument.notes:
            note.velocity = 0
    else:
        for note in instrument.notes:
            if note.velocity > 0:
                bajosFlag = True

## Bloque Cuerdas
for instrument in cuerdas.instruments:
    if not (instrument.program>=40 and instrument.program<=47):
        for note in instrument.notes:
            note.velocity = 0
    else:
        for note in instrument.notes:
            if note.velocity > 0:
                cuerdasFlag = True

## Bloque Ensemble
for instrument in ensemble.instruments:
    if not (instrument.program>=48 and instrument.program<=55):
```

```
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    ensembleFlag = True

    ## Bloque Metales
    for instrument in metales.instruments:
        if not (instrument.program>=56 and instrument.program<=63):
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    metalesFlag = True

    ## Bloque Maderas
    for instrument in maderas.instruments:
        if not (instrument.program>=64 and instrument.program<=79):
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    maderasFlag = True

    ## Bloque Sintetizadores
    for instrument in sintetizadores.instruments:
        if not (instrument.program>=80 and instrument.program<=103):
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    sintetizadoresFlag = True

    # Bloque Etnicos
    for instrument in etnicos.instruments:
        if not (instrument.program>=104 and instrument.program<=111):
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    etnicosFlag = True

    ## Bloque Percusion
    for instrument in percusion.instruments:
        if not (instrument.program>=112 and instrument.program<=119):
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    percusionFlag = True

    ## Bloque Efectos Especiales
    for instrument in efectosespeciales.instruments:
        if not (instrument.program>=120 and instrument.program<=127):
            for note in instrument.notes:
                note.velocity = 0
        else:
            for note in instrument.notes:
                if note.velocity > 0:
                    efectosespecialesFlag = True

    ##Se guardan los archivos midi que contengan algun instrumento del bloque

    if pianosFlag == True:
        pianos.write("C:/xampp/htdocs/tfg/midgenerados/1_pianos.mid")
    else:
        if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/1_pianos.mid'):
            remove("C:/xampp/htdocs/tfg/midgenerados/1_pianos.mid")

    if percusioncromaticaFlag == True:
        percusioncromatica.write("C:/xampp/htdocs/tfg/midgenerados/2_percusioncromatica.mid")
```

```
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/2_percusioncromatica.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/2_percusioncromatica.mid")

if organosFlag == True:
    organos.write("C:/xampp/htdocs/tfg/midgenerados/3_organos.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/3_organos.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/3_organos.mid")

if guitarrasFlag == True:
    guitarras.write("C:/xampp/htdocs/tfg/midgenerados/4_guitarras.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/4_guitarras.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/4_guitarras.mid")

if bajosFlag == True:
    bajos.write("C:/xampp/htdocs/tfg/midgenerados/5_bajos.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/5_bajos.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/5_bajos.mid")

if cuerdasFlag == True:
    cuerdas.write("C:/xampp/htdocs/tfg/midgenerados/6_cuerdas.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/6_cuerdas.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/6_cuerdas.mid")

if ensembleFlag == True:
    ensemble.write("C:/xampp/htdocs/tfg/midgenerados/7_ensemble.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/7_ensemble.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/7_ensemble.mid")

if metalesFlag == True:
    metales.write("C:/xampp/htdocs/tfg/midgenerados/8_metales.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/8_metales.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/8_metales.mid")

if maderasFlag == True:
    maderas.write("C:/xampp/htdocs/tfg/midgenerados/9_10_maderas.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/9_10_maderas.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/9_10_maderas.mid")

if sintetizadoresFlag == True:
    sintetizadores.write("C:/xampp/htdocs/tfg/midgenerados/11_12_13_sintetizadores.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/11_12_13_sintetizadores.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/11_12_13_sintetizadores.mid")

if etnicosFlag == True:
    etnicos.write("C:/xampp/htdocs/tfg/midgenerados/14_etnicos.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/14_etnicos.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/14_etnicos.mid")

if percusionFlag == True:
    percusion.write("C:/xampp/htdocs/tfg/midgenerados/15_percusion.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/15_percusion.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/15_percusion.mid")

if efectosespecialesFlag == True:
    efectosespeciales.write("C:/xampp/htdocs/tfg/midgenerados/16_efectosespeciales.mid")
else:
    if os.path.exists('C:/xampp/htdocs/tfg/midgenerados/16_efectosespeciales.mid'):
        remove("C:/xampp/htdocs/tfg/midgenerados/16_efectosespeciales.mid")


print(True)
```

Nótese que los sintetizadores y los instrumentos de madera se han agrupado en un conjunto mayor, ya que no suelen ser tan relevantes en la mayoría de partituras y así se reduce considerablemente el número final de archivos.

- **Conversión MIDI->WAV:**

Una vez se obtienen todas las fuentes separadas, que se corresponden con cada instrumento de la partitura, en diferentes archivos de audio MIDI, se deberá realizar una conversión de formato .mid a formato .wav. Es banal encontrar una aplicación o página web que te haga la conversión online, tales como la que se expone: https://audio.online-convert.com/es/convertir/midi-a-wav. Pero para una automatización del proceso, se optó por crear un script llamado **conversor.py**, que hace uso de la librería midi2audioque se muestra a continuación:

```
from midi2audio import FluidSynth
fs = FluidSynth()
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/1_pianos.mid","C:/xampp/htdocs/tfg/midgenerados/1_pianos.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/2_organos.mid","C:/xampp/htdocs/tfg/midgenerados/2_organos.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/3_organos.mid","C:/xampp/htdocs/tfg/midgenerados/3_organos.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/4_guitarras.mid","C:/xampp/htdocs/tfg/midgenerados/4_guitarras.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/5_bajos.mid","C:/xampp/htdocs/tfg/midgenerados/5_bajos.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/6_cuerdas.mid","C:/xampp/htdocs/tfg/midgenerados/6_cuerdas.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/7_ensemble.mid","C:/xampp/htdocs/tfg/midgenerados/7_ensemble.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/8_metales.mid","C:/xampp/htdocs/tfg/midgenerados/8_metales.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/9_10_maderas.mid","C:/xampp/htdocs/tfg/midgenerados/9_10_maderas.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/11_12_13_sintetizadores.mid","C:/xampp/htdocs/tfg/midgenerados/11_12_13_sintetizadores.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/14_etnicos.mid","C:/xampp/htdocs/tfg/midgenerados/14_etnicos.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/15_percusion.mid","C:/xampp/htdocs/tfg/midgenerados/15_percusion.wav")
fs.midi_to_audio("C:/xampp/htdocs/tfg/midgenerados/16_efectosespeciales.mid","C:/xampp/htdocs/tfg/midgenerados/16_efectosespeciales.wav")

print(True)
```

- **Normalización audios .wav:**

Cuando se parte de una partitura original sea el formato que sea, es habitual que cada instrumento musical tenga un peso diferente en la partitura, es decir, lo normal es que en ciertas ocasiones se le dé más notoriedad y resalten ciertos instrumentos sobre otros.

En este proyecto, se han realizado pruebas para comprobar la diferencia entre normalizar todos los instrumentos o no hacerlo, y aunque a continuación se exponen las técnicas para normalizarlos, se ha optado por incluir en la aplicación ejemplos de obras musicales sin normalizar, con el fin de mantener la fidelidad de la obra original, a excepción de un ejemplo, para que también se pueda apreciar el cambio que supone normalizarlos.

Para normalizar las diferentes fuentes generadas se ha utilizado el programa Audacity, el cual, además de normalizar, permite comprobar si ha habido una disminución en la calidad de sonido tras el proceso de separación de fuentes y conversión, ya que es capaz de reproducir los instrumentos que se deseen elegir.

Lo único que se tendrá que realizar para normalizar las distintas fuentes de audio es importar todas las fuentes pertenecientes a la partitura, seleccionarlas todas y utilizar uno de los efectos que incluye llamado *normalización*. En la *Ilustración 19*, se puede observar un ejemplo de normalización para todos los archivos de audio.

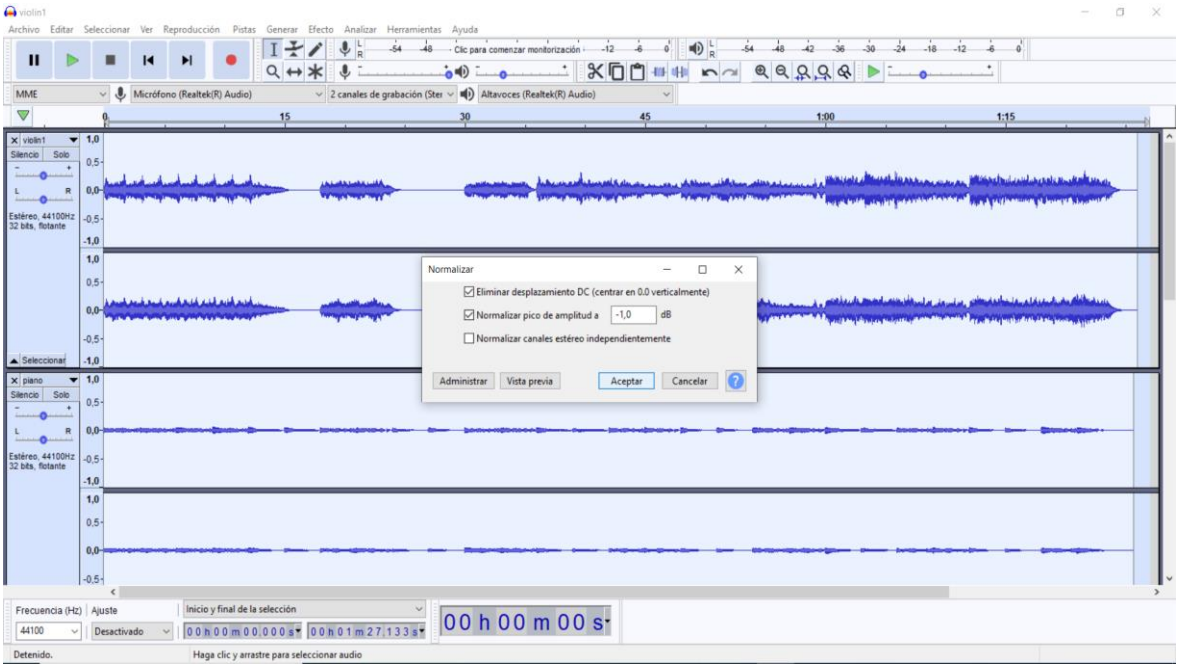


Ilustración 21: Normalización de los archivos de audio en Audacity

SERVIDOR LOCAL:

Para la realización de este proyecto, se pensó en crear un servidor local que alojara y arrojara los archivos de audio MIDI procesados, para su posterior utilización en la aplicación, con el fin de obtener e introducir los archivos correspondientes a cada instrumento de manera automática en la aplicación Android.

Para poder obtener los archivos de audio alojados en el servidor, se debe hacer uso de la función *UnityWebRequestMultimedia.GetAudioClip* y obtener los archivos desde *Streaming Assets* de Unity con la función *Application.streamingAssetsPath*, pero el problema se acentúa ya que esta opción solo permite obtener los archivos para la versión de escritorio de Unity, no para Android. Para comunicarse con servidores locales vía https en Android es necesario el uso de la función *DownloadHandlerAudioClip*, pero para este

proyecto no se ha podido configurar de manera correcta ya que al exportar el proyecto y transformarlo en formato .apk, tanto la carpeta Streaming Assets, como todos los archivos que aloja, se comprimen y se reordenan de manera que no ha sido posible encontrar una ruta específica para añadirlos a la función AudioSource. Añadido a este problema, se encuentra la dificultad de que la mayoría de scripts desarrollados por terceros para realizar funciones similares o de información disponible en la web se basa en funciones ya obsoletas de Unity, que no permiten utilizarlas a día de hoy, como la función WWWForm. Se estudiará en 7.LINEAS FUTURAS la forma de poder comunicarse con el servidor y alojar los archivos de audio de manera automatizada.

Como consiguiente, el resultado de este estudio ha finalizado con la creación de un servidor local Apache, en el que se puede introducir cualquier archivo MIDI previamente descargado de la web, y una vez cargado en el formulario se envía, procesando y convirtiendo el archivo MIDI inicial en diferentes archivos .wav, uno por cada instrumento, y alojándolos en el servidor local. Una vez alojados en el servidor local, bastará con arrastrar cada audio a su función AudioSource correspondiente.

Para la configuración correcta del servidor, se han seguido los siguientes pasos:

- Descarga del programa XAMPP, que permite habilitar el servidor local Apache.

Ilustración 22: Configuración servidor Apache a través de XAMPP

- Instalación de los paquetes Pretty-Midi y FluidSynth, que permitirán el correcto funcionamiento de los scripts desarrollados. Para ello bastará con descargar los instaladores desde la web oficial e instalarlos a través del Símbolo de Sistema

de Windows mediante los comandos *pip install pretty_midi y pip install fluidsynth,* respectivamente.

- Adjuntar los scripts **script.py** y **conversor.py** dentro de la carpeta Pretty-Midi, que se encuentra en la ruta C:\xampp\htdocs\tfg\pretty-midi.
- Adjuntar los scripts **procesado.php** y **index.php** en la ruta C:\xampp\htdocs\tfg\, que se han programado con el fin de procesar los scripts Python anteriormente mencionados y de crear el formulario correspondiente en el servidor Apache.

A continuación, se adjunta el script procesado.php y el script index.php, en forma de figura:

```php
<?php
$uploaddir = 'C:\xampp\htdocs\tfg\pretty-midi/'; // Path del directorio donde se va a guardar el archivo mid
 // Archivo que se recibe del formulario que su campo es file
 //Se accede al indice "name" para cambiarle el nombre por uno genérico
$_FILES['midi']['name'] = "midi.mid";

$uploadfile = $uploaddir . basename($_FILES['midi']['name']);//Directorio final con el archivo


//El metodo move_uploaded_file mueve el archivo de la carpeta temporal de apache al directorio que se le indica
// Se tiene que acceder al indice "tmp_name" ya que es el nombre que le dara el servidor apache temporalmente
$archivoSubido = move_uploaded_file($_FILES['midi']['tmp_name'], "$uploadfile");
//true subida correcta, false incorrecta

//Se ejecuta la funcion que a su vez ejecuta internamente los scripts de Python
EjecutarScriptProcesado($archivoSubido);


/**
 * Funciones
 */

/**
 * Ejecuta Un script Python
 * @param $archvivoSubido, respuesta de la copia de un archivo de la carpeta temporal a la indicada,
 *        si el archivo se ha movido con éxito será true en caso contrario false
 *
 * @return boolean, Devuelve false en caso de no completarse el proceso y true si lo completa con éxito
 */
function EjecutarScriptProcesado($archivoSubido)
{
    if ($archivoSubido == true) {
        $command = escapeshellcmd('py pretty-midi/script.py');
        $output = shell_exec($command);
        if ($output == true) {
            echo "Archivos generados, Convirtiendo archivos de mid a wav...";
            if (ejecutarScriptConversion() == true) {
                echo "Archivos convertidos a wav";
            } else {
                echo "No se ha podido convertir a wav";
            }
        } else {
```

```
            echo "No se ha podido subir el archivo...";
        }
    }
    return false;
}

/**
 * Ejecuta Un script Python
 */
function ejecutarScriptConversion()
{
    $command = escapeshellcmd('py pretty-midi/conversor.py');
    $output = shell_exec($command);
    if ($output == true) {
        return true;
    }
    return false;
}
```

script.py | conversor.py | procesado.php | descargar.php | index.php

```
1  <form action="procesado.php" method="post" enctype="multipart/form-data">
2      <p>archivoMidi:
3      <input type="file" name="midi" />
4      <input type="submit" value="Enviar" />
5      </p>
6  </form>
7
8  <form action="descargar.php" method="get">
9      <p>descargar:
10     <input type="text" name="path" />
11     <input type="submit" value="Enviar" />
12     </p>
13 </form>
```

Ilustración 23: Script index.php

- De manera opcional, se ha desarrollado un script llamado **descargar.php** que permite descargar los archivos de audio alojados en el servidor local.

script.py | conversor.py | procesado.php | descargar.php | index.php

```php
<?php
//Petición debe ser la ruta del archivo ejemplo "midgenerados/bajo.mid"
if(isset($_GET['path']))
{
//Obtiene la ruta y el archivo
$filename = $_GET['path'];

//Comprueba que existe
if(file_exists($filename)) {

//Define header information
header('Content-Description: File Transfer');
header('Content-Type: application/octet-stream');
header("Cache-Control: no-cache, must-revalidate");
header("Expires: 0");
header('Content-Disposition: attachment; filename="'.basename($filename).'"');
header('Content-Length: ' . filesize($filename));
header('Pragma: public');

//Limpiamos buffer
flush();

//lee tamaño
readfile($filename);

//kill
die();
}
else{
echo "File does not exist.";
}
}
else
echo "Filename is not defined.";
```

Ilustración 24: Script descargar.php

El servidor local creado se encuentra alojado en la dirección web http://localhost/tfg/. A continuación, se muestra una captura de su funcionamiento.

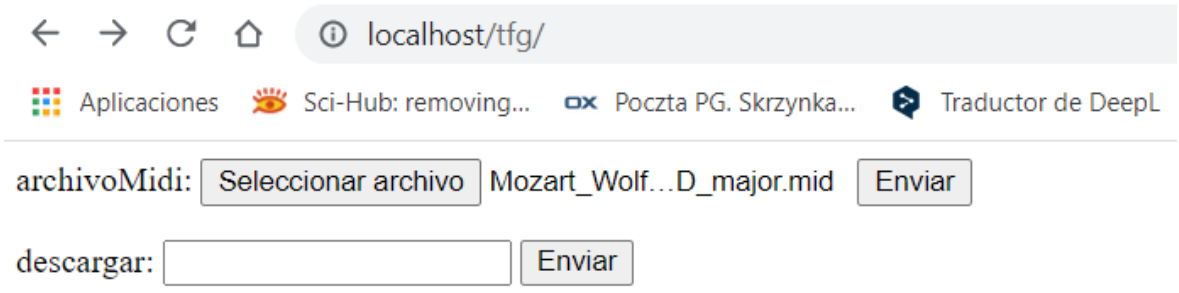


Ilustración 25: Formulario del servidor local Apache

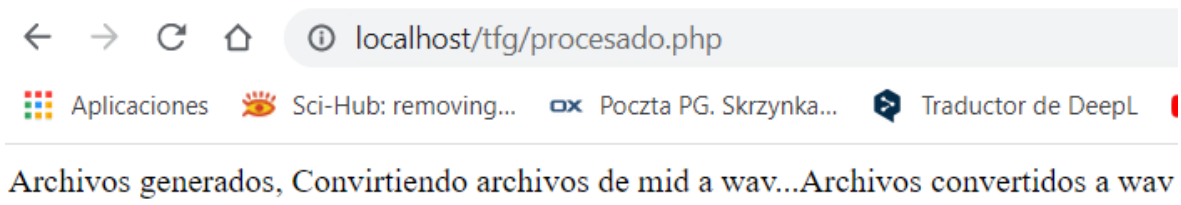


Ilustración 26: Respuesta del servidor local

Cuando se envíe la solicitud, si el archivo MIDI ha sido subido de manera correcta y ha sido procesado y convertido en los diferentes archivos de audio, devolverá la respuesta que se muestra en la *Ilustración 26*. De esta forma, se sabrá que todo ha ido según lo previsto. En caso contrario, devolverá un mensaje de error dependiendo de dónde haya surgido el problema, para poder subsanarlo con mayor facilidad.

Si no se desea realizar la configuración del servidor Apache, también se podrá optar por ejecutar los scripts a través de Símbolo de Sistema de Windows o de un entorno de desarrollo integrado (IDE) que soporte los lenguajes de programación PHP y Python. También es posible integrarlo en Ubuntu, tan solo bastaría con cambiar las rutas de los archivos por sus correspondientes en Linux.

3.2.3. Configuración Sonido 3D

Para reproducir el sonido 3D dentro de la escena, se pensaron y estudiaron varias opciones, en concreto, se indagó en 4 vías distintas para la resolución de este objetivo, las cuales se mencionan a continuación:

- **Complemento MPTK (MIDI Tool Kit Pro):**

Como se comentó anteriormente, existe un plug-in para Unity capaz de situar a los instrumentos en un escenario para dotarlos de sonido espacial o 3D mediante una serie de configuraciones, y, además, teniendo la capacidad de introducir los sonidos en formato MIDI. Esta idea fue la primera en desecharse, ya que suponía un desembolso de dinero, y se optó por estudiar otras vías que pudieran satisfacer de igual forma este objetivo.

- **3D Audio Plugin for Unity:**

En este caso, merece la pena comentar otro complemento para Unity creado por *Qualcomm Advanced Content Group,* que es capaz de integrar el audio 3D en cualquier proyecto Unity compatible y proporciona herramientas para ayudar a convertir proyectos no diseñados originalmente para 3D o realidad virtual. Es una herramienta muy profesional pensada para ser utilizada en juegos de última generación, por lo que el correcto uso de la misma es complicado. Se hicieron pruebas y no se obtuvo el resultado deseado, por lo que finalmente se desechó como solución, debido también a que se encontró un método más eficaz para resolver este objetivo.

- **Programar un script propio:**

Se pensó también en desarrollar un script que pudiera situar a los objetos 3D en el escenario, y a partir de aquí, conseguir un efecto de sonido 3D dependiendo de la

localización de los mismos. Aunque era la vía más compleja, ya que consistía en programar de cero, había información disponible en la web que daba posibilidad para lograr su desarrollo. Esta idea se desechó, tras días de pruebas, debido a como se ha mencionado en anteriores ocasiones, la información que se encontró estaba muy desactualizada con respecto a las nuevas versiones de Unity.

- **Script *AudioSource* de Unity:**

Las nuevas versiones de Unity incluyen opciones para configurar sonido 3D dentro de la función *AudioSource*. Esta vía fue la que, finalmente, se decidió tomar ya que fue la más eficaz para resolver el objetivo con gran precisión, y, además, de una manera más práctica, ya que facilitaba la introducción de nuevas partituras sin gran demora de tiempo.

La función AudioSource es una completísima herramienta capaz de reproducir un clip de audio con respecto a un AudioListener o a través de un AudioMixer. Se puede configurar de manera que los reproduzca como sonido 2D, 3D o una mezcla (SpatialBlend). Es capaz de sonar desde estéreo hasta 7.1. Para controlar la intensidad del sonido con respecto al oyente (AudioListener) se pueden utilizar las llamadas curvas RollOff. Además, tiene funciones para incluir reverberaciones y aplicarlas en la fuente, así como filtros individuales que se pueden aplicar a cada fuente de audio para una experiencia de audio más enriquecedora, a través de lo que se denomina AudioEffects.

Cabe destacar lo que se llaman funciones de distancia, que engloban varias propiedades de audio que pueden ser modificadas como una función de la distancia entre el AudioSource y el AudioListener:

- **Volume:** Permite modificar la amplitud (0.0 - 1.0) sobre la distancia al AudioListener.
- **Spatial Blend:** Permite modificar el sonido 2D a 3D. Para ello, mezcla todos los canales a monocanal y los atenúa de acuerdo a la distancia y dirección del GameObject con respecto al AudioListener.
- **Spread:** Establece el ángulo de propagación (en grados) de un sonido estéreo o multicanal en el espacio de los altavoces. Para un valor igual a 0, todos los canales de sonido están situados en la misma ubicación del altavoz y es 'mono'. Para un valor de 360, todos los canales se sitúan en la ubicación del altavoz opuesta a la que debería estar según la posición 3D.

- **Low-Pass:** Establece un filtro paso bajo con frecuencia de corte sobre la distancia del AudioListener (rango de 22000-10 Hz). Para hacer uso de esta propiedad se debe adjuntar la función LowPassFilter junto al AudioSource.
- **Reverb Zone**: Esta propiedad permite distorsionar el AudioClip y crear una reverberación dependiendo de dónde se encuentre el AudioListener dentro de la zona de reverberación. Sería útil para simular escenarios como salas de conciertos o teatros, que cuentan con su propia reverberación, y reproducir con fidelidad como sonaría en esos escenarios de manera virtual.

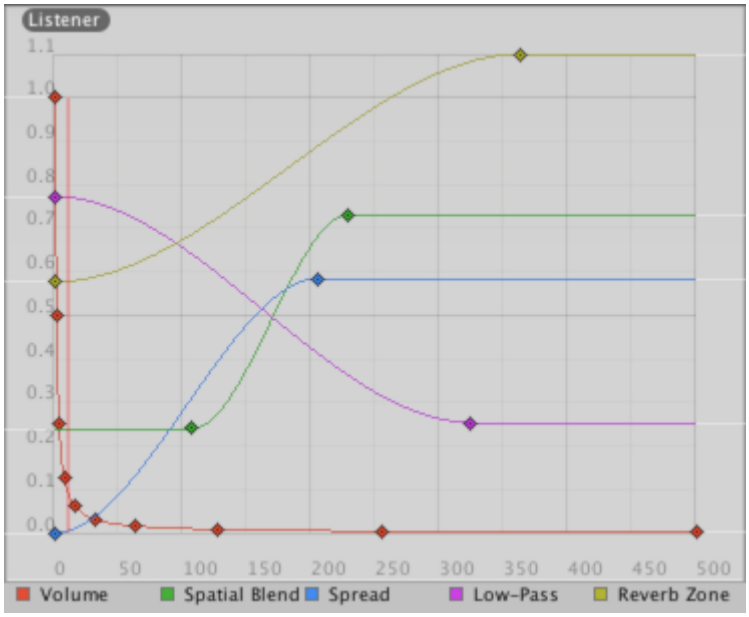


Ilustración 27: Las funciones de distancia de la función AudioSource. La distancia actual al Audio Listener está marcada en la gráfica por la línea roja vertical.

Fuente: https://docs.unity3d.com/es/2020.1/uploads/Main/AudioDistanceFunctions.png

En esta solución propuesta, se puede configurar el sonido 3D de cualquier clip de audio que se introduzca mediante una curva exponencial, que puede ser modificada conforme más interese, pudiendo crear un cambio brusco con respecto a la posición del objeto, o suavizarlo para obtener una transición en el volumen del instrumento musical más leve.

Primeramente, al introducir el clip de audio, tras haberlo procesado tal y como se indica en el apartado 3.2.2. Procesamiento del audio, se deberán activar las casillas *Play on Awake* y *Loop*, tal y como se indica en la *Ilustración 20*. Esto permitirá que los audios correspondientes a cada instrumento queden sincronizados, y, por tanto, la partitura suene fiel a la original.

También, se deberá posicionar la casilla *Spatial Blend* de 0 a 1, lo cual como su nombre indica, dará la mezcla espacial o sensación de espacialidad al reproducir la escena.

Por último, se cambiará la casilla *Max Distance*, que indica la distancia máxima a la que dejará de sonar el audio, y se modificará la exponencial predefinida de manera que el cambio de volumen con respecto a la distancia del objeto no sea muy brusco.

Es importante configurar en la AR Camera la distancia a la que se encuentra con respecto a los instrumentos de la escena. Se deberá alejar lo suficiente para crear la sensación de sonido 3D, ya que dentro de la AR Camera se encuentra la función o script AudioListener, que es la que se encarga de posicionar al oyente dentro de la escena. En este caso, el oyente sería la cámara desde la que se está observando la escena, por eso es importante colocarla alejada de los marcadores, para que exista cierto margen a la hora de atravesar la curva exponencial programada de cada audio.

La función AudioListener posiciona al oyente en el plano 3D del mundo real haciendo uso del giroscopio del dispositivo móvil, y, de esta forma, es capaz de posicionar en los auriculares los sonidos que provienen de las distintas fuentes según donde se encuentren estas fuentes, de manera que si el marcador está a orientado a 0º grados con respecto al objetivo de la cámara, se percibirá como si viniese de frente, pero, en cambio, si se realiza un giro de 180º, posicionándose de espaldas al marcador, se percibirá el sonido como si estuviese detrás del oyente.

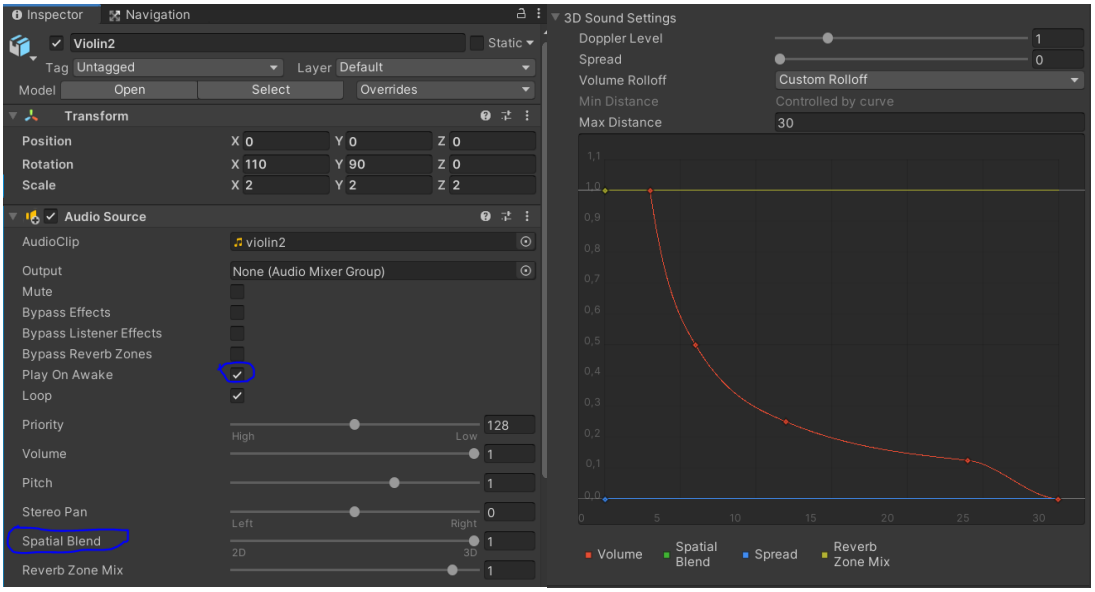


Ilustración 28: Configuración de Sonido 3D para un audioclip cualquiera

3.2.4. Diseño del menú

Se ha creado un menú principal que contiene 5 botones (excluyendo el botón para salir de la aplicación y el logo de la universidad de Jaén que nos redirecciona a su página web), y la función de cada uno de ellos se expone a continuación:

- **Información App**: Contiene un submenú que incluye información acerca de cómo utilizar la aplicación y un botón para volver al menú principal.
- **Orquesta Virtual**: Contiene un submenú en el que se inicia la AR Camera para poder comenzar el reconocimiento de marcadores. También dispone de un botón para volver al menú principal.
- **Elegir Composición Musical**: Contiene un submenú en el que se podrá seleccionar la partitura que se desee reproducir en la Orquesta Virtual. Consta de un espacio en blanco que facilita la incorporación de una nueva obra musical al proyecto.
- **Vídeo demostrativo**: Nos redirecciona a un enlace de YouTube, que contiene el vídeo demostrativo del funcionamiento de la aplicación.
- **Descargar Marcadores**: Nos redirecciona a un enlace de Google Drive, que contiene las imágenes que se utilizan como reconocimiento de marcadores.

Para salir de la aplicación se ha utilizado un script sencillo que se muestra en la *Ilustración 29*. Siguiendo el mismo procedimiento, para redireccionar a una página web concreta al hacer clic, se ha diseñado un script como el que se muestra en la *Ilustración 30*.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Salir : MonoBehaviour
6   {
7       public void SalirAp()
8       {
9           Application.Quit();
10      }
11
12  }
13
```

Ilustración 29: Script función salir

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AbrirYouTube : MonoBehaviour
{
    public void AbrirYT()
    {
        Application.OpenURL("https://www.ujaen.es/");
    }
}
```

Ilustración 30: Script función redireccionar a enlace (Ejemplo web Ujaen)

En la *Ilustración 31,* la *Ilustración 32* y la *Ilustración 33* se puede observar el diseño del menú principal, del submenú Información App y del submenú Elegir Composición Musical, respectivamente.

Ilustración 31: Diseño del menú principal de la App

Ilustración 32: Diseño del submenú Información App

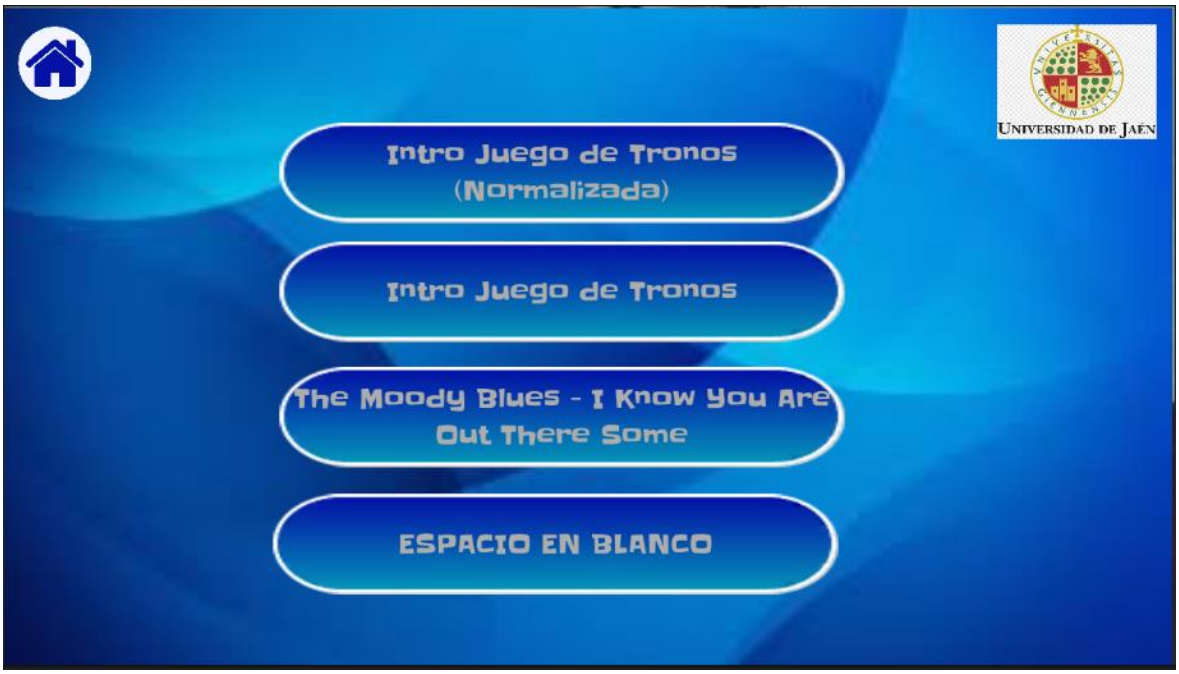


Ilustración 33: Diseño del submenú Elegir Composición Musical

En el futuro, tal y como se indica en 7. LINEAS FUTURAS, se podría estudiar la posibilidad de expandir este menú con nuevas características tales como la grabación de vídeo desde la cámara AR.

3.2.5. Construcción de la APK para Android

Tras la configuración de todos los parámetros tal y como se indica en anteriores puntos, ya solo quedaría confeccionar la aplicación para Android.

Para ello, se deberá hacer clic en la pestaña Archivo (“*File*”) y entrar en las opciones de construcción del proyecto (“*Build Settings*”). Una vez aquí, se deberá elegir la plataforma Android, si no ha sido seleccionada anteriormente, y hacer clic sobre Construir (“*Build*”). Con estos sencillos pasos, la aplicación funcional con la extensión “APK” estará disponible donde haya sido guardada y solo habrá que enviarla a un dispositivo móvil e instalarla, concediendo permiso de cámara a la aplicación, y permiso de instalación de aplicaciones de origen desconocido a dicho dispositivo.

Si en las APIs gráficas estuviera incluida ‘*Vulkan’*, habría que desactivarla y dejar solamente la API ‘*OpenGLES3’,* ya que ‘*Vulkan’* no es compatible con *VuforiaEngine* y daría error en el ensamblado del proyecto- Para ello, habría que ir a ‘*Player Settings’* y una vez allí hacer click sobre ‘*Other Settings*’ y eliminar ‘*Vulkan*’ del apartado ‘*Graphics APIs’*.

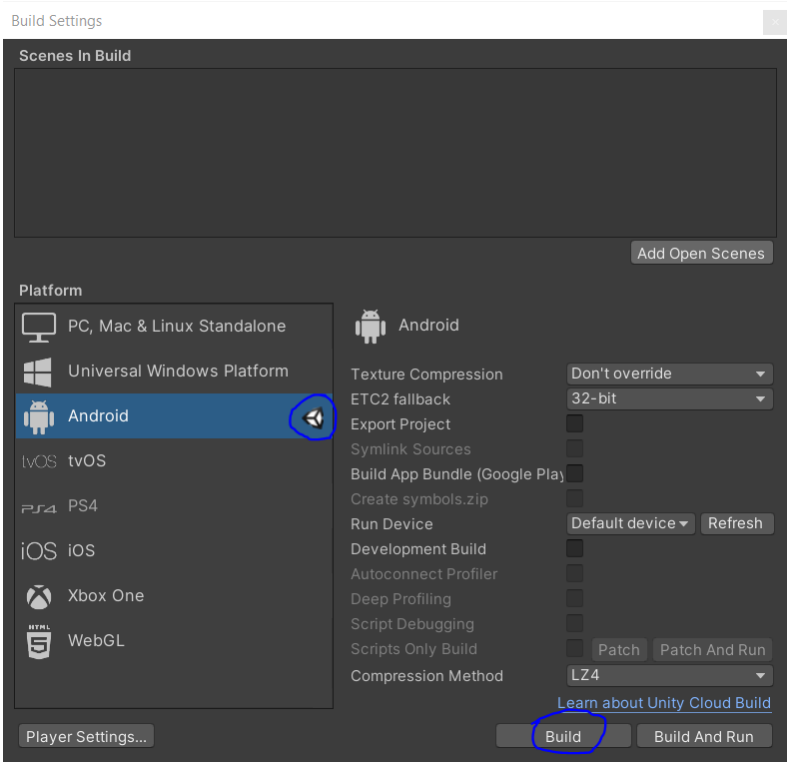


Ilustración 34: Construcción de la APK para Android

4. RESULTADOS Y DISCUSIÓN

Como resultado final, se presenta una aplicación funcional, tanto para ordenador como para smartphone, capaz de reproducir partituras con diferentes instrumentos musicales en formato .wav, y de manera sincronizada con tecnología RA y uso de sonido espacial. Todo ello implementado en la plataforma de desarrollo Unity, haciendo uso de Vuforia, como herramienta para proporcionar el entorno de realidad aumentada. Además, presenta una interfaz simple e intuitiva para facilitar su uso.

Tras varios intentos, se consiguió lograr los objetivos con éxito, pese a la dificultad de llevar a cabo algunos de ellos. Aunque no se planteó como un objetivo, se considerará en un futuro introducir las partituras en formato MIDI directamente, para ahorrarse el proceso de conversión y normalización de audio, así como de crear un servidor local o en la nube capaz de obtener y procesar los archivos de audio directamente desde la aplicación móvil.

Es cierto que, aunque se logró la sincronización de todos los instrumentos musicales con éxito para la correcta reproducción de la partitura, al desaparecer en escena, tras ser detectados los marcadores, los instrumentos musicales siguen reproduciéndose, lo cual en ciertas ocasiones puede suponer un problema, pero mirando con perspectiva se puede incluso sacar provecho de esta situación.

Se puede considerar un problema, pero también se puede observar como una ventaja o una característica única, ya que, al dejar de detectar el marcador, es decir, cuando se apunta con la AR Camera a una posición distinta y éste deja de ser visible, el sonido del instrumento que se estaba reproduciendo en escena, se va desplazando por los auriculares dependiendo de la posición, de manera que, si por ejemplo, el marcador queda a la izquierda del oyente tras haber hecho un giro de 90º con la cámara, el sonido se concentrará principalmente en el auricular izquierdo del usuario, de manera más marcada que si se observara de frente, estando el mismo a la izquierda de la pantalla de la cámara. De igual forma, tras detectar el target, si el usuario realiza una vuelta de 360º, se observa como el sonido va ‘orbitando’ alrededor de la cabeza. Esto es gracias al uso de la función AudioListener, que utiliza el giroscopio del teléfono móvil, y una vez detecta el marcador, o lo que es lo mismo, lo sitúa en una posición con respecto al oyente, es capaz de recrear este entorno de sonido espacial o 3D con gran acierto.

Gracias a esta característica, se puede comprobar cómo se escucharían diferentes instrumentos en distintas localizaciones en un plano de 360º, lo único que habría que hacer es asegurarse primero de que los marcadores hayan sido detectados al comienzo al menos una vez, para que se inicien o activen.

Igualmente, si se desea evitar que el instrumento musical que no se encuentre en pantalla pare de escucharse, bastará con alejar el marcador lo suficiente de la AR Camera antes de enfocar hacia otro lado para que cese su sonido, ya que el sonido espacial o 3D está configurado con una curva exponencial modificada para que tenga un decaimiento en cero cuando la localización esté lo suficientemente alejada, y dado que esta exponencial se calcula mediante la distancia del marcador respecto al oyente, es decir, respecto a la AR camera, si la AR camera se aleja a una cierta distancia, esta curva llegará a cero.

Dadas las ventajas que presenta lo que a priori puede parecer un problema, no se considera como un inconveniente dicho comportamiento. Aunque dicho esto, en un futuro se podría investigar acerca de una nueva configuración que si pudiese lograr que el marcador solo se reproduzca cuando aparezca en pantalla, y no siga presente cuando se deje de enfocar en el mismo, tal y como se comentará en el apartado 7. LÍNEAS FUTURAS.

En lo que concierne a este proyecto, se han probado distintos métodos, mediante el uso de la función '*DefaultTrackableEventHandler*', y si bien es cierto que se consiguió lograr que el marcador solo se reproduzca cuando aparece en escena, y que cese al desaparecer, se perdía la sincronización de las pistas y no sé conseguía reproducir la partitura con total fidelidad, por lo que se optó por sacar provecho de este comportamiento para que fuese beneficioso en casos como los descritos anteriormente.

Esta aplicación está diseñada para reproducir cualquier partitura que se encuentre en formato MIDI, o para cualquier partitura que se encuentre con sus respectivos instrumentos musicales en fuentes separadas en formato WAV.

Este diseño permite reproducir simultáneamente y con total sincronización tantos instrumentos como se desee, o tantos como requiera la obra musical. En este proyecto se han probado un total de 9 instrumentos musicales, para los que ya están todos sus parámetros configurados. Si se quiere añadir un nuevo instrumento tan solo habrá que seguir los pasos indicados en el apartado 6.1. ANEXO I: Manual de usuario para incorporar un nuevo instrumento. De igual forma, si lo que se desea es añadir una nueva obra musical se deberá proceder como se indica en el apartado 6.2. ANEXO II: Manual de usuario para incorporar una nueva obra musical.

En esta aplicación, se ha desarrollado como ejemplo la banda sonora correspondiente a la introducción de la serie de HBO, Juego de Tronos, producida por el compositor Ramin Djawadi. Se ha conseguido reproducir con total fidelidad esta obra musical ejemplar, compuesta por violín, viola, cello, piano, bombo y bajo, dando un total de 6 instrumentos musicales con los que se puede experimentar cambiando su posición a tiempo real para así recrear una sensación auditiva de sonido 3D única. Se puede apreciar claramente como

los instrumentos posicionados en la izquierda, se escucharán mayoritariamente en el auricular izquierdo, y viceversa. También se puede experimentar con la profundidad del sonido, escuchándose con mayor volumen a medida que se acorta la distancia de la AR Camera con respecto al marcador, o disminuyendo la intensidad del audio conforme se aleja, llegando hasta desaparecer el sonido, si la distancia es considerablemente grande. Para el procesado de audio de esta partitura, se ha utilizado la función script.py modificada tal y como se indica en la página 34, para así poder separar de manera correcta instrumentos como el violín, viola y cello, ya que estos se encuentran en la misma agrupación de instrumentos de la General Midi List Instrument, y con el script original quedarían los tres integrados en el mismo archivo de audio.

También se ha integrado como ejemplo de esta aplicación la canción "*I know you are out there some*" de "*The Moody Blues*". Es una obra prácticamente antagónica a la anterior y gracias al uso de esta obra musical se pueden observar otras agrupaciones de la especificación GM Level 1 tales como el grupo de sintetizadores y maderas. Para el procesado de audio de esta canción se ha utilizado el script script.py original, dando lugar a un total de 5 archivos de audio (bajos, maderas, ensemble, percusión y sintetizadores).

En la siguiente ilustración se muestra la Orquesta RA con sonido 3D reproduciendo el escenario para la obra musical Intro Juego de Tronos (normalizada):

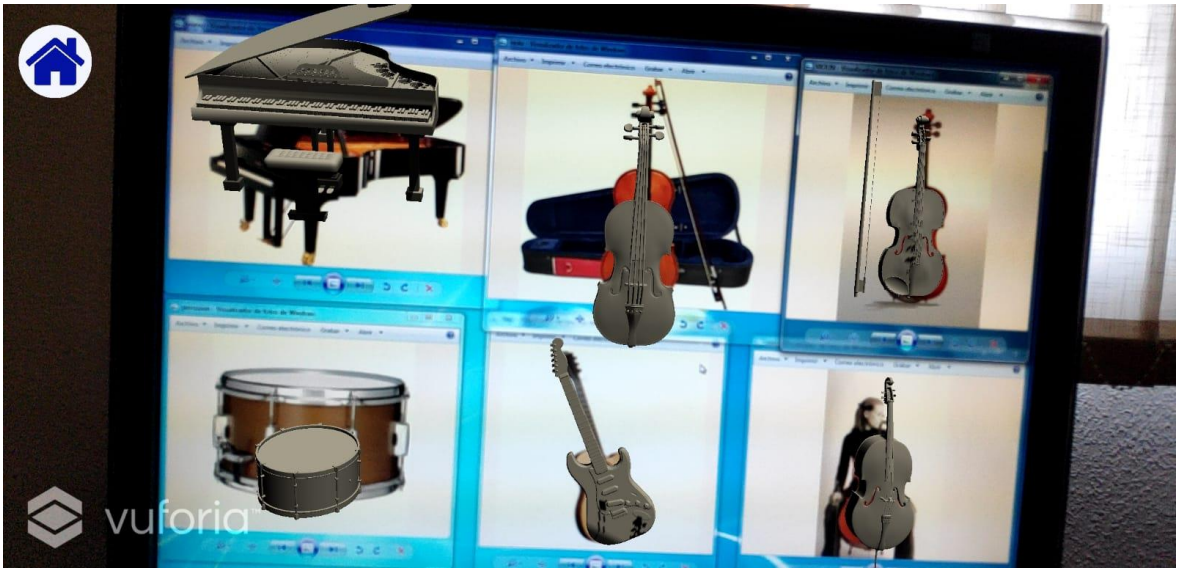


Ilustración 35: Orquesta RA con sonido 3D reproduciendo el escenario para la obra musical Intro Juego de Tronos

Como se puede observar, se reconocen los marcadores sin problemas, aunque esto quedará mejor reflejado en el vídeo demostrativo del funcionamiento de la aplicación. Hay que tener en cuenta que en condiciones de baja luminosidad o por desenfoque inesperado

de la cámara, puede haber fallos a la hora de detectar el target, y si en este intervalo de tiempo en el que no se detecta el target se cambia de posición, puede haber un cambio brusco en el volumen o en la localización de la fuente de audio al ser detectado de nuevo.

En la siguiente ilustración se muestra la distancia máxima a la que los targets pueden ser detectados de manera óptima:

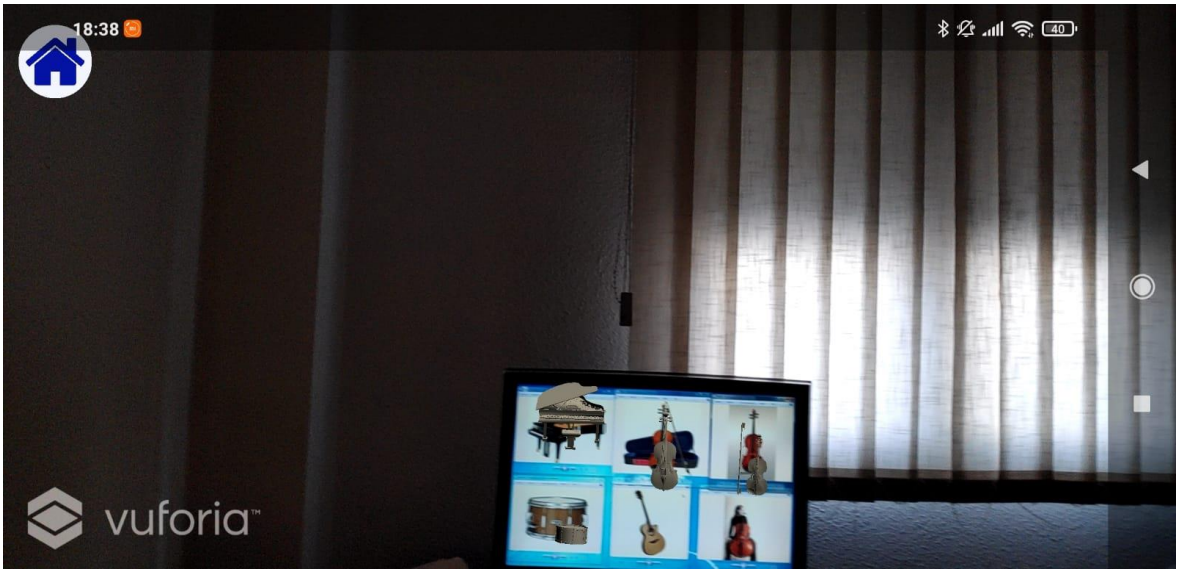


Ilustración 36: Distancia máxima a la que los targets pueden ser detectados de manera óptima

A esta distancia límite, los targets con peor puntuación, según VuforiaEngine, ya no son reconocibles. Además, a esta distancia el sonido queda bastante atenuado, ya que la función AudioListener integrada en la AR Camera (el oyente) se encuentra bastante lejos, ya que ésta decae con una curva dependiente de la distancia (CustomRollOff), lo que dificulta reconocer la localización de las diferentes fuentes de audio.

A continuación, se ilustra la creación de la escena para la obra musical "*I know you are out there some*" de "*The Moody Blues*":

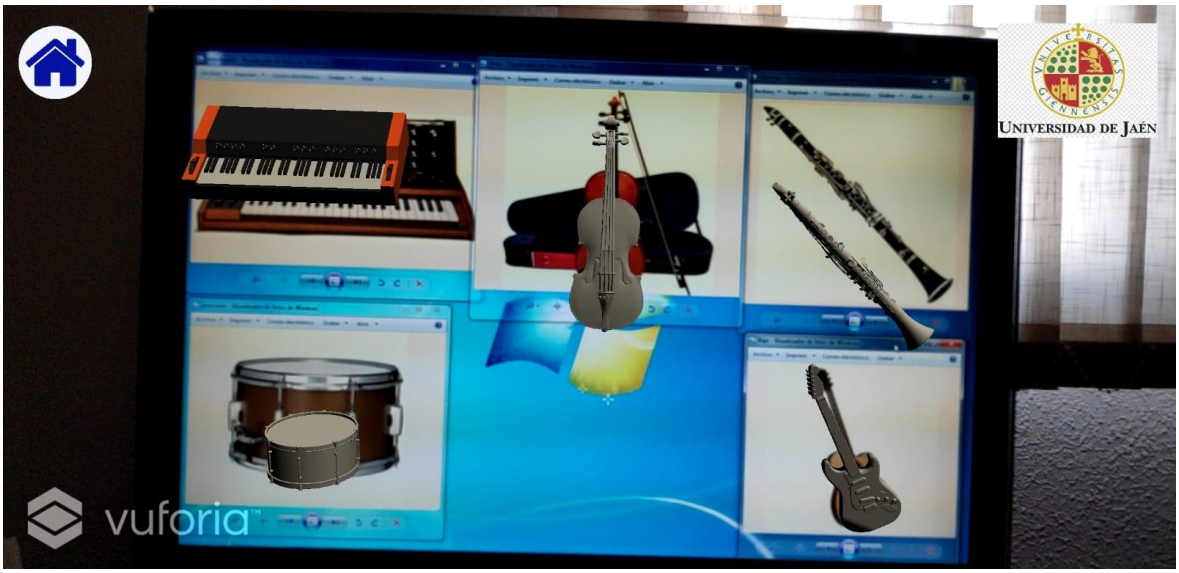


Ilustración 37: Orquesta RA con sonido 3D reproduciendo el escenario para la obra musical de "The Moody Blues"

En este escenario, se aprecia una mayor amplitud en los archivos de audio asociados, debido a que no están normalizados, por lo que el decaimiento de volumen con respecto a la distancia a la que se encuentra la AR Camera es más moderado.

En el vídeo demostrativo que se aloja en Youtube y por el que se puede acceder directamente clicando el botón 'Video Demostrativo' en la aplicación Android, se pueden apreciar mejor las características de este proyecto, pero no todas, ya que al grabar y guardar el sonido del vídeo desde un dispositivo móvil, éste lo almacena todo en mono, aunque se usen auriculares, por lo que en un vídeo grabado se aprecia principalmente el aumento o disminución de volumen dependiendo de si el oyente se encuentra más cerca o más alejado.

Para observar plenamente todas las características y funcionalidades de este proyecto, la mejor opción es probar directamente la aplicación con unos auriculares, ya que así verdaderamente se podrá apreciar el efecto creado por el sonido 3D, que es el núcleo del proyecto, y de este modo, el oyente podrá sumergirse en una experiencia auditiva excepcional.

5. CONCLUSIONES

Como conclusión, creo que se ha logrado cumplir con los objetivos que se plantearon al inicio, obteniendo una aplicación innovadora, práctica y útil, que puede ser utilizada con diversos fines, ya sean comerciales y/o recreativos, y que abre la posibilidad al desarrollo de nuevos productos o proyectos partiendo de esta base sólida.

Se ha conseguido hallar un método eficaz para solventar los problemas de los que se partían, proporcionando una solución pragmática, tras un intenso estudio teórico-práctico que abordó todas las opciones posibles.

He de decir que no fue fácil llevar a cabo este proyecto, debido, primeramente, a mi falta de conocimiento e inexperiencia en los campos de estudio de la realidad aumentada y sonidos polifónicos, pero también a causa de la situación excepcional en la que nos encontramos, que ha provocado una inestabilidad y un caos parcial en la vida de la mayoría de personas.

Tras este largo recorrido explorando dichos campos, me he dado cuenta de todo lo que aún queda por aprender, y doy gracias por la oportunidad que se me brindó al seleccionarme para la realización de este proyecto, ya que me ha abierto la posibilidad de seguir desarrollando estos fundamentos básicos adquiridos, en campos que están creciendo enormemente en la actualidad, y que, incluso, me podrían brindar la posibilidad de conseguir un trabajo en el futuro.

Ha sido todo un reto lograr una aplicación funcional que cumpliera con las características necesarias que se plantearon al inicio del desarrollo de la misma. También ha sido un reto el desarrollo de una memoria que cumpliera con todas las indicaciones. Fue un arduo trabajo de horas y horas redactando, hasta que se obtuvo una lectura comprensible y capaz de abordar punto a punto todos los aspectos del trabajo desarrollado de una manera consistente.

En mi opinión personal, creo que el esfuerzo mereció la pena y que he crecido tras conseguir encontrar una vía para solucionar cada problema que ha ido apareciendo.

6. ANEXOS

6.1. ANEXO I: Manual de usuario para incorporar un nuevo instrumento

Para incorporar un nuevo instrumento dentro de la aplicación desarrollada se deberán seguir los siguientes pasos:

1. Crear una cuenta en Vuforia Engine developer portal, si aun no se dispone de una.
2. Acceder a la opción Target Manager dentro de Develop e introducir el o los instrumentos deseados dentro de una database. En la pestaña Width, se deberá de elegir un tamaño acorde al GameObject descargado. Una vez añadido, se podrá observar una puntuación de uno a cinco dependiendo de lo buena que sea esa imagen como target.

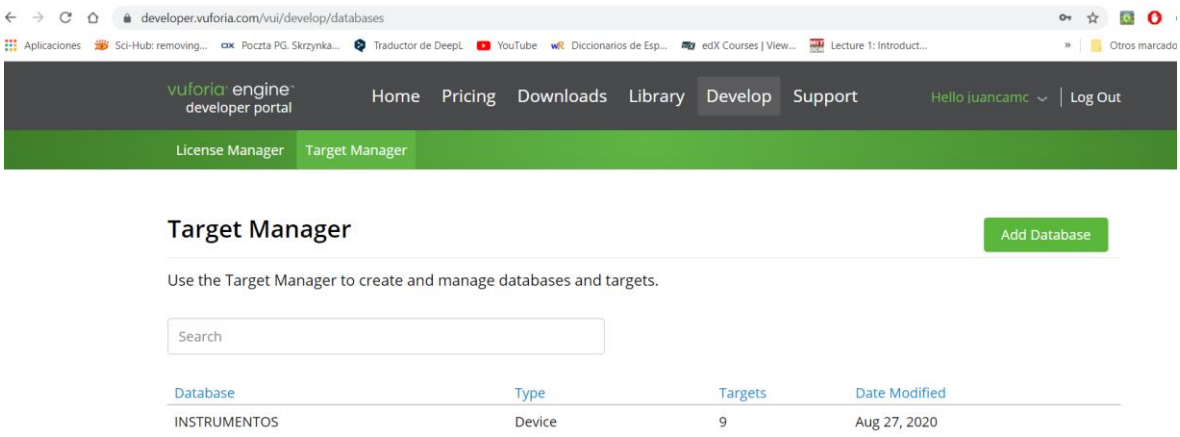


Ilustración 38: Creación de una base de datos para alojar los instrumentos/targets

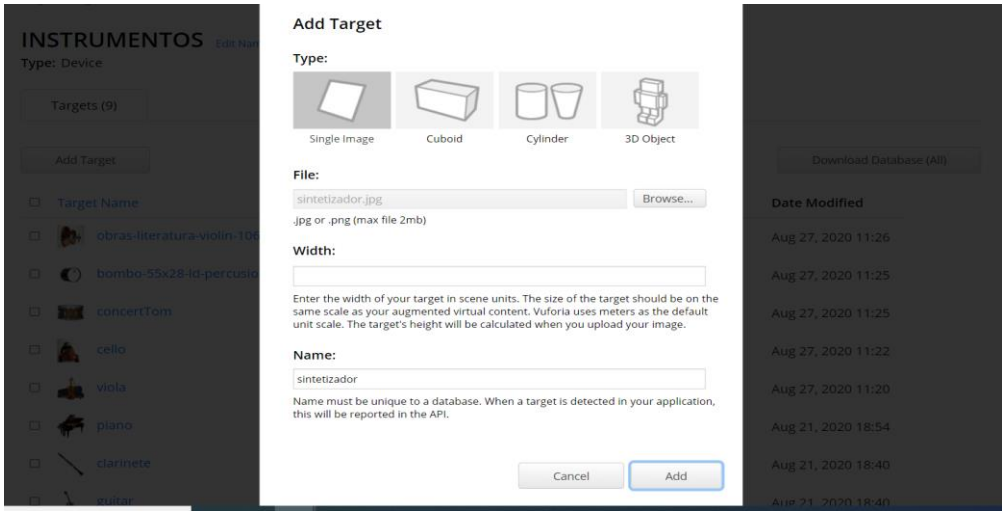


Ilustración 39: Añadir nuevo target a la base de datos

3. Una vez añadida la imagen deseada, se deberá descargar la base de datos e importarla en Unity.

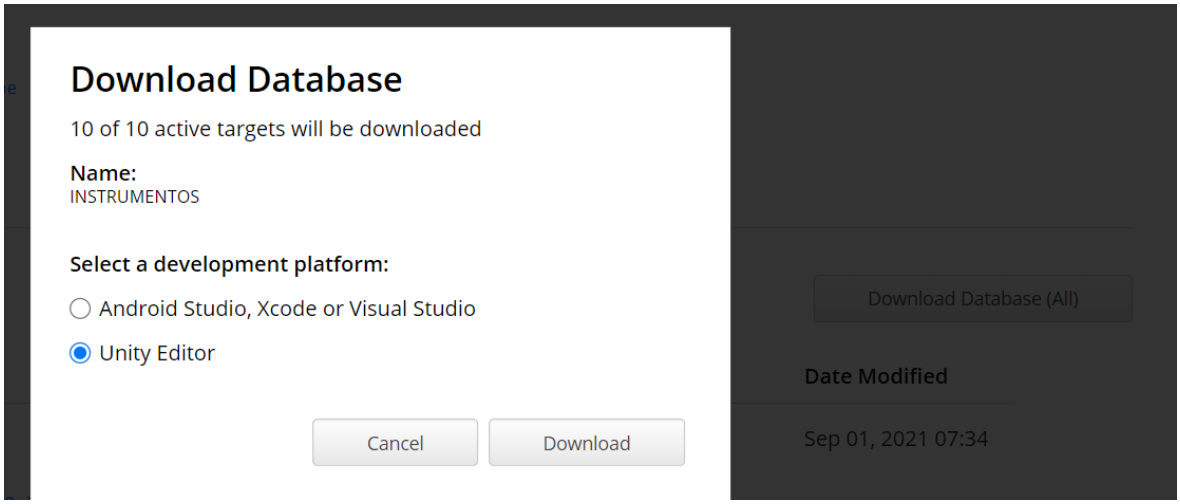


Ilustración 40: Exportación de la base de datos

4. Una vez importada, se creará un Image Target y se le asociará la imagen correspondiente.

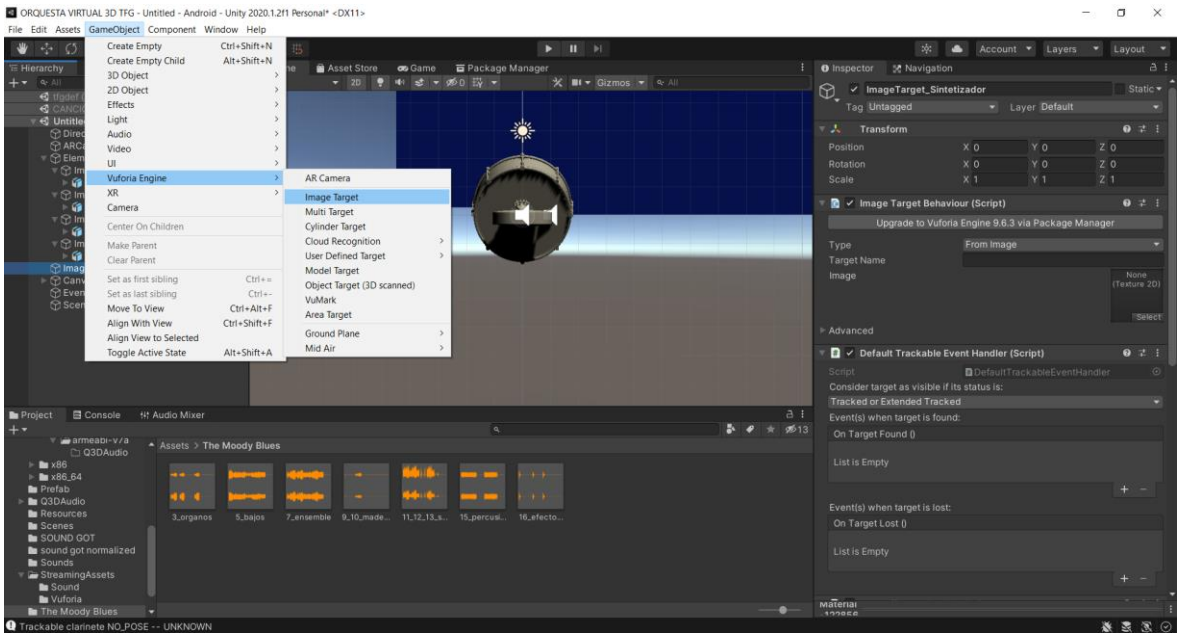


Ilustración 41: Creación de un ImageTarget

5. Se introduce el GameObject.obj previamente descargado en un Asset de Unity y se arrastra hasta el Image Target correspondiente. Normalmente es necesario ajustar

la escala y la rotación del GameObject.obj para que quede acorde al tamaño del Target y en una posición correcta con respecta a la imagen.

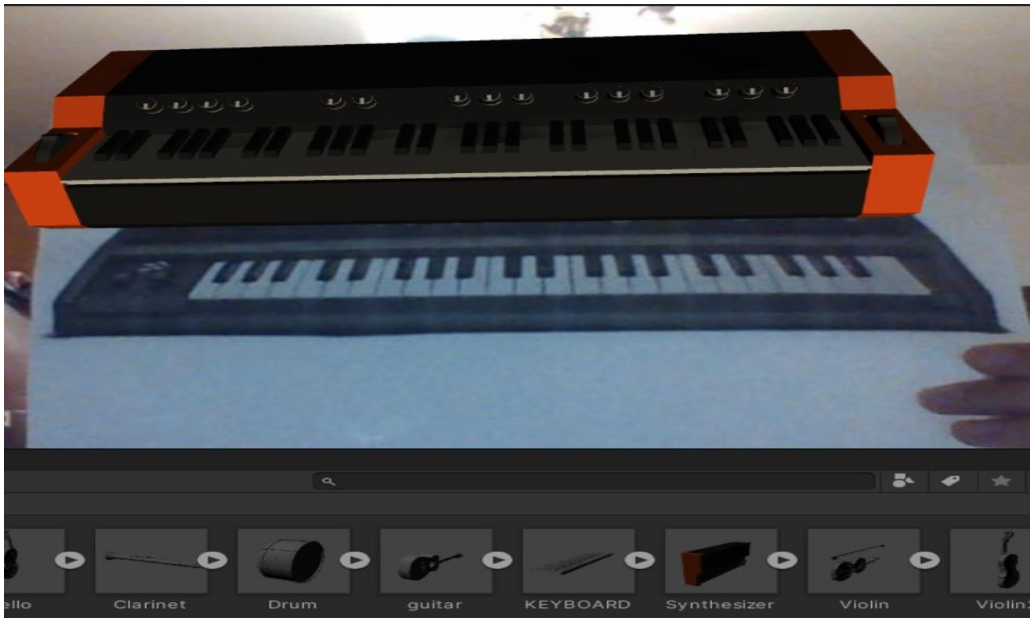


Ilustración 42: GameObject utilizado para representar el sintetizador, asociado a su correspondiente ImageTarget

6. Una vez se realiza la configuración anterior, bastará con copiar la función AudioSource de cualquier otro instrumento y pegarla en el nuevo instrumento (GameObject) para tener los valores perfectamente configurados. Ya solo quedaría cambiar el AudioClip asociado por el que le corresponde y la incorporación del nuevo instrumento quedaría finalizada.

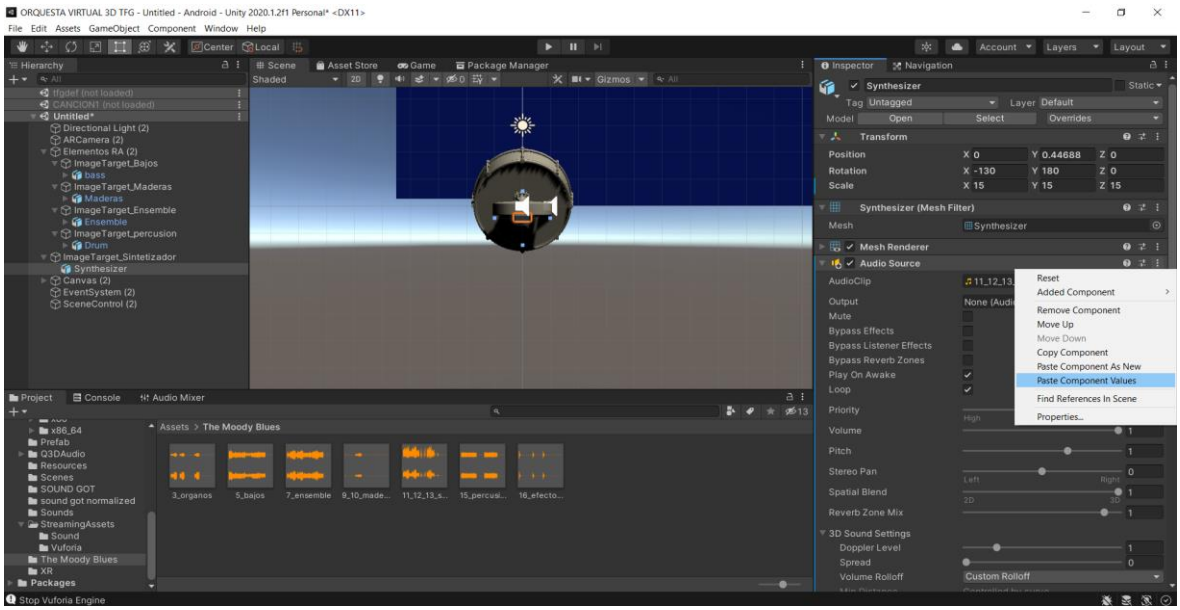


Ilustración 43: Copia de los valores de componentes de la función AudioSource

6.2. ANEXO II: Manual de usuario para incorporar una nueva obra musical

Si se desea incorporar una nueva obra musical al proyecto, deberán seguirse los siguientes pasos:

1. Se escoge y se descarga la obra musical en formato MIDI con especificación General Midi Level 1 (normalmente la mayoría de obras musicales siguen esta especificación por defecto). En este ejemplo, se ha seleccionado la canción *BB King – How Blue Can You Get.*
2. Se introduce y se envía esta obra musical a través del formulario web del servidor Apache. Si todo ha salido correcto, el servidor devolverá la respuesta: '*Archivos generados, Convirtiendo archivos de mid a wav...Archivos convertidos a wav'*.

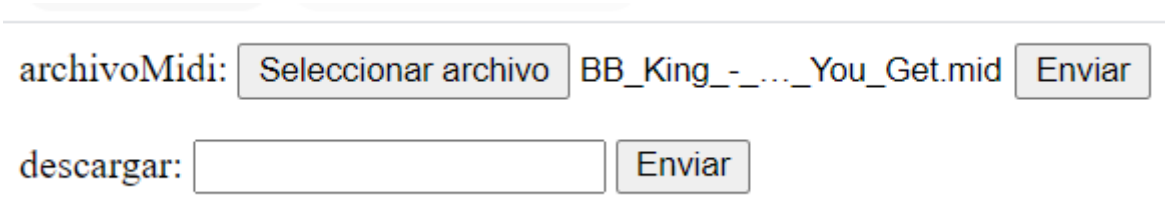


Ilustración 44:Envío de la obra musical a través del formulario web del servidor Apache

3. Se seleccionan todos los archivos de audio generados y se copian a una carpeta en Assets de Unity para una mayor comodidad. La ruta en la que se crean los clips de audio es siempre la misma. Además, si se introduce una obra diferente en el formulario se borrarán los archivos de audio antiguos y se generarán los nuevos, por lo que no habrá ninguna confusión entre archivos de audio de diferentes partituras.

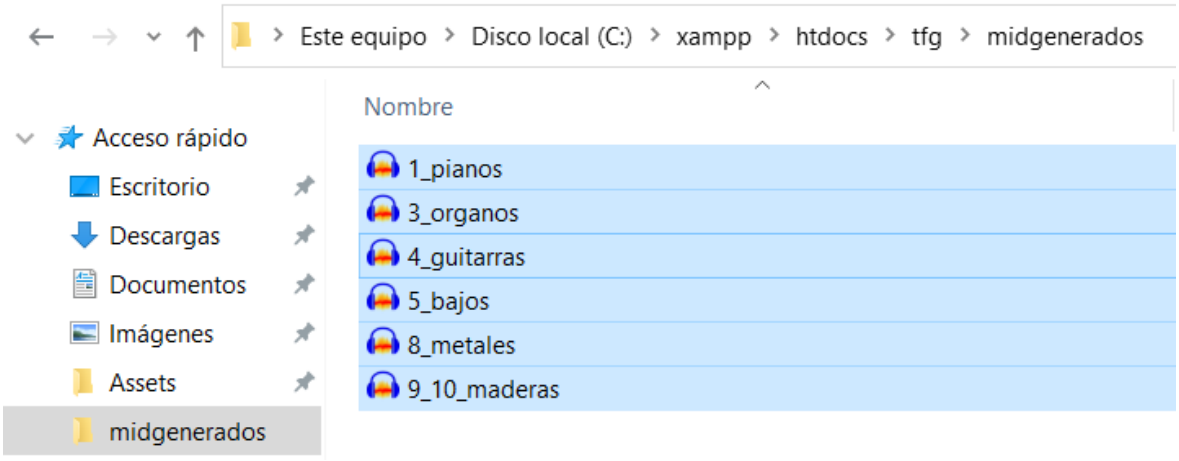


Ilustración 45: Selección de archivos de audio generados por el servidor

4. Una vez se dispongan de todos los archivos de audio que componen la obra musical, se pasará a la modificación de la aplicación en Unity. Para ello se deberá crear una nueva escena.

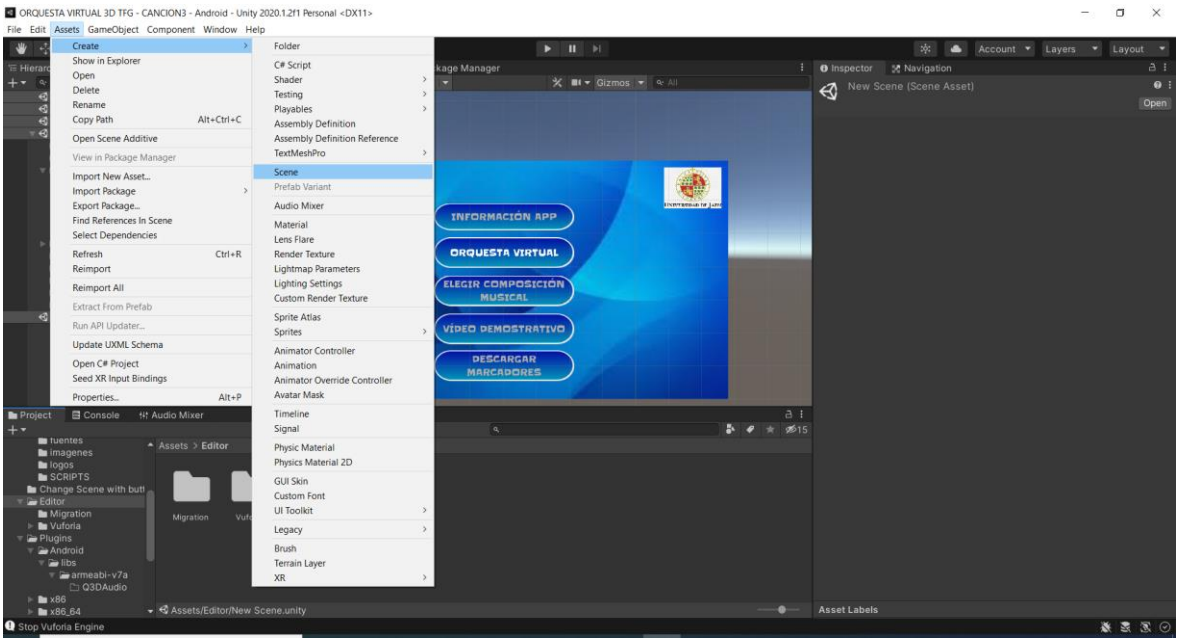


Ilustración 46: Creación de una nueva escena en el entorno de Unity

5. Para una rápida configuración, se podrá optar por copiar el contenido de cualquier otra escena a la nueva escena creada. De esta manera, se ahorrará bastante tiempo ya que no habrá que configurar la AR Camera, ni crear los instrumentos (ImageTarget con su respectivo GameObject asociado) que ya estén incluidos en otra obra musical.

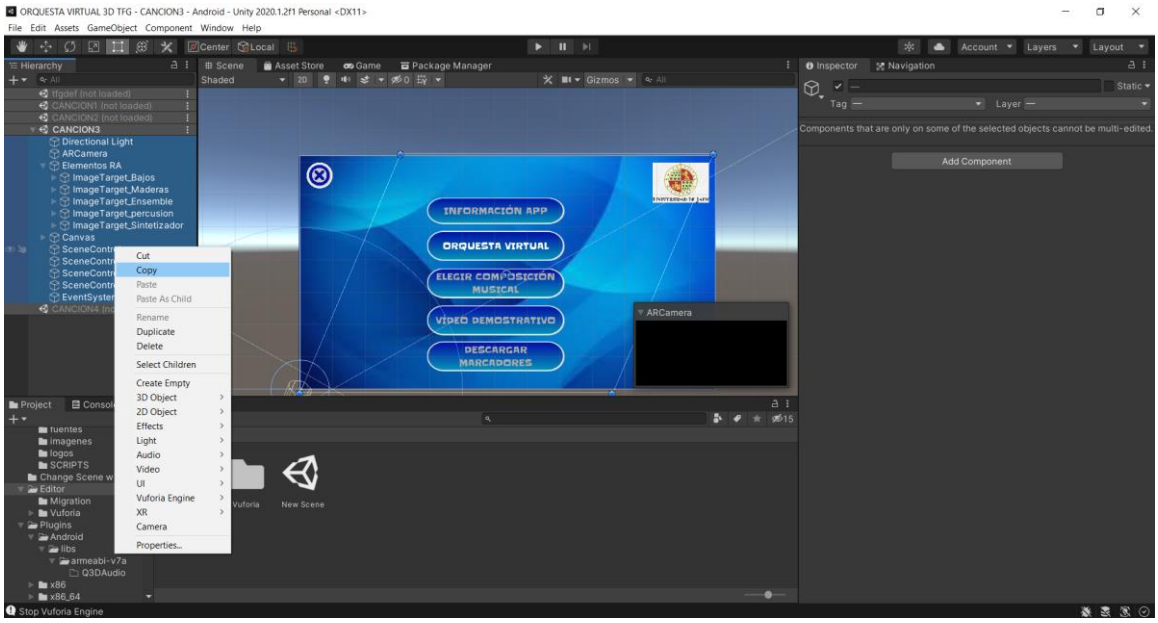


Ilustración 47: Copia de todo el contenido de una escena para incluirlo en la nueva escena

6. Una vez copiado todo el contenido a la nueva escena, se deberán crear los ImageTargets con sus correspondientes GameObjects si no estuvieran todos. Para realizar este proceso se deberán seguir los pasos que se indican en el Anexo I: Manual de Usuario para incorporar un nuevo instrumento.
7. En este proyecto, se ha reservado un botón vacío en el panel Canciones con el fin de ahorrar tiempo si se requiere introducir una nueva obra musical. Si se quiere introducir dicha obra en este botón destinado, tan solo habrá que arrastrar el GameObject SceneControl a las propiedades del botón, y seleccionar la opción *ChangeSceneWithButton.LoadScene*. SceneControl contiene un sencillo script que permitirá cambiar la escena actual por la que contenga la obra musical deseada.

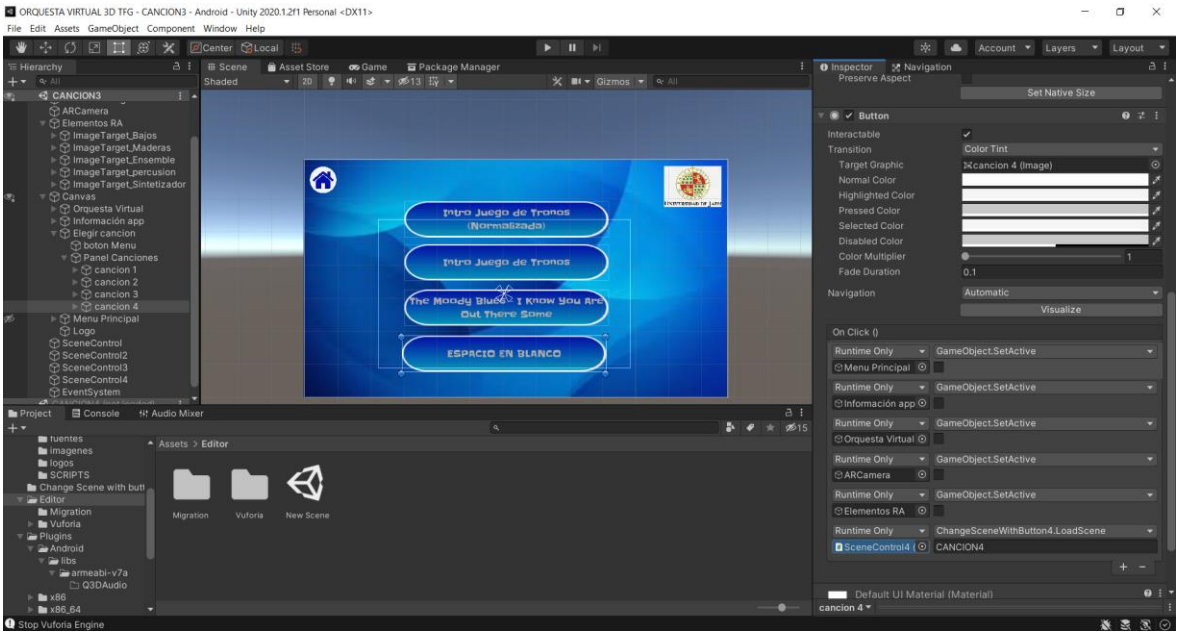


Ilustración 48: Introducir script ChangeSceneWithButton en el botón correspondiente

8. Si se desean introducir nuevas obras musicales sin borrar ninguna de las anteriores, se deberán crear más botones dentro de Panel Canciones. Bastará con copiar los componentes y pegarlos en el nuevo botón creado.

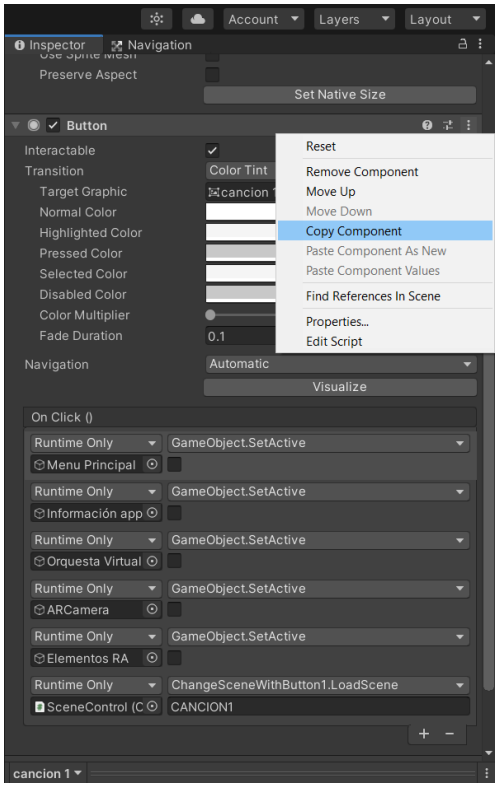


Ilustración 49: Copia de las componentes de un botón

9. Para finalizar la incorporación de la nueva obra musical, tan solo bastará con arrastrar los archivos de audio generados con su correspondiente instrumento, tal y como se indica en el último paso del Anexo I: Manual de Usuario para incorporar un nuevo instrumento.

7. LÍNEAS FUTURAS

A continuación, se proponen líneas futuras para continuar el desarrollo de esta aplicación:

- *Orquesta 3D mediante realidad aumentada en interpretación polifónica en la nube*: Esta línea futura sería la más interesante para la continuación de este proyecto. Tendría básicamente las mismas funcionalidades que este proyecto, pero los archivos de audio MIDI podrían cargarse y procesarse directamente desde la web a la aplicación. Para ello se debería de crear un servidor local o en la nube capaz de comunicarse directamente con la aplicación.

- *Orquesta 3D con sonido polifónico realizada con la herramienta MIDI Tool Kit Pro:* Trataría de continuar con el funcionamiento de esta app, con la diferencia que en vez de tratar audios .wav, podría reproducir audios.mid. Para ello se debería de configurar un escenario 3D en el que se alojasen los diferentes instrumentos de manera flotante, y aunque se tuviera que prescindir de utilizar realidad aumentada, si que se podría desarrollar una aplicación para realidad virtual.

- *Orquesta 3D mediante realidad aumentada con nuevas funcionalidades:* Esta línea futura podría considerarse como una actualización o versión 2.0 de este proyecto. Aquí podrían explorarse vías como la grabación de vídeo y audio 3D a través de la cámara de realidad aumentada, funcionalidades que permitiesen retroceder o adelantar los clip de audio en tiempo real, añadir efectos especiales o filtros a estos audios, así como crear zonas de reverberación para poder simular de manera más realista un escenario o una sala acústica. También se podría explorar la vía para que los instrumentos dejen de reproducirse cuando desaparezca el target, y así poder comprobar si este comportamiento pudiera ser útil para la aplicación.

8. REFERENCIAS

- *Audio Source*, Unity User Manual. 2020. Disponible https://docs.unity3d.com/es/2019.3/Manual/class-AudioSource.html

- CHANG, S., CHEN, W. Does visualize industries matter? A technology foresight of global virtual reality and augmented reality industry. *International Conference on Applied System Innovation (ICASI)*, (2017), 382-385

- COLLIN, R., ELLIS, D. Intuitive analysis, creation and manipulation of MIDI data with pretty_midi. *15th International society for music information retrieval conference*, (2014), 1-2

- *General MIDI*, Wikipedia. 2020. Disponible en https://es.wikipedia.org/wiki/General_MIDI

- HOCKING, J. *Unity in Action, second edition.* Nueva York: Manning, 2018

- LINOWES, J., BABILINSKI, K. *Augmented reality for developers.* Birmingham: Pack Publishing, 2017

- MARTÍNEZ, Marcos, 2020. Ponte los auriculares, relájate y disfruta de las mejores canciones 8D. En Nobbot [en línea]. Disponible en: https://www.nobbot.com/off-topic/sonido-8d-mejores-canciones-8d/

- PÉREZ, Enrique, 2019. Qué es el audio 3D/360 y por qué el sonido holofónico está volviendo a resurgir después de varias décadas eclipsado por el estéreo. En Xataka [en línea]. Disponible en https://www.xataka.com/audio/que-audio-3d-que-sonido-holofonico-esta-volviendo-a-resurgir-despues-varias-decadas-eclipsado-estereo

- *Pretty MIDI*, Github. 2020. Disponible en https://github.com/craffel/pretty-midi

- SINCLAIR, J. *Principles of game audio and sound design: sound design and audio implementation for interactive and immersive media.* Nueva York: Routledge, 2020

- *Síntesis del sonido: el protocolo y el formato MIDI.* Universitat Politècnica de València. Disponible en http://www.disca.upv.es/adomenec/IASPA/tema5/Midi.html

- SMITH, M. *Unity 2018 Cookbook.* Birmingham: Pack Publishing, 2018

- *UnityWebRequest*, Unity Documentation. 2021. Disponible en https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html

- *Unity Android screen recorder*, Github. 2020. Disponible en https://github.com/thanh-nguyen-kim/Unity_Android_Screen_Recorder

- URBINA, Moisés, 2019. SkyView, una aplicación que te hará ver las estrellas. En 4to Mono [en línea]. Disponible en https://www.4tomono.com/mono-sapiens/skyview-una-aplicacion-que-te-hara-ver-las-estrellas/

- *Vuforia engine developer portal*, Vuforia Engine. 2014. Disponible en https://developer.vuforia.com/forum/faq/unity-how-can-i-play-audio-when-targets-get-detected

- *What is augmented reality?* The Franklin Institute. 2021. Disponible en https://www.fi.edu/what-is-augmented-reality

- *3D audio plugin for Unity*, Quallcomm developer network. 2021. Disponible en https://developer.qualcomm.com/software/3d-audio-plugin-unity/quick-start-guide