Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

Universidad de Jaén

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

PROCESADO DE EFECTOS DE AUDIO EN REALIDAD AUMENTADA

Alumno: Luis Ortega Rubio

Tutor: Prof. D. Julio José Carabias Orti

Depto.: Departamento de Ingeniería de Telecomunicaciones

Septiembre, 2024

Universidad de Jaén

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

PROCESADO DE EFECTOS DE AUDIO EN REALIDAD AUMENTADA

Firma autor:

Firma tutor:

MATA CAMPOS RAUL - 26480125H

Firmado digitalmente por MATA CAMPOS RAUL - 26480125H
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-26480125H, givenName=RAUL, sn=MATA CAMPOS, cn=MATA CAMPOS RAUL - 26480125H
Fecha: 2024.09.05 12:11:35 +02'00'

CARABIAS ORTI JULIO JOSE - 53591108G

Digitally signed by CARABIAS ORTI JULIO JOSE - 53591108G
Date: 2024.09.05 12:36:13 +02'00'

Septiembre, 2024

Agradecimientos

Desde que me embarqué en este arduo e intenso camino, mi padre y mi hermana han sido la fuerza que me ha ayudado a seguir día a día, trabajando a pesar de los días oscuros o resultados inesperados. Habéis sido y seréis siempre para mí el mayor ejemplo a seguir. La personificación de la perseverancia, la constancia y el trabajo duro. Vuestra dedicación incansable y vuestra capacidad de superar cualquier obstáculo son fuente de inspiración constante para mí.

Gracias a mi compañero y amigo José Antonio, por haberme hecho sentir capaz de lo imposible hasta cuando yo no creía, en esos momentos complicados de la carrera siendo mi pilar fundamental. Por haber estado a mi lado en todo momento, brindándome tu apoyo incondicional y dedicación incansable día y noche, siempre dispuesto a escucharme, comprenderme y ofrecerme tu compañía cuando más lo necesitaba.

Gracias a Marina por haber sido parte de este camino, por ser una más en cada proyecto que inicio. Siempre a mi lado con tu paciencia, alegría y amor, convirtiéndome en la mejor versión de mí mismo

Gracias a mi madre, mi ángel en el cielo. La persona que me ha llevado a donde estoy. El primer y último pensamiento de mi mente. Si estoy en el lugar que estoy, es gracias a tu infinito amor y sacrificio. Estoy seguro de que tú has sido la razón de mis éxitos. Por todo lo que me has dado y por todo lo que espero que estés viendo. Para que te sientas orgullosa de ver tu rosa crecer.

Índice

Índice de figuras

Resumen

El presente proyecto representa la culminación de un esfuerzo dedicado a la creación de una experiencia única en el ámbito del procesamiento de efectos de audio. Con el uso de Unity como plataforma de desarrollo y la integración de la tecnología de Realidad Aumentada (RA) a través de Vuforia, se ha logrado diseñar una mesa de mezclas en un entorno tridimensional.

Esta aplicación desarrollada para Android, permite crear mezclas musicales mediante el uso de tarjetas. Cada una de las tarjetas o targets funcionarán como un QR, mediante la cámara de nuestro teléfono, tendremos que enfocarlos detenidamente para que se apliquen sus funcionalidades.

Tendremos 3 tipos de targets: Fuentes de sonido, efectos y variaciones de estos. Podremos seleccionar el conjunto de fuentes que queremos cargar desde el menú principal de la pantalla, y mediante giros y posición de los targets, crear nuestra propia mezcla musical.

A lo largo de este documento, se detallarán los procesos y desafíos enfrentados durante el desarrollo de esta aplicación, así como las soluciones encontradas y las lecciones aprendidas. Además, se presentarán los resultados obtenidos y se discutirán las posibles aplicaciones y futuras direcciones para esta tecnología innovadora en el ámbito de la música y el entretenimiento.

Abstract

The present project represents the culmination of a dedicated effort to create a unique experience in the field of audio effects processing. Using Unity as the development platform and integrating Augmented Reality (AR) technology through Vuforia, we have successfully designed a mixing console in a three-dimensional environment.

This application, developed for Android, allows users to create musical mixes using cards. Each of the cards or targets functions like a QR code; by carefully focusing on them with our phone's camera, their functionalities are applied.

There are three types of targets: sound sources, effects, and variations of these. We can select the set of sources we want to load from the main menu on the screen, and by rotating and positioning the targets, we can create our own musical mix.

Throughout this document, the processes and challenges encountered during the development of this application will be detailed, along with the solutions found and the lessons learned. Additionally, the results obtained will be presented, and potential applications and future directions for this innovative technology in the field of music and entertainment will be discussed.

1. Introducción

Actualmente, la tecnología está transformando la manera en que interactuamos con la música y los efectos de audio. Cada vez son más las personas que buscan herramientas accesibles y creativas para explorar y manipular sonidos. Sin embargo, las herramientas tradicionales de procesamiento de audio pueden ser complejas y poco intuitivas, lo que limita el potencial creativo de los usuarios y su capacidad para experimentar con nuevos efectos y técnicas.

Para abordar esta necesidad, se desarrollará una aplicación en Unity que utilice la Realidad Aumentada (RA) a través de Vuforia para crear una mesa de mezclas tridimensional. Esta herramienta permitirá a los usuarios manipular diversos parámetros de efectos de audio de manera instintiva y visualmente atractiva, utilizando tarjetas físicas que representan diferentes sonidos y efectos. Al girar y mover estas tarjetas, se podrán ajustar parámetros como la frecuencia o el retardo del chorus entre otros, de manera precisa y directa.

Desde la aparición de las primeras mesas de mezclas en estudios de grabación alrededor de 1947, estas herramientas han evolucionado significativamente, convirtiéndose en elementos esenciales para la producción musical y el entretenimiento. Sin embargo, su complejidad y costo han sido barreras para muchos usuarios. La aplicación propuesta pretende democratizar el acceso a estas tecnologías, ofreciendo una solución innovadora y accesible. [1]

Figura 1: Mesa de mezclas en estudio de radio

El proyecto se centra en ofrecer una experiencia de usuario fluida y envolvente, aprovechando las capacidades de la RA para transformar la manera en que los usuarios crean y manipulan la música. La combinación de Unity y Vuforia permite superar las barreras tradicionales entre el mundo físico y digital, abriendo nuevas posibilidades creativas en el campo del procesamiento de audio. Esta herramienta no solo permite una interacción más natural y atractiva, sino que también proporciona una comprensión más profunda y práctica de los procesos de manipulación de audio.

Durante el desarrollo de este trabajo, se explorarán los métodos y obstáculos superados en la creación de esta aplicación, junto con las estrategias implementadas para resolverlos y las experiencias adquiridas. Asimismo, se expondrán los resultados alcanzados y se analizarán las potenciales aplicaciones y futuras tendencias de esta tecnología innovadora en el campo de la música y el entretenimiento.

1.1 Objetivos

El objetivo principal de este proyecto es desarrollar un sistema de procesamiento de efectos de audio, integrando la tecnología de Realidad Aumentada para crear una experiencia interactiva y envolvente en la manipulación de efectos de sonido.

Además, la RA nos será útil para proporcionar una nueva dimensión a la creación musical, permitiendo a los usuarios aprender con la mesa de mezclas y los efectos de sonido en su entorno. Esta integración se realizará mediante el uso de Unity como plataforma de desarrollo y la implementación de algoritmos en C# para el procesamiento de efectos.

Por otro lado, se busca facilitar el acceso y aprendizaje de los efectos de audio mediante el diseño de una interfaz de usuario intuitiva y tarjetas atractivas, que permitan a los usuarios aplicar los efectos de manera sencilla. Esto incluirá la creación de elementos gráficos y visuales para representar los efectos de audio y sus parámetros en el entorno de Realidad Aumentada.

A partir de estos objetivos generales, se han establecido los siguientes objetivos detallados:

- Analizar, definir y crear los efectos de audio que usaremos en la aplicación.
- Crear una interfaz de usuario en Unity para la manipulación de efectos de audio
- Programar algoritmos en C# para la implementación de estos efectos en Unity y para su modificación.
- Integrar la funcionalidad de Realidad Aumentada en la aplicación, permitiendo a los usuarios interactuar con la mesa de mezclas y los efectos en su entorno.
- Crear elementos gráficos y visuales para representar los efectos de audio.

1.2 Organización de la memoria

En el Capítulo 1 de esta memoria se presentará la Introducción. Donde se hablará de cuáles han sido los objetivos principales a cumplir de este proyecto, a partir de los que se detallarán los objetivos de manera más específica.

En el Capítulo 2 se analizarán los fundamentos de la realidad aumentada en aplicaciones musicales, con ejemplos actuales y utilidades.

En el Capítulo 3 se centrará en el Estado del Arte. Para comprender cómo ha evolucionado el concepto de mesa de mezclas. Sus orígenes, las funciones básicas que posee y la inspiración del proyecto en Reactable.

En el Capítulo 4 se comentará sobre los materiales y métodos seguidos. La decisión de porque usar Unity junto con Vuforia. Los primeros pasos en el software y demos. La creación de la escena y audio. El uso de AudioMixer y la implementación de los efectos. Las pruebas, solución de los problemas y el diseño de las tarjetas finales.

En el Capítulo 5 se establecerán las conclusiones. El aspecto final de la aplicación y el resultado final del código y funcionamiento.

En el Capítulo 6 se mostrarán los planos y anexos. Donde se podrá ver el diseño corporativo y el manual de usuario.

2. Fundamentos de la realidad aumentada

A lo largo de este capítulo, se explorarán los fundamentos de la realidad aumentada, analizando su historia por completo. Sus inicios, los primeros inventos basados en esta tecnología, sus autores y sus aplicaciones en la actualidad reflejado en el ámbito musical

2.1 Contexto, antecedentes y actualidad

La Realidad Aumentada es una tecnología emergente que combina elementos virtuales con el entorno real, proporcionando una experiencia interactiva en tiempo real. Para comprender los fundamentos de la RA, es crucial explorar su contexto histórico, sus antecedentes tecnológicos y su situación actual. [2]

Contexto y antecedentes

La primera mención del concepto de RA se remonta a 1901, cuando el escritor Frank L. Baum imaginó unas gafas electrónicas capaces de mostrar información adicional sobre las personas en su novela "The Master Key". Este concepto, aunque visionario, se adelantó a su tiempo y no se materializó tecnológicamente hasta varias décadas después.

La primera implementación tecnológica de una experiencia similar a la RA fue realizada por Morton Heilig en 1957 con su "Sensorama", un dispositivo que ofrecía una experiencia multisensorial combinando elementos visuales, sonoros y olfativos. Aunque no era exactamente RA, el Sensorama sentó las bases para futuras exploraciones en esta área al integrar múltiples sentidos para crear una experiencia inmersiva.

Figura 2: Sensorama

En 1973, Myron W. Krueger desarrolló la primera instalación interactiva que mezclaba cámaras de vídeo con sistemas de proyección para crear un entorno interactivo sensible a los movimientos del usuario. Este desarrollo, aunque primitivo, marcó un hito importante en la historia de la RA.

El término "Realidad Aumentada" fue acuñado por Tom Caudell, un investigador de Boeing, en la década de 1990. Caudell utilizó este término para describir una aplicación que ayudaba a los ingenieros a ensamblar cableados complejos proyectando instrucciones directamente sobre el entorno de trabajo real. Esta fue una de las primeras aplicaciones prácticas de la RA en la industria. [3]

Actualidad

Hoy en día, la RA ha evolucionado significativamente gracias a los avances en tecnologías de computación, sensores y dispositivos móviles. Los smartphones modernos, equipados con potentes procesadores, cámaras, GPS y sensores de movimiento, se han convertido en herramientas clave para la RA, permitiendo que esta tecnología sea accesible a un público mucho más amplio.

La RA se ha implementado en una variedad de sectores, incluyendo la medicina, la industria, el entretenimiento y el comercio. En la medicina con la planificación de cirugías gracias a EchoPixel. En herramientas como la app de medición de Apple que permiten a los usuarios medir objetos y espacios con precisión utilizando la cámara de sus dispositivos móviles, demostrando la utilidad de la RA en tareas cotidianas. [4]

Figura 3: Aplicación de Apple: Medición

La RA ha recorrido un largo camino desde sus inicios conceptuales en la literatura hasta convertirse en una tecnología real. La combinación de avances tecnológicos y el acceso generalizado de los teléfonos móviles han llevado a la RA a una nueva era.

2.2 Realidad aumentada en aplicaciones musicales

La RA ha emergido como una herramienta poderosa y flexible en el ámbito de la educación musical. La capacidad de integrar elementos virtuales en el entorno real

permite interactuar con la música de nuevas maneras. Siendo una de las aplicaciones más destacadas en este campo el uso de pianos virtuales y partituras virtuales basadas en RA.

Figura 4: Metas Quest 3 en la aplicación PianoVision

Con la introducción de dispositivos avanzados como Apple Vision Pro y Meta Quest, la RA ha alcanzado nuevas alturas en el campo de la música. Estas tecnologías permiten superponer notas musicales y guías de aprendizaje directamente en el entorno real del usuario.

Aplicaciones que pueden proyectar las teclas de un piano sobre una superficie plana, mostrando las notas correctas a medida que el usuario toca. Dando lugar a una nueva perspectiva del aprendizaje, permitiendo que los estudiantes practiquen en cualquier lugar sin necesidad de un instrumento. [5]

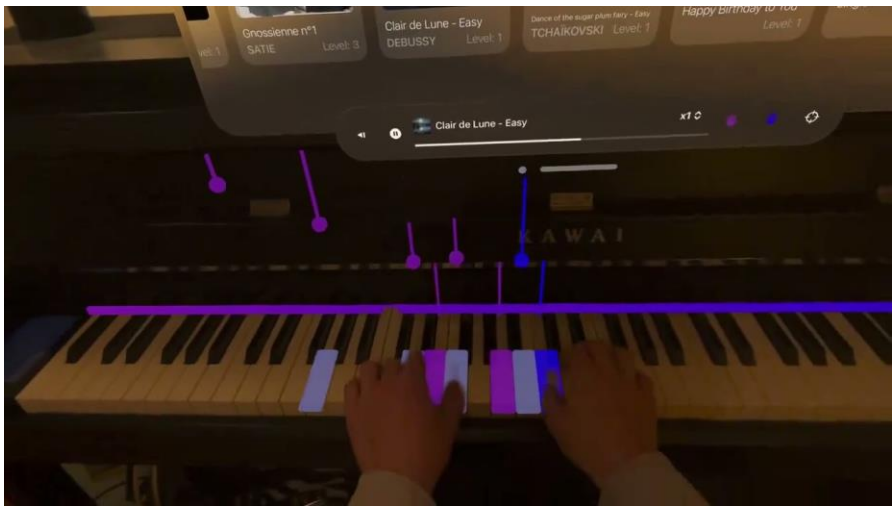


Figura 5: Partituras con Apple Vision Pro

Además, estudios recientes han demostrado que el uso de RA en la educación musical aumenta significativamente la motivación de los estudiantes. La interactividad y el componente visual atractivo de la RA hacen que el aprendizaje sea más envolvente y menos intimidante para los principiantes.

Además, al ser una dinámica diferente a la que los alumnos no suelen estar acostumbrados, hace que estos muestren más atención consiguiendo así ampliar sus conocimientos. [6]

Las aplicaciones en el entorno de la RA no solo se basan en el aprendizaje musical. Existen aplicaciones como StageVerse que ofrecen experiencias de conciertos en realidad aumentada, permitiendo a los usuarios ver actuaciones en vivo y grabadas de artistas en un entorno AR. Utilizando dispositivos como los mencionados anteriormente de Apple y Meta, los usuarios pueden disfrutar de conciertos interactivos, donde se superponen elementos visuales sobre el entorno real. No solo mejora la experiencia de los conciertos en vivo, sino que también permite una mayor accesibilidad a eventos musicales. [7]

Figura 6: App StageVerse

Hay más aplicaciones, como AR Synth que permite a los usuarios crear y experimentar con sonidos y música utilizando sintetizadores en realidad aumentada. La aplicación coloca diferentes módulos de sonido en el entorno real del usuario, permitiendo la manipulación en tiempo real de diversos parámetros de sonido. Esta herramienta es especialmente útil para músicos y productores que buscan una manera innovadora de crear música, combinando elementos visuales y sonoros. Hablando específicamente de sus características encontramos: [8]

- Generadores de sonido, como osciladores y sintetizadores, que permiten crear una amplia variedad de tonos. Junto con multitud de efectos como reverb, delay y distorsión, que se pueden aplicar en tiempo real a los sonidos que se están generando.
- Incluye sintetizadores analógicos, pianos, cuerdas y percusiones. Cada instrumento tiene controles específicos para modificar su timbre y carácter. Además, los usuarios pueden ajustar parámetros como la frecuencia, la envolvente y el filtro para personalizar el sonido.
- Los usuarios pueden grabar sus sesiones de creación musical directamente en la aplicación, lo que permite revisitar ideas y desarrollarlas. Estas grabaciones se pueden exportar en formatos comunes (como WAV o MP3) para compartir en redes sociales o plataformas de música.

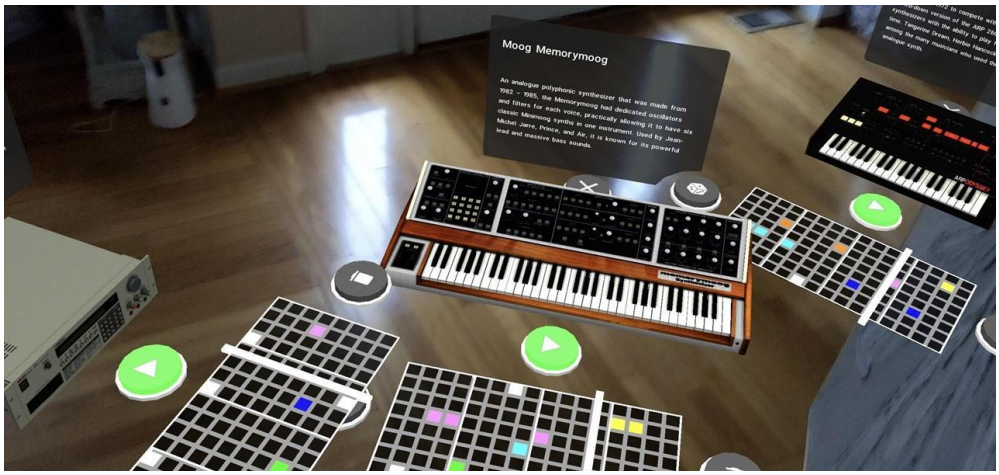


Figura 7: App AR Synth

Profundizando en más aplicaciones musicales, encontramos AR Pianist**,** es una aplicación que utiliza la realidad aumentada para ofrecer una experiencia inmersiva en la que los usuarios pueden ver a pianistas virtuales tocando música en cualquier piano físico o superficie plana que tengan a su disposición. Si hacemos un análisis exhaustivo podemos encontrar en esta app funciones como:

- Mostrar un pianista virtual que toca el piano en tiempo real. Usando la cámara del dispositivo, la aplicación proyecta un pianista en 3D en el entorno. Lo que permite que cualquier escenario se convierta en el escenario. Tomando el entorno la posición y la luz haciendo un ajuste realista.

- Amplia Biblioteca de Piezas: La aplicación incluye una variedad de piezas clásicas y populares, desde composiciones de Chopin y Beethoven hasta canciones más modernas.

Figura 8: AR Pianist pianista en entorno

- Nivel de Dificultad: Permite ajustar el nivel de dificultad, mostrando la interpretación a diferentes velocidades para que los usuarios puedan aprender las piezas más fácilmente.

- Incluye funciones educativas, como tutoriales que desglosan las piezas para los usuarios que quieran aprender a tocar el piano.
- Sincronización con MIDI para sincronizarse con un piano MIDI, permitiendo una experiencia de aprendizaje aún más interactiva.

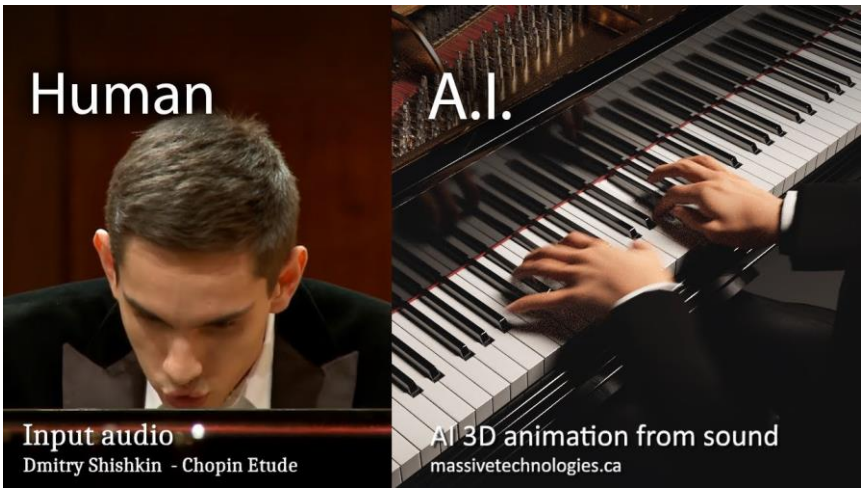


Figura 9: AR Pianist Chopin

Al final, esta aplicación representa una manera ideal para aprender sobre el piano o simplemente para escuchar música. Y todo de una manera visual en nuestro entorno físico con la realidad aumentada.

3. Estado del arte

3.1 Orígenes y evolución de las mesas de mezclas

El desarrollo de las mesas de mezclas de audio ha sido fundamental en la historia de la producción sonora y la ingeniería de audio. Este apartado explora en detalle los orígenes y la evolución de estas herramientas esenciales, desde los primeros experimentos con el triodo hasta la sofisticación de las mesas de mezclas digitales en la era contemporánea.

El triodo, inventado por Lee De Forest en 1906, permitió la amplificación de señales de audio y marcó el inicio de la electrónica aplicada al sonido. Las primeras mesas de mezclas analógicas fueron simples dispositivos diseñados para combinar señales de micrófonos y otros dispositivos de audio en un solo canal de salida. Estas consolas tempranas utilizaban válvulas y componentes electrónicos básicos para manejar la mezcla y el enrutamiento de señales, estableciendo los fundamentos para la producción de audio profesional.

Figura 10: Triodo

En sus inicios, las mesas de mezclas eran dispositivos relativamente simples, diseñados para combinar señales de audio de diferentes fuentes, como micrófonos y grabadores de audio, en un solo canal de salida. Estas primeras consolas analógicas, con sus controles físicos y limitadas capacidades de procesamiento, sentaron las bases para la producción de audio profesional y establecieron los estándares para la mezcla y el enrutamiento de señales.[9]

Figura 11: Mesa de mezclas analógica en estudios de la BBC.

A medida que la tecnología de grabación y reproducción de audio avanzaba, las mesas de mezclas evolucionaron para adaptarse a las nuevas demandas del mercado. En la década de 1950, con el advenimiento del rock 'n' roll y la creciente popularidad de la música grabada, las consolas de audio se convirtieron en elementos esenciales para los estudios de grabación, permitiendo a los ingenieros de sonido manipular y modificar el sonido de manera creativa.

La introducción de la tecnología digital en la década de 1980 marcó un punto de inflexión en la evolución de las mesas de mezclas. Las consolas digitales ofrecían una mayor flexibilidad y capacidad de procesamiento que sus predecesoras analógicas, lo que permitía a los ingenieros de sonido realizar cambios y ajustes en tiempo real y almacenar configuraciones predefinidas para su uso posterior. Además, la integración de interfaces de usuario gráficas y pantallas táctiles simplificó el proceso de mezcla y proporcionó a los usuarios un mayor control sobre su entorno de trabajo.

Figura 12: Mesa de mezclas digital Allen & Heath Qu-32.

Las mesas de mezclas digitales son la norma en la industria de la producción de audio, desde estudios de grabación profesionales hasta conciertos en vivo y emisiones de televisión. Estas consolas ofrecen una amplia gama de características y funcionalidades, incluyendo ecualización paramétrica, compresión multibanda, efectos de procesamiento de señales y enrutamiento flexible de canales. Además, la conectividad en red y la integración con software de grabación y edición de audio han

ampliado aún más las capacidades de estas poderosas herramientas, permitiendo crear y manipular audio con una precisión y flexibilidad sin precedentes. [10]

3.2 Funciones básicas de una mesa de mezclas

Si queremos diseñar una mesa de mezclas en RA, primero debemos conocer cuáles son sus funciones básicas que las diferencian y cuál es el proceso de funcionamiento que sigue. Queda esquematizado en la siguiente imagen:

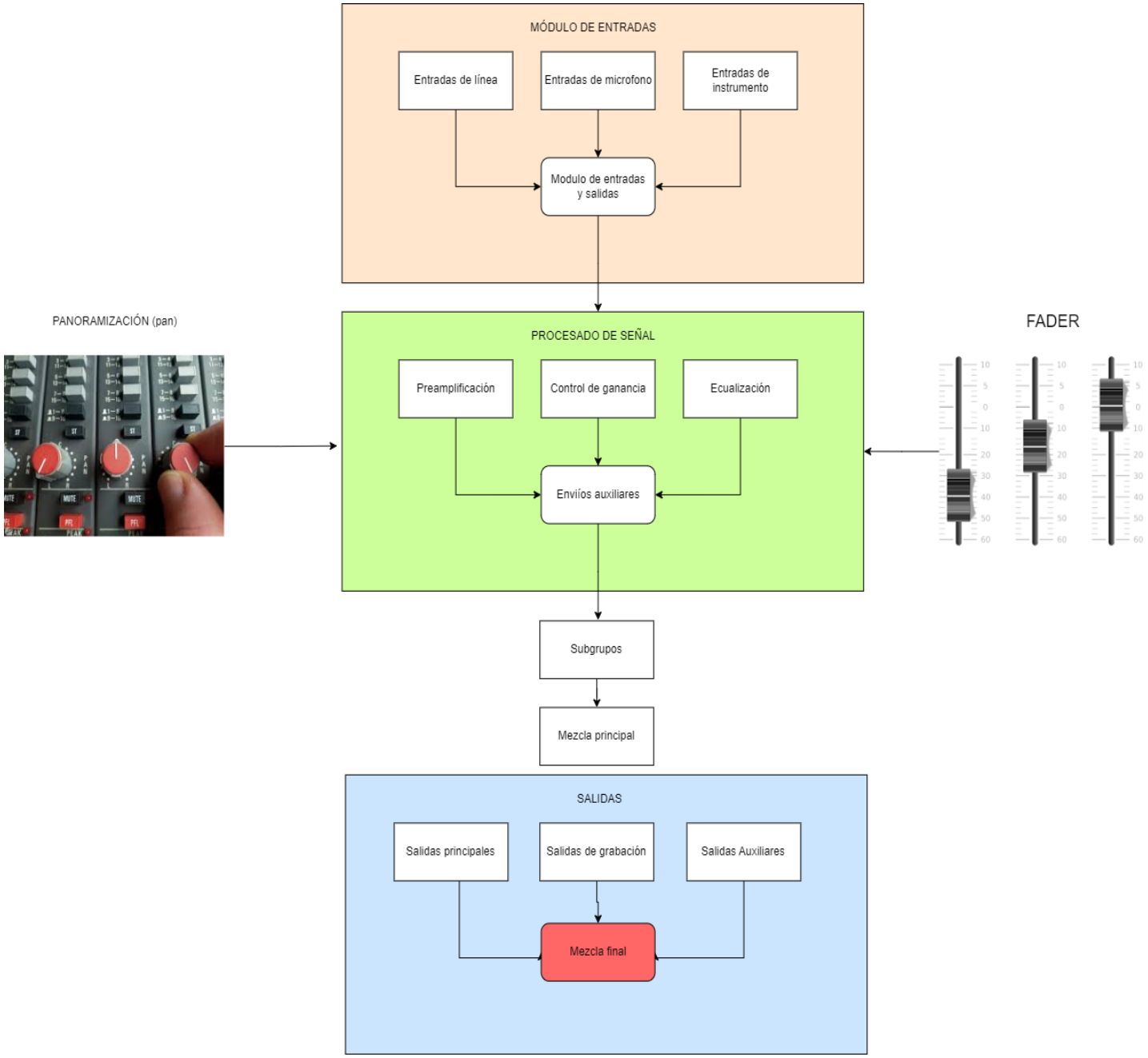


Figura 13: Flujo de una mesa de mezclas

Una vez definidas, podremos transformarlas al entorno de Unity e imitarlas. Las funciones anteriormente ilustradas son: [11]

1. Control de canales**:** Permite la gestión individual de múltiples fuentes de audio, como micrófonos y dispositivos de reproducción.
2. Control de ganancia: Cada canal de entrada de la mesa de mezclas cuenta con un control de ganancia que permite ajustar el nivel de la señal.
3. Ecualización: La ecualización es una función esencial que permite ajustar las frecuencias de la señal de audio en cada canal. Mediante controles de ecualización para las bandas de frecuencias bajas, medias y altas.
4. Paneo: El control de panning permite asignar la posición de cada señal de audio en el campo estéreo, moviéndola hacia la izquierda, derecha o centro.
5. Envíos y retornos de efectos: Las mesas de mezclas suelen tener buses de envío que permiten dirigir la señal de uno o más canales hacia procesadores de efectos externos. Los retornos de efectos integran la señal procesada de vuelta en la mezcla.
6. Mezcla y subgrupos: Las señales de audio de los diferentes canales se combinan en la sección de mezcla de la consola, con la posibilidad de crear subgrupos
7. Salida principal: La salida principal de la mesa de mezclas envía la mezcla final hacia sistemas de amplificación, altavoces o dispositivos de grabación.
8. Monitoreo: Las mesas de mezclas proporcionan opciones de monitoreo para que los usuarios puedan escuchar y controlar la mezcla en tiempo real.

Figura 14: Controles e indicadores de una mesa de mezclas

A rasgos generales, estas son las funciones principales de una mesa de mezclas. Intentaremos replicarlas dentro de las posibilidades que nos ofrece Unity pero dándole

la perspectiva de realidad aumentada y la facilidad de manejo. Para que pueda usarla desde un ingeniero de sonido hasta un estudiante para aprender sobre los efectos y sus aplicaciones

3.3 Reactable como inspiración.

Reactable es una innovadora interfaz musical basada en una mesa de mezclas táctil y tangible que ha cautivado a músicos y entusiastas de la tecnología desde su creación a principios de los años 2000. Diseñada por el equipo de investigación en música interactiva de la Universidad Pompeu Fabra en Barcelona, Reactable se ha convertido en un ícono en la convergencia entre música, arte y tecnología.

Reactable es esencialmente una mesa de mezclas interactiva que permite a los usuarios crear música de manera colaborativa y experimental. Consiste en una superficie circular transparente que se utiliza como pantalla táctil, sobre la cual se colocan diferentes bloques y fichas llamadas "tangibles". Estos tangibles representan distintos elementos musicales, como generadores de sonido, filtros, efectos, bucles, entre otros. Al mover y combinar estos elementos en la mesa, los usuarios pueden manipular y modificar el sonido en tiempo real, creando composiciones musicales de forma intuitiva y estimulante. [12]

Figura 15: Mesa de Reactable

La ciencia detrás de Reactable se basa en principios de síntesis de sonido, procesamiento digital de señales y diseño de interfaces de usuario. Utiliza algoritmos y técnicas avanzadas para convertir los movimientos físicos de los usuarios en acciones musicales.

Reactable funciona mediante la detección y seguimiento de los tangibles colocados sobre la mesa, utilizando tecnologías como la computación visual y la detección de objetos. Cada tangible representa un elemento musical con propiedades específicas, como tono, volumen, efectos, etc. Al manipular y combinar estos tangibles, los usuarios pueden crear y modificar sonidos de manera creativa y fluida. [13]

Figura 16: Funcionamiento e interfaz de Reactable

La inspiración en Reactable radica en la exploración de nuevas formas de interactuar con la música a través de “targets”. Al igual que Reactable, nuestro proyecto busca ofrecer una experiencia única y envolvente en el procesamiento de efectos de audio, utilizando tecnologías de realidad aumentada. Nos inspiramos en el enfoque experimental de Reactable para desarrollar una mesa de mezclas espacial que permita explorar y manipular efectos de sonido de manera creativa.

4. Materiales y Métodos

4.1 Unity y Vuforia

La elección de Unity y Vuforia como herramientas de desarrollo para este proyecto se basa en su idoneidad para potenciar la creación de aplicaciones de RA.

Unity, como plataforma ampliamente utilizada en el desarrollo de juegos y aplicaciones interactivas, nos proporciona un entorno familiar y robusto para la creación de contenido 3D. Permitiendo el uso de scripts con la capacidad de aprovechar las funciones de Unity, adaptando la lógica de comportamiento de los elementos de la aplicación según los objetivos del proyecto.

Por otro lado, la integración de Vuforia en Unity ha facilitado la implementación de funciones de detección y seguimiento de objetos del mundo real de manera eficiente, como se observa con los scripts de detección de marcadores y la asignación de efectos de audio en función de la posición de los objetos detectados. Esta combinación de herramientas ha permitido enfocarse en el diseño de la experiencia de usuario y la implementación de características distintivas, optimizando el tiempo dedicado a tareas de desarrollo específicas y permitiendo una mayor concentración en el diseño y la innovación dentro del proyecto de Realidad Aumentada.[14]

Figura 17: Logos de Unity y Vuforia

4.2 Instalación de Unity

Antes de comenzar a crear la escena de Unity primero debemos descargarnos las aplicaciones y solicitar las licencias. Ambos son gratuitos, aunque la de Unity con la excepción de que nuestra aplicación no se va a comercializar, en caso contrario tendríamos que pagar la licencia.

La instalación la haremos desde la página de Unity y nos descargaremos Unity Hub, el cual funciona como un launcher. Solo tendremos que seleccionar la versión que deseamos instalar, la cual por temas de compatibilidad será la del 2022, e instalar los módulos correspondientes.

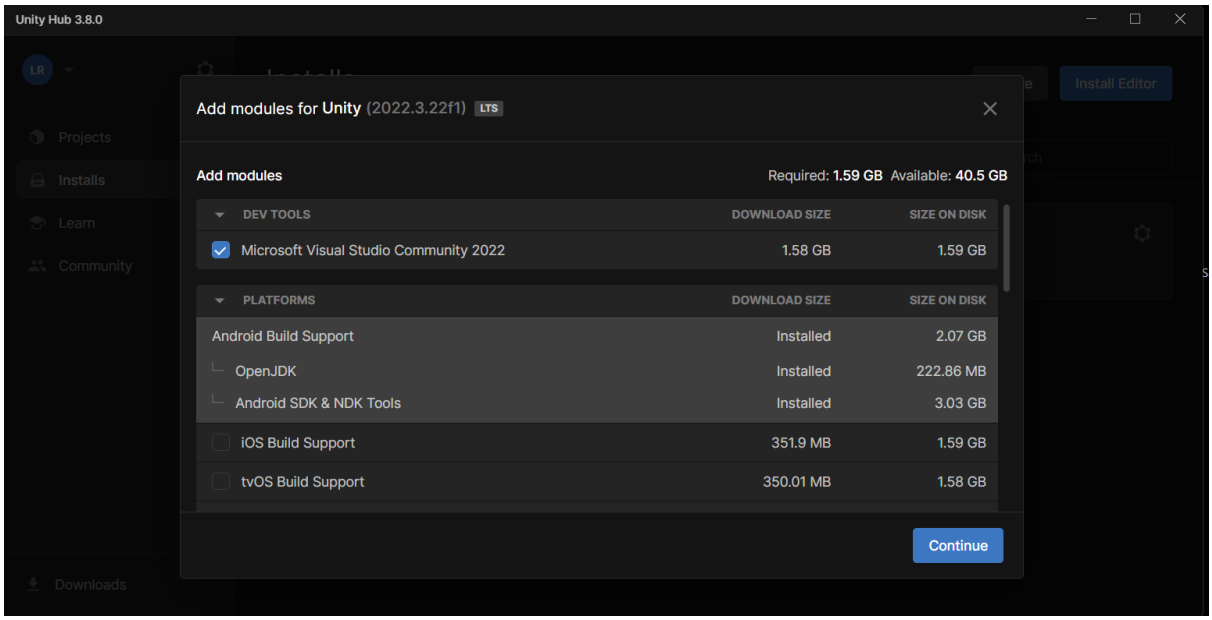


Figura 18: Unity Hub y módulos

Necesitaremos los módulos de “OpenJDK” y “Android”, debido a que la aplicación se desarrollará en el ambiente de Android. Ya que iOS nos limitaría mucho el proyecto por el problema de las compatibilidades. [16]

Una vez solucionado el primer problema, nos falta la licencia de Vuforia. Desde su página tendremos que registrarnos y solicitar la licencia para nuestra aplicación.

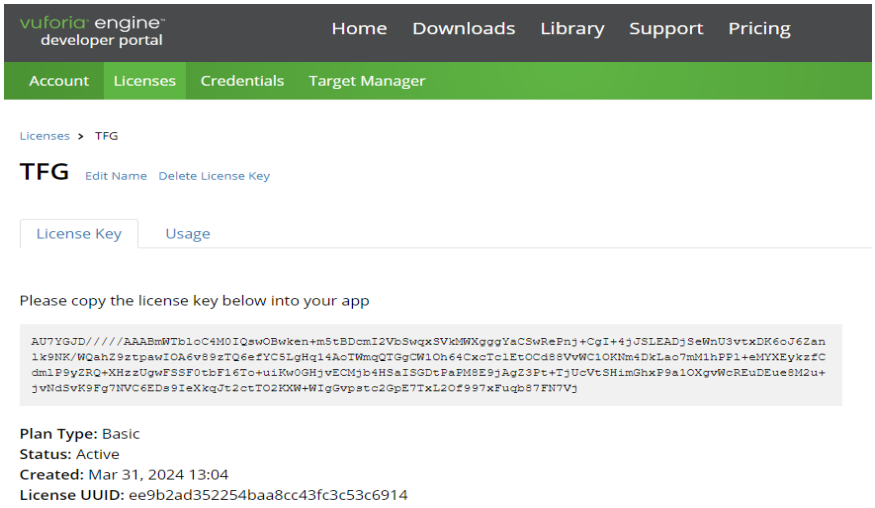


Figura 19: Licencia de Vuforia

Este código nos será fundamental para ingresar nuestra licencia “TFG” en Unity. Ahora, iniciando la aplicación de Unity Hub anteriormente descrita, deberemos de abrir la versión del 2022. [17]

Nos encontraremos un escenario vacío, solo tendremos que hacer clic derecho sobre el apartado de la izquierda de “Sample Scene” y añadir la ARCamera. Este elemento, proporcionado por Vuforia, nos será fundamental para que nuestra aplicación tenga las funcionalidades de Realidad Aumentada. [18]

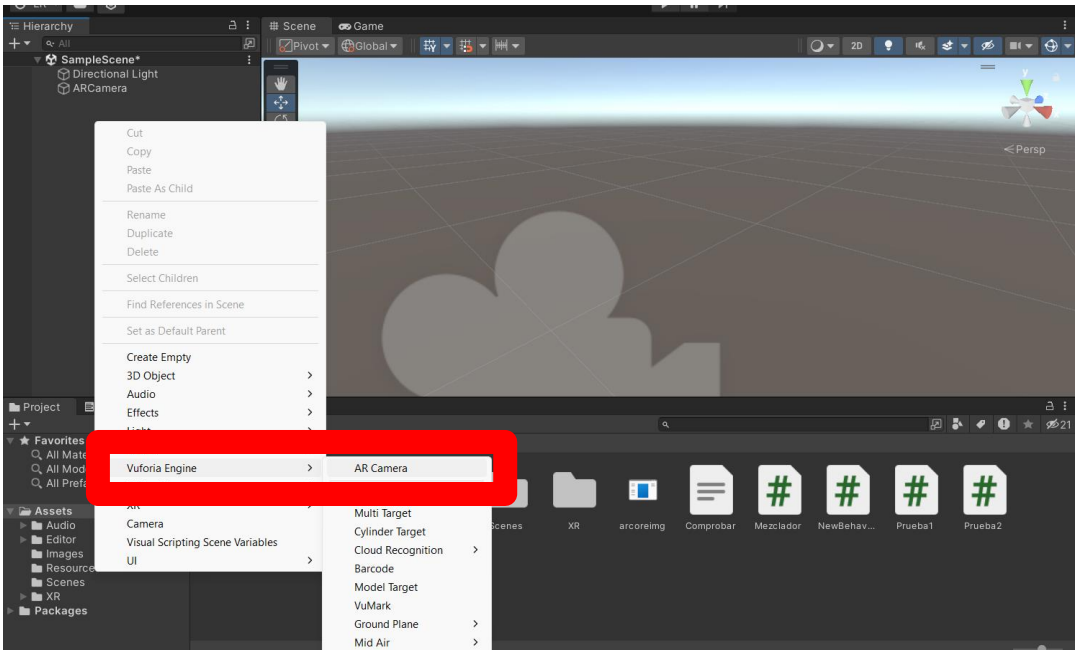


Figura 20: Incorporación de ARCamera

Introduciremos en el elemento de ARCamera, en la pestaña de “Licencia” el código proporcionado anteriormente, terminando así, el proceso de instalación de los programas que usaremos para el desarrollo de nuestro proyecto.

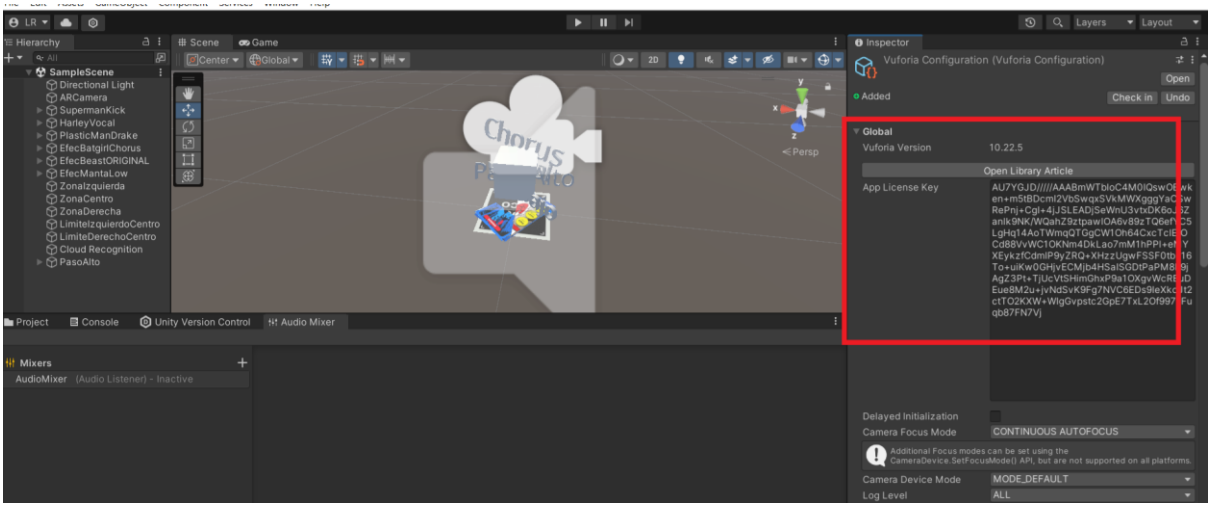


Figura 21: Licencia de Vuforia en ARCamera

4.3 Desarrollo de la aplicación

En este subcapítulo, se tratará el proceso completo del desarrollo de la aplicación. Se seguirá un proceso que se explica a continuación

1. **Creación de la escena:** En este apartado, se describe el proceso de diseño de la escena en Unity, que es el entorno donde interactuarán los elementos de la aplicación de realidad aumentada (RA).
    a. Definición de la escena: La escena es el espacio virtual donde se colocan todos los GameObjects (objetos de juego) que componen la aplicación. En este caso, se utilizará una escena de muestra (Sample Scene) en Unity, que servirá como base para construir el entorno interactivo.

    b. Añadir ARCamera: Se incorpora un elemento crucial, la ARCamera, proporcionada por Vuforia. Esta cámara es responsable de detectar las tarjetas en el mundo real y proyectar los elementos virtuales correspondientes en la pantalla del dispositivo.

    c. Configuración de los GameObjects:

        i. GameObjects: Se definen como los componentes básicos de la escena. Cada uno puede representar un sonido, un efecto o una tarjeta.
        ii. Image Targets: Se utilizan imágenes específicas como targets que la cámara reconocerá. Al ser detectadas, ejecutarán funciones programadas, como activar sonidos o efectos.

2. **Generación de efectos:** Este apartado detalla como se implementan los efectos de audio en la aplicación:
    a. Uso de Audio Mixer: Se utiliza el Audio Mixer de Unity para gestionar y modificar el audio. Este componente permite crear mezclas complejas y aplicar efectos de sonido en tiempo real.

    b. Configuración de efectos:

        i. Subgrupos de efectos: Se crean subgrupos dentro del Audio Mixer para cada efecto deseado (como Chorus, Flanger, etc.). Cada subgrupo puede tener sus propios parámetros ajustables.

        ii. Asignación dinámica: Los efectos se asignan a los sonidos de manera dinámica. Esto significa que los usuarios pueden aplicar diferentes efectos a los sonidos según su preferencia, y estos se activan o desactivan en función de la interacción del usuario con las tarjetas.

    c. Parámetros ajustables: Los efectos no son estáticos; los usuarios pueden modificar parámetros como la frecuencia, la profundidad y otros, lo que permite personalizar la experiencia de mezcla de audio.

1. **Módulos de la aplicación:** Este apartado se refiere a las diferentes funcionalidades y componentes que hacen que la aplicación sea interactiva y dinámica.:
    a. Módulo de cambio de escena: Este módulo permite a los usuarios navegar entre diferentes escenas de la aplicación. Al iniciar la aplicación, los usuarios pueden elegir entre varias opciones (por ejemplo, diferentes conjuntos de sonidos). Se implementa mediante un script que gestiona la carga de diferentes escenas en Unity.

    b. Módulo de panning y detección de sonidos: Este módulo se encarga de la distribución del sonido en el campo estéreo. Según la posición de las tarjetas en el espacio físico, el sonido se reproduce en el canal izquierdo, derecho o ambos. Utiliza la función AudioSource.panStereo para ajustar el balance del audio en función de la ubicación de las tarjetas.

    c. Módulo de asignación de efectos: Permite aplicar efectos de audio a las fuentes sonoras detectadas. Cuando una tarjeta de efecto es reconocida, el sonido correspondiente se enruta a través del subgrupo del Audio Mixer que contiene el efecto.

       Cada tarjeta puede modificar un parámetro específico del efecto, permitiendo un control granular sobre cómo suena cada fuente de audio.

Por último, se mostrará un esquema donde se podrá visualizar todo el proceso detallado anteriormente

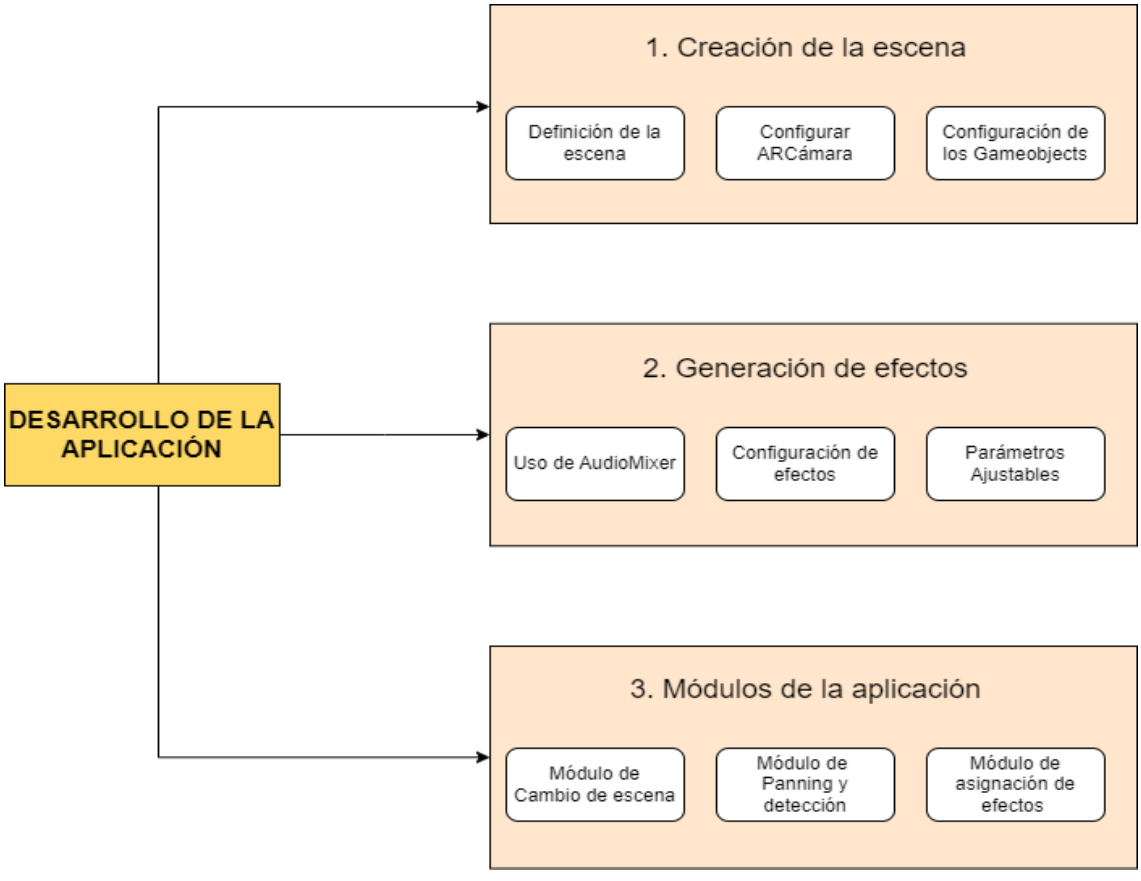


Figura 22: Esquema del desarrollo de la aplicación

4.3.1 Creación de la escena

Antes de empezar a diseñar nuestra mesa de mezclas, definiremos lo que es un GameObject, su uso será clave, siendo el componente más elemental de nuestra programación.

En Unity, un GameObject es el componente base sobre el cual se construyen todos los elementos de un juego. Es un contenedor que puede representar cualquier objeto en la escena, desde un personaje o un enemigo hasta un elemento del entorno como una luz o una cámara. Por sí solo, un GameObject es bastante simple, ya que su funcionalidad proviene de los componentes que se le añaden, como scripts, modelos 3D, sonidos, y otros elementos. [19]

En nuestro proyecto, utilizamos GameObjects para representar y manipular los diferentes elementos que componen la mesa de mezclas en realidad aumentada. Cada GameObject puede estar asociado con una tarjeta de realidad aumentada que, al ser reconocida por la cámara del dispositivo, activará una serie de componentes y comportamientos programados, como la generación de un sonido o la aplicación de un efecto.

Estos GameObjects son esenciales para la interacción del usuario con la aplicación, ya que permiten que las tarjetas físicas que el usuario manipula en el mundo real se traduzcan en acciones y efectos dentro del entorno digital tridimensional.

A lo largo del proyecto se irán completando estos GameObjects. Iremos añadiendo componentes mediante el botón de “Add Component” y creando sinergias entre los demás elementos de Unity.

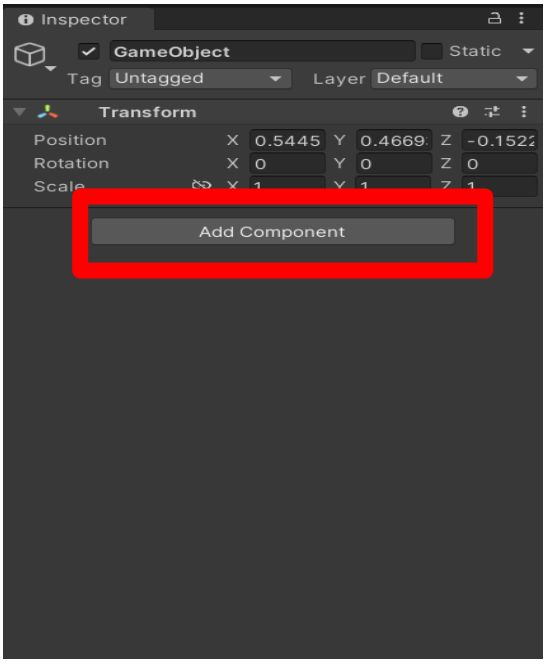


Figura 23: GameObject vacío

Para empezar con nuestra mesa de mezclas, usaremos imágenes que serán reconocidas por la cámara de nuestro teléfono. Al ser detectadas ejecutarán el código y funciones que hayamos creado. En esencia, así funcionará la aplicación.

Lo primero de todo será buscar los elementos físicos que usaremos en el mundo real para invocar funciones y sonidos. Estas imágenes, en formato de tarjetas a utilizar, deben de ser objetos diferentes entre sí, con patrones distintos, y mientras más contraste mejor, para así ayudar al programa a detectar mejor los elementos. Funcionarán como si de un QR se tratara. [20]

Las primeras pruebas se realizarán con elementos cotidianos, como en mi caso son los cromos. Tendremos que escanear la imagen de los cromos para tenerlas digitalizadas y poder usarlas.

Figura 24: Cromos de superhéroes

Pasaremos de nuevo a Unity, donde incorporaremos en el apartado de “Sample Scene” un nuevo GameObject, pero esta vez entraremos en las opciones de Vuforia y crearemos un GameObject con algunas funciones ya adaptadas al reconocimiento de imágenes en RA.

Este nuevo objeto tendrá en sus propiedades incorporados ya 3 scripts que podremos manejar desde la interfaz gráfica de Unity. Sus funciones son [21]:

- Image Target Behaviour: Script incorporado por Vuforia donde podremos incorporar la imagen escaneada de nuestro cromo que activará las funcionalidades al ser detectado

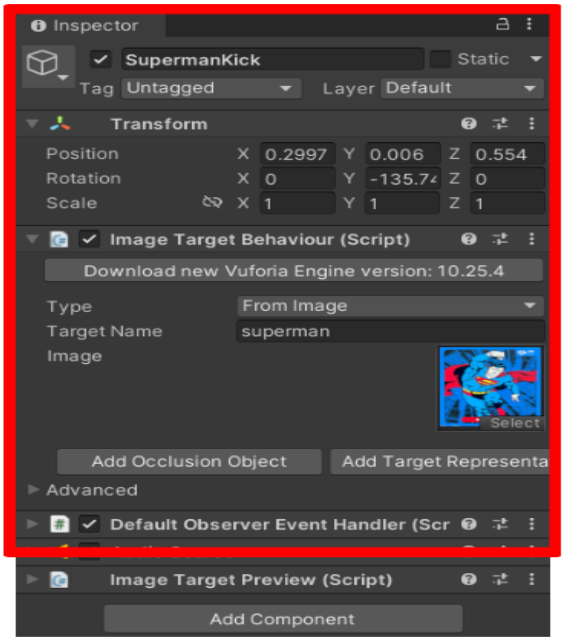


Figura 25: Image Target de GameObject

- Default Observer Handler: Script con el que definiremos las funciones que hará nuestra tarjeta y sus opciones de sobre la detección. Indicando 2 secciones:
    - On target found: En esta sección definiremos los eventos y acciones que ocurrirán cuando la imagen que hemos definido en el script superior, se detecte, en este caso, nuestro cromo
    - On target lost: Acciones y eventos que ocurrirán cuando la tarjeta deje de ser detectada por la cámara.

Desde estos apartados podremos llamar a infinidad de eventos como activaciones de scripts, modificaciones de volumen, interacciones con las tarjetas, etc.

Estas secciones nos servirán como un interruptor tanto para activar y desactivar las funciones de nuestras tarjetas.

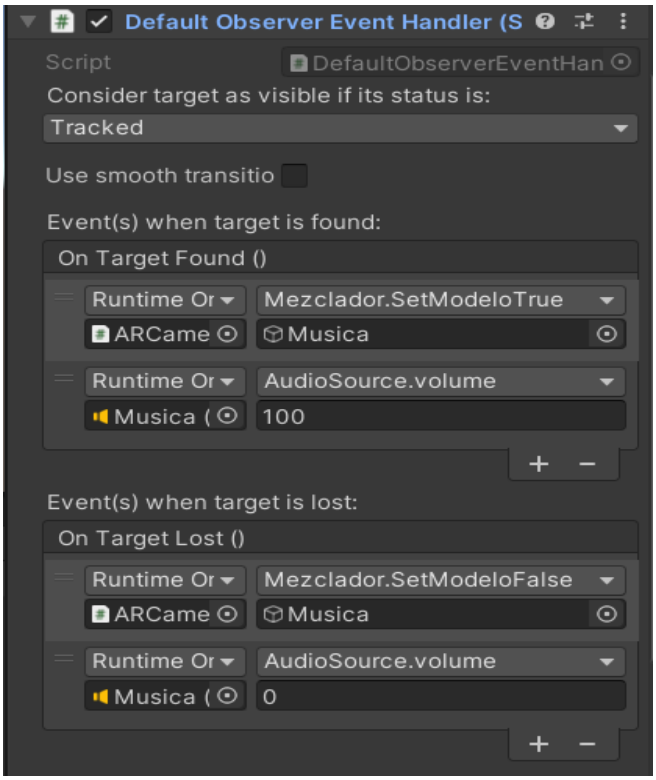


Figura 26: Default Observer Event

· Image Target Preview: Nos servirá para tener una vista previa de la tarjeta en la interfaz gráfica de Unity

Ahora, tendremos que asignarles a estos cromos los sonidos que queremos mezclar. Por lo tanto, usaremos un separador de fuentes, En mi caso, la página web de vocalremover.org para aislar los sonidos de una canción. Esta web permite separar la música en secuencias individuales como la voz, el bajo, la percusión, y te permite reequilibrar sus volúmenes. Esta es la forma más sencilla de obtener multipistas de cualquier canción.

Una vez que elijamos una canción, se separará la música en partes: voz, bajo, batería y otros (“otros” ha sido denominada en el proyecto como “música").

Tras esto, usaremos 4 GameObjects, que serán 4 tarjetas, una para cada uno de los sonidos que hemos separado. [22].

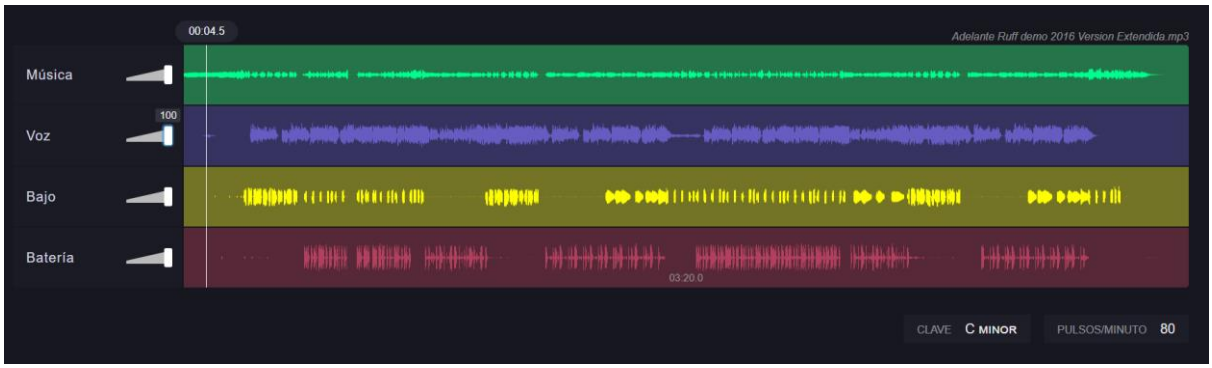


Figura 27: IA separadora de fuentes

Seguidamente, entraremos en la pestaña de nuestro GameObject de Vuforia, e incorporaremos un nuevo apartado, el cual será el Audio Source y procederemos a ajustarlo. [23]

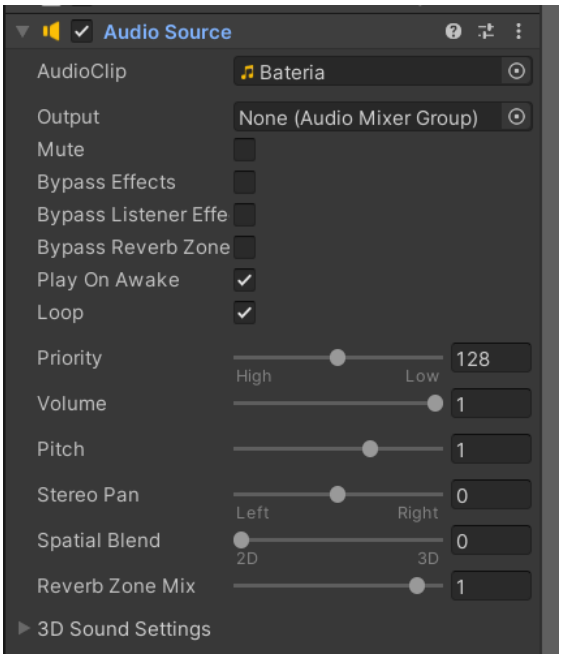


Figura 28:Audio Source en GameObject

Seleccionaremos en la sección de AudioClip el sonido que queremos incorporarle a la tarjeta. Es ahora cuando daremos uso a las 4 pistas de la canción que hayamos seleccionado anteriormente.

La sección de Output la dejaremos vacía, será clave para el procesamiento de efectos, pero será explicado más adelante.

Por último, dejaremos activadas las opciones de Play on awake y la de loop. Esta primera para definir que suene la pista solo cuando se detecte el GameObject, ya que, en caso contrario, empezaría a sonar nada más iniciar la aplicación. Y la segunda opción para establecer un bucle en la pista. Así conseguiremos que cuando los sonidos lleguen al final, se vuelvan a reproducir todos desde el inicio. Evitando que tengamos que cerrar la aplicación para volver a reproducir la música

Aparentemente la sección del AudioSource debería de estar finalizada, pero no es así. Debido a que, desde nuestro GameObject, podemos hacer configuraciones adicionales de una sección, desde otra sección distinta. En este caso lo haremos volviendo la sección de Default Observer Handler. Desde aquí llamaremos a más funciones de AudioSource, que usaremos para que la mezcla siempre siga en sincronismo y vayan a la misma velocidad todos los sonidos.

No haremos que se reproduzcan los sonidos cuando aparezcan en cámara, ya que eso provocaría que cada uno ellos comenzasen solo cuando lo mostrásemos (definido en Unity como AudioSource.Play). En cambio, lo que haremos será, que siempre se estén reproduciendo, pero si no están mostrándose en la cámara, no sonarán (definido en Unity como AudioSource.volume). Lo que provocará que todos los sonidos siempre vayan al mismo compás de la música. Y esto se hará desde dentro del GameObject añadiendo las funciones de volumen con AudioSource.volume, para

establecer que sea “0” cuando la tarjeta no se detecte y estableciendo que el volumen sea “100” cuando la tarjeta se encuentre visible en la cámara. [24]

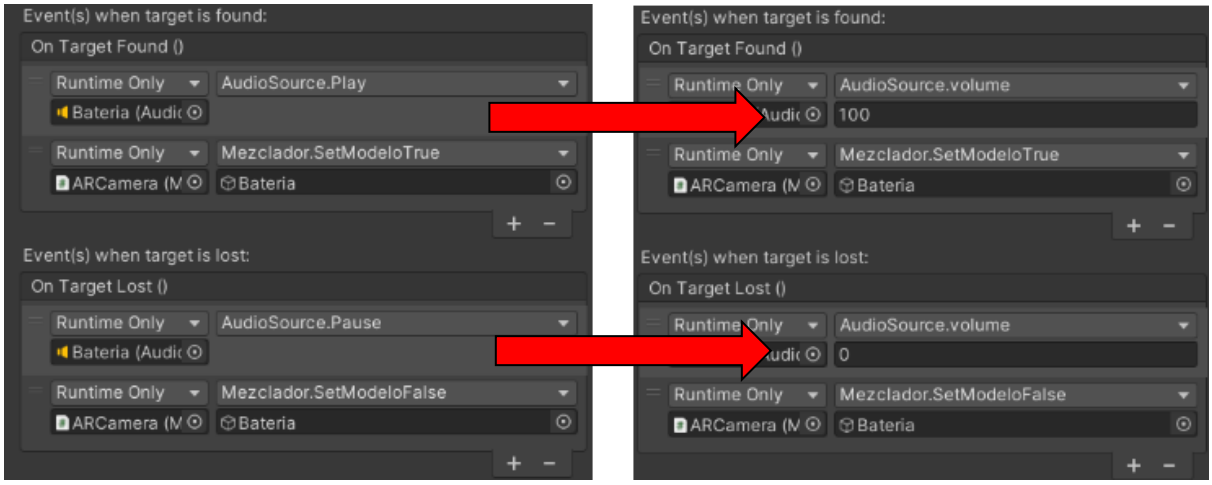


Figura 29: Cambio en el volumen según la detección

Dando así por finalizada la configuración de las fuentes de nuestra aplicación.

4.3.2 Generación de efectos

Nuestra mesa de mezclas virtual necesitará de efectos para poder modificar la música. Esta implementación la haremos haciendo uso del “Audio Mixer”, una herramienta para manejar y modificar el audio dentro del proyecto. Nos da la capacidad de crear mezclas de audio complejas y profesionales, similar a cómo lo haría un ingeniero de sonido en un estudio de grabación. Con el AudioMixer, es posible ajustar volúmenes, aplicar efectos, y crear rutas de señal para múltiples fuentes de audio, todo en tiempo real. Esto se logra a través de un sistema de mezcladores y grupos de mezcladores, que permiten organizar y controlar diferentes elementos de audio de manera eficiente. [25]

Todos los sonidos tienen la posibilidad de redirigirlos por una salida del Audio Mixer. Como se ha mencionado en el parrado superior, funcionaría como un grupo de mezcladores de una mesa real de mezclas. Y cada una de las salidas estará controlada por un master. Por lo tanto, estas salidas del Audio Mixer serán las que nos servirán para asignar los efectos.

El Audio Mixer será un pilar fundamental para nuestra mesa de mezclas. Y este lo podremos cohesionar con todos los demás elementos mediante los GameObjects. Como habíamos visto, podíamos incorporar a los GameObjects desde la sección de Audio Source un apartado de Output. Será desde este dónde asignaremos las salidas de los mezcladores y haremos las agrupaciones.

Teóricamente para implementar efectos en Unity debemos de unir el Audio Mixer, que será el lugar donde definamos los efectos de sonido, con las fuentes de sonido que queremos

Esta configuración se hará desde dentro del GameObject de cada fuente sonora. En un principio el output estará en “none”. Esta decisión se explicará posteriormente.

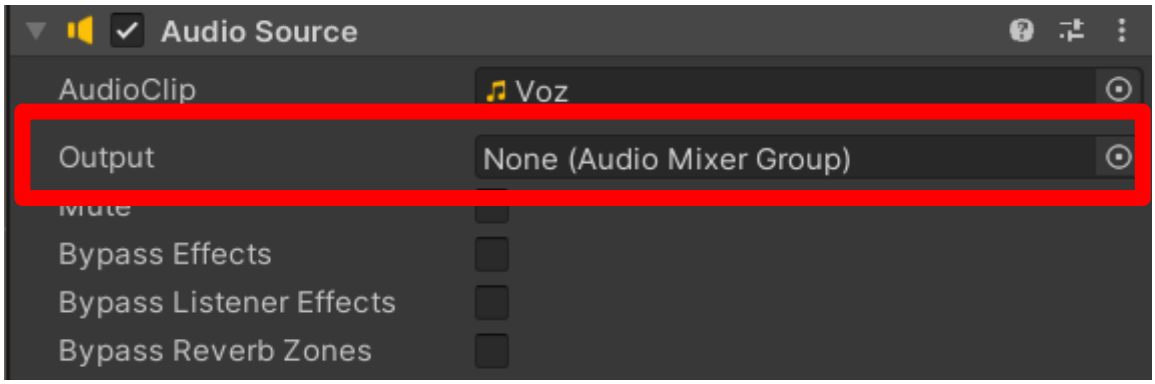


Figura 30: Selección del Audio Mixer desde el Output

El siguiente paso será entrar en la pestaña de Audio Mixer, en la parte inferior de la pantalla. Y ahí crearemos las salidas que irán en como output de nuestras tarjetas de sonido. Y tendremos varios componentes, El master y los subgrupos:

El "master" controla el nivel de volumen general de toda la mezcla de audio, asegurando que el sonido combinado se envíe a los altavoces o dispositivos de grabación al nivel adecuado. Además, permite monitorear los niveles de salida para evitar distorsiones y puede gestionar el envío de la señal a procesadores de efectos externos.

Los subgrupos tendrán las siguientes funciones:

- Control Independiente: Cada subgrupo tiene controles de volumen, efectos (como Chorus o LowPass), y otros parámetros. Nos permitirá manejar de manera independiente distintos aspectos del audio, como los efectos aplicados a ciertos sonidos o el volumen relativo entre diferentes tipos de sonido en el proyecto.
- Efectos Personalizados: Los subgrupos permiten añadir efectos específicos (mostrados la Figura 29) solo a ciertos grupos de sonidos, en lugar de aplicarlos globalmente a todo el audio del juego. Por ejemplo, podríamos aplicar un efecto de reverb solo a las voces en el subgrupo "Chorus", mientras que otros sonidos permanecen sin ese efecto.
- Jerarquía de Mezcla: Los subgrupos permiten crear una jerarquía de mezcla, donde los sonidos se procesan en pasos. Los sonidos podrían pasar por ejemplo, primero por el subgrupo "Original" para un ajuste inicial, luego por "Chorus" para un efecto adicional, y finalmente por "Master" para el control final de salida.

Figura 31: Selección del Audio Mixer desde el Output

Entendido esto, lo que haremos será crear un subgrupo para cada uno de los efectos que deseemos. Los subgrupos son elementos vacíos que redireccionan el audio. Si no hacemos ninguna modificación de los subgrupos realmente no tienen ninguna utilidad.

Unity nos ofrece un listado de 16 efectos. Haremos una selección de los que más nos interesen y crearemos un subgrupo. Le modificaremos el nombre y le añadiremos el efecto deseado. Un subgrupo para cada efecto. Ahora para aplicar los efectos, tenemos que conseguir que los sonidos tengan como output el efecto que deseamos aplicarle. Esto se debe hacer de manera dinámica permitiendo la activación y desactivación de los efectos cuando el usuario lo desee, ya que, si lo asignamos manualmente desde Unity, siempre tendrá la fuente sonora el efecto que le hayamos asignado desde el GameObject. Esta asignación dinámica se explicará más adelante. [26]

Con todos estos elementos. Ya tendremos la base de nuestra mesa de mezclas establecida. Ahora necesitaremos relacionar todos los elementos entre sí. Tarea que se hará mediante el Script y que se explicará mediante el diagrama de flujo

4.3.2.1 Generación de efectos.

Si queremos que nuestra aplicación de mesa de mezclas sea útil e innovadora para los usuarios, no basta con ofrecer solo efectos. Para proporcionar una experiencia verdaderamente personalizada, es esencial que los usuarios puedan ajustar y modificar los parámetros de estos efectos de manera directa. Esta capacidad de personalización es clave para que los usuarios puedan experimentar y explorar diferentes posibilidades sonoras según sus preferencias y necesidades creativas.

En nuestra aplicación, vamos a mostrar cómo se han diseñado una serie de efectos adicionales no solo para su aplicación estática, sino también para permitir que los usuarios modifiquen cada uno de los parámetros asociados a estos efectos. Esto incluye ajustes precisos como la profundidad, la tasa de modulación, el retardo, entre otros, ofreciendo así un control granular sobre el sonido final.

A lo largo de este apartado, se detallan los parámetros disponibles para cada efecto, explicando cómo cada uno de ellos puede ser ajustado por el usuario. Esta flexibilidad en la modificación de efectos permite una personalización avanzada, transformando la experiencia de mezcla de audio en algo mucho más interactivo y dinámico.

- **Paso bajo:** Nos permitirá el paso de frecuencias bajas y atenuará o bloqueará las frecuencias altas. Su único parámetro modificable es:
    - **Cutoff frequency:** Será el punto en el que el filtro comienza a reducir la amplitud de las frecuencias que pasan a través de él. Por encima de esta frecuencia, las señales son atenuadas progresivamente más a medida que la frecuencia aumenta.
        - **Valor:** Desde los 10 Hz hasta 22000 Hz

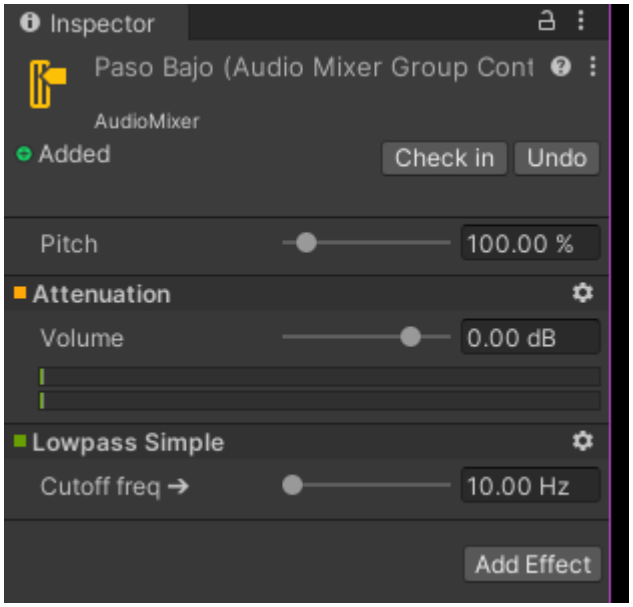


Figura 32: Parámetros del paso bajo

- **Paso alto:** Nos permitirá el paso de frecuencias altas y atenuará o bloqueará las frecuencias bajas. Su único parámetro modificable es:
    - **Cutoff frequency:** Será el punto en el que el filtro comienza a reducir la amplitud de las frecuencias que pasan a través de él. Por debajo de esta frecuencia, las señales son atenuadas progresivamente más a medida que la frecuencia disminuye.
        - **Valor:** 10 Hz hasta 22000 Hz

Figura 33: Parámetros del paso alto

- **Compresor:** Su función principal es reducir la diferencia entre los niveles de

volumen más altos y los niveles más bajos de una señal de audio. Esto lo lograremos mediante la aplicación de ganancia automática: cuando la señal supere un umbral predeterminado, el compresor reducirá el nivel de la señal, disminuyendo así su dinámica. Sus parámetros son:

- **Attack:** El ataque determina cuánto tiempo tarda el compresor en aplicar la compresión una vez que la señal supera el umbral. Un ataque más rápido hará que el compresor comience a reducir el volumen inmediatamente después de que la señal exceda el umbral.
- **Release:** El release establece cuánto tiempo tarda el compresor en dejar de actuar después de que la señal vuelva por debajo del umbral. Un tiempo de liberación más largo significa que el compresor continuará aplicando reducción de volumen incluso después de que la señal haya vuelto a niveles normales.
- **Make up gain:** Ajusta automáticamente el nivel de salida después de aplicar la compresión dinámica. Aumentar el volumen global de la señal comprimida para compensar la reducción de amplitud realizada por el compresor.
- **Threshold:** Este parámetro determina el nivel de volumen a partir del cual el compresor comenzará a actuar. Cuando la señal de audio supera este umbral, el compresor comenzará a reducir el volumen de la señal. Solo nos interesará modificar este parámetro en Unity
  - **Valor:** -60 dB hasta 0 dB

Figura 34: Parámetros del compresor

- **Chorus:** Simularemos la percepción auditiva de múltiples sonidos simultáneamente pero ligeramente desfasados en tiempo y tono. Esto se logra duplicando la señal original, aplicando variaciones sutiles en el tono y tiempo de reproducción de cada copia, y luego mezclando estas señales con la original.
  - **Rate:** Controla la velocidad a la que las variaciones de tono se aplican a la señal de audio, emulando el efecto de vibrato. Este será el primer parámetro que modificaremos
    - Valor: 0 Hz a 20 Hz
  - **Depth:** Ajusta la amplitud o intensidad de las variaciones en el tono, determinando cuán pronunciado es el efecto de desfase en el sonido. Seleccionaremos este parámetro para modificarlo.
    - Valor: 0 a 1

- **Feedback:** Controla la cantidad de señal de salida que se retroalimenta al efecto, afectando la densidad y persistencia del sonido del Chorus.
- **Dry/Wet Mix:** Regula la proporción entre la señal de audio original (Dry) y la señal procesada por el efecto (Wet), permitiendo ajustar la intensidad del efecto sobre la señal final.
- **Delay:** Controla el tiempo de retardo entre las copias de la señal de audio duplicada. Ajustaremos este parámetro modificando la separación temporal entre las versiones desfasadas del sonido, afectando directamente la percepción del efecto de coro. Este parámetro nos será de interés.
    - **Valor:** 0 ms a 100 ms

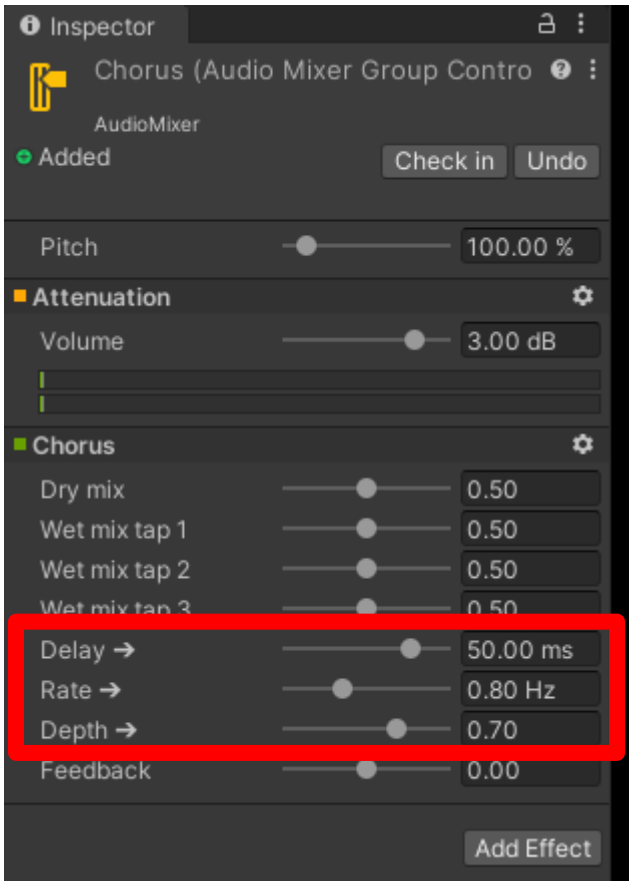


Figura 35: Parámetros del Chorus

- **Pitch Shifter:** Modifica la frecuencia fundamental de una señal de audio, lo que resulta en un cambio perceptible en el tono del sonido reproducido.
    - **Pitch:** Controla la cantidad de cambio en la frecuencia fundamental. Un valor positivo aumenta el tono, convirtiéndolo en agudo, mientras que un valor negativo lo hace más grave. Este será el único parámetro que nos será fundamental modificar.
        - **Valor:** 0.5x a 2x

- **FFT size:** Define el tamaño de la transformada rápida de Fourier (FFT), que afecta la resolución y la calidad del efecto. Valores más altos pueden mejorar la precisión del pitch shifting, pero también incrementan el uso de recursos.
- **Overlap:** Determina la cantidad de superposición entre los segmentos de audio procesados durante el pitch shifting. Incrementar este valor puede mejorar la suavidad del efecto, reduciendo artefactos audibles.
- **Max Channels:** Establece el número máximo de canales de audio que pueden procesarse simultáneamente con el efecto de pitch shifting. Esto afecta la cantidad de instancias del efecto que se pueden aplicar en tiempo real y puede ser ajustado según las necesidades específicas del proyecto.

Figura 36: Parámetros del Pitch Shifter

- **ParamEQ:** Nos permitirá ajustar de manera precisa la amplitud de frecuencias específicas dentro de una señal de audio. Amplificaremos o atenuaremos bandas específicas de frecuencias, consiguiendo un control detallado sobre el perfil sonoro. Para este efecto nos será fundamental los 3 parámetros
  - **Center Freq**: Este parámetro define la frecuencia central alrededor de la cual se aplicará el ajuste del ecualizador. Es la frecuencia que será afectada directamente por los otros ajustes del ParamEQ.
    - **Valor:** 20 Hz a 22000 Hz
  - **Octave Range**: Controla el ancho de banda alrededor de la frecuencia central que será afectada. Un rango de octava más amplio afectará una mayor porción del espectro de frecuencias alrededor de la frecuencia central, mientras que un rango más estrecho afectará una banda más específica y limitada de frecuencias.
    - **Valor:** 0.2 a 5 octavas
  - **Frequency Gain**: Ajusta la cantidad de amplificación o atenuación aplicada a la frecuencia central y su rango de octava. Valores positivos incrementan la amplitud de las frecuencias seleccionadas (haciendo que suenen más fuertes), mientras que valores negativos reducen su amplitud (haciendo que suenen más suaves).
    - **Valor**: 0 a 3

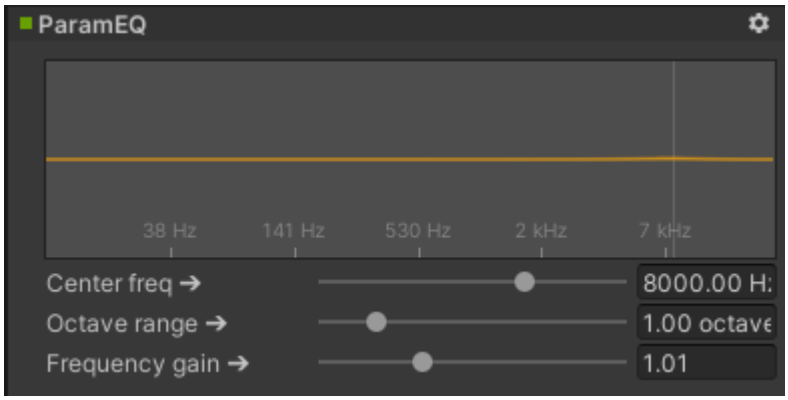


Figura 37: Parámetros del ParamEQ

- **Distorsión:** Altera la forma de onda de una señal de audio al introducir armónicos adicionales, creando un sonido más agresivo y saturado. En este efecto modificaremos su único parámetro
    - **Level:** Este parámetro controla la intensidad general de la distorsión aplicada al audio. A medida que se aumenta el nivel, se incrementa la cantidad de distorsión, lo que nos dará como resultado un sonido más saturado y agresivo.
        - **Valor: 0 a 1**

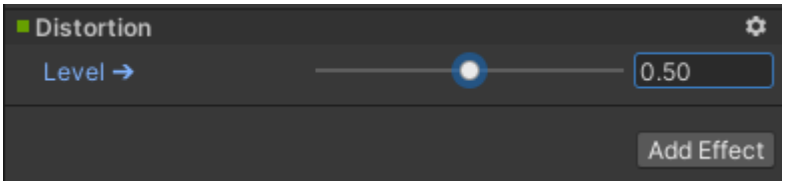


Figura 38: Parámetros de la distorsión

- **Flanging:** Modifica una señal de audio al combinarla con una versión ligeramente retardada de sí misma. Creando así un efecto de barrido al modificar dinámicamente las frecuencias de la señal. Modificaremos los parámetros de depth y rate, ya que serán los más notorios a la hora de tocarlos.
    - **Dry/Wet Mix:** Este parámetro controla la proporción entre la señal de audio dry y la señal procesada por el efecto wet.
    - **Depth**: Define la amplitud máxima de la modulación retardada. Aumentar este valor resulta en una modulación más pronunciada, lo que puede causar efectos más notorios de "subida y bajada" en el sonido.
        - **Valor:** 0.01 a 1
    - **Rate:** Controla la velocidad de la modulación, es decir, cuántas veces por segundo la señal retardada se modula en fase con la señal original. Un valor más alto produce una modulación más rápida y audible, mientras que un valor más bajo resulta en una modulación más lenta y suave.
        - **Valor:** 0 Hz a 20 Hz

Figura 39: Parámetros del Flanging

- **Echo**: Aplica una serie de repeticiones de una señal de audio después de un retraso determinado, simulando una reverberación en nuestro entorno. Dando como resultado profundidad y ambiente al sonido. El único parámetro que nos será de interés para modificarlo será el delay.
    - **Delay:** Este parámetro controla el tiempo de espera entre el sonido original y cada repetición de eco.
        - **Valor**: 1ms a 5000ms
    - **Decay:** Determina la rapidez con la que disminuye la intensidad de cada repetición de eco.
    - **Max Channels:** Define el número máximo de repeticiones simultáneas que pueden ser reproducidas.
    - **Dry/Wet Mix**: Ajusta la proporción entre la señal de audio original sin procesar (dry) y la señal procesada con efecto de eco (wet).

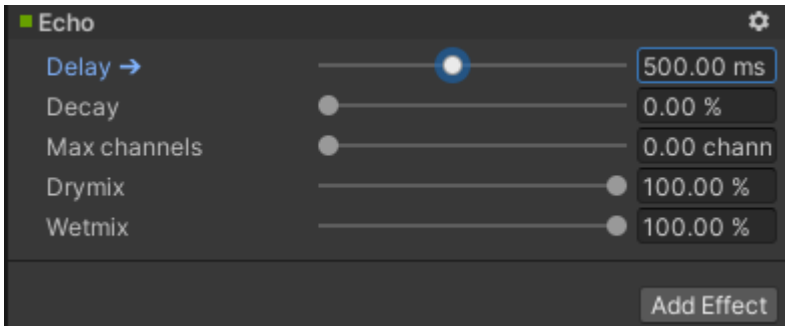


Figura 40: Parámetros del Echo

Procederemos ahora a estudiar todo lo que nuestra app debe de cumplir mediante el script para poder considerarla una mesa de mezclas:

Efectos individualmente y asignación dinámica: La primera opción fundamental, será poder aplicar los efectos al sonido que deseemos. De manera individual y cada uno de ellos se guarde en memoria. Si por ejemplo, está sonando la batería y aplicamos el efecto de Chorus, que nuestra batería cambie su output al Audio Mixer que contenga este efecto.

Variación de los parámetros: Como si de una mesa de mezclas se tratara, queremos tener la posibilidad de modificar los parámetros de nuestros efectos a nuestra voluntad, dándonos la posibilidad de crear mezclas únicas. Si hemos aplicado el efecto

Chorus, que mediante algún tipo de acción, pudiéramos modificar cada uno de los parámetros significativos del efecto

Panning: Para darle más inmersión al usuario distribuiremos el sonido a través del campo estéreo, creando la percepción de que una fuente de sonido proviene de una dirección específica. [27]

Estas 3 funciones serán los pilares de nuestra aplicación. Mantener efectos individualmente garantiza que cada sonido pueda ser manipulado y almacenado con precisión, haciendo que el usuario pueda realizar su propia combinación de efectos y fuentes sonoras y poder tener una composición final, distinta. La variación de los parámetros ofrece un control detallado sobre las características de cada efecto. Finalmente, el panning añade una dimensión espacial al audio, mejorando la inmersión y la percepción del entorno sonoro.

4.3.3 Módulos de la aplicación.

Durante este apartado se mostrará todo el proceso de creación de nuestro software, junto con un diagrama de flujo y el código de programación en Unity para comprender su funcionamiento. Comentaremos ahora paso a paso el funcionamiento de la aplicación y como se ha desarrollado apoyándonos en el diagrama de flujo. En esta parte podremos explicar cómo se han cohesionado los GameObjects, el audio mixer, las escenas y los scripts.

Primeramente, se mostrará la leyenda sobre el diagrama de flujo y los bloques:

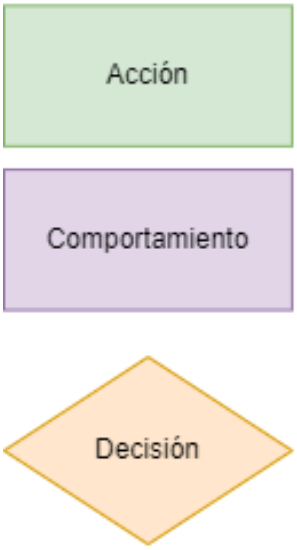


Figura 41: Leyenda sobre el diagrama de flujo

Se muestra a continuación el diagrama de flujo completo:

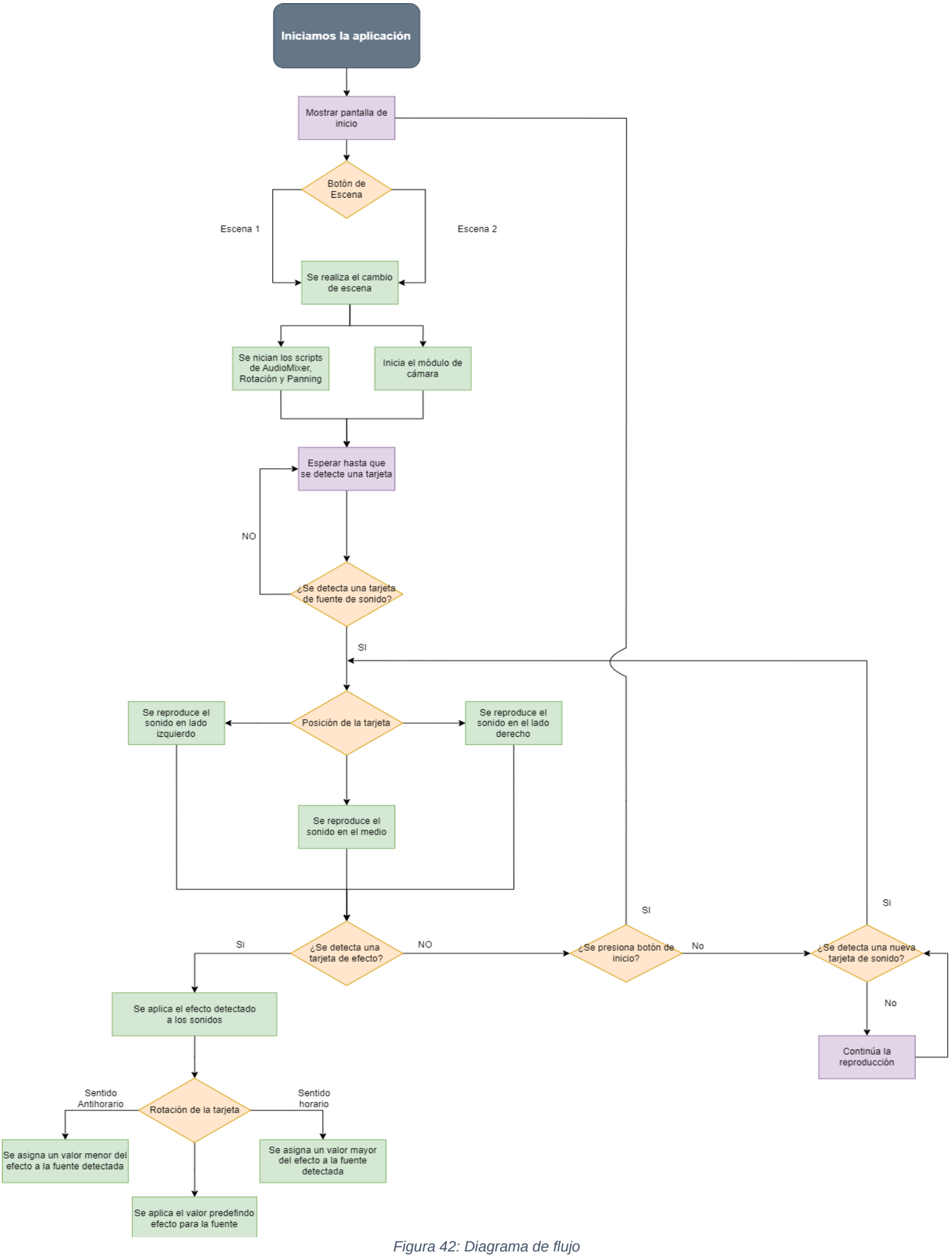


Figura 42: Diagrama de flujo

4.2.5.1 Módulo de cambio de escena.

Al iniciar la aplicación, encontraremos una pantalla de inicio, desde la que tendremos 2 botones. Cada uno de ellos para acceder a una escena distinta. Como habíamos visto, en Unity las escenas son como pantallas. Así que, tendremos para la aplicación 2 escenarios. En cada uno de ellos habrá una canción distinta, para darle mayor posibilidad de creación al usuario, y existirá un botón dentro de las escenas que servirá para volver al menú principal. Queda explicado en el siguiente diagrama [28]:

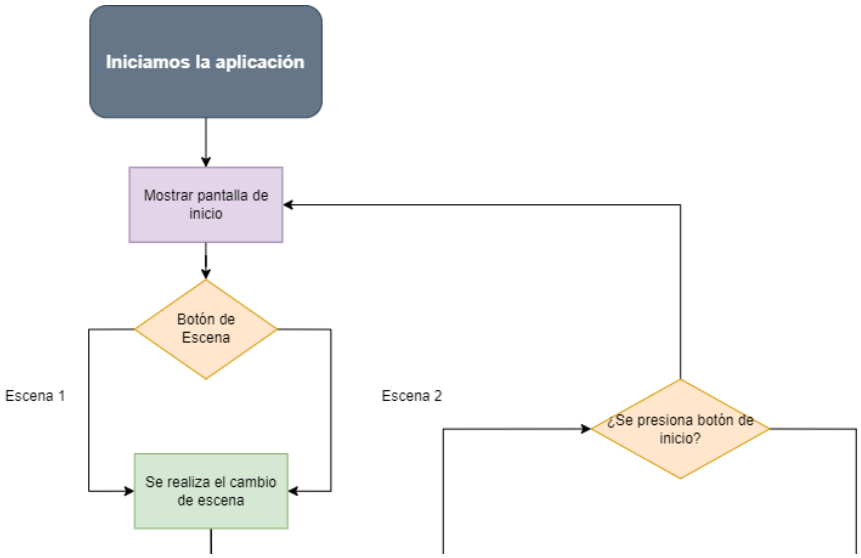


Figura 43: Zoom cambio de escenas, Diagrama de flujo.

Este proceso se ha incorporado en el script:

CambioEscena.cs

Assets > Scripts > CambioEscena.cs > CambioDeEscena

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
```

Figura 44: Parte 1 Script Cambio de escena

Este Script, CambioDeEscena.cs, es muy simple y su función principal es cambiar la escena actual del juego a otra cuando se llama al método GoToStartScene.

Las primeras 4 líneas son para incorporar funcionalidades de Unity, como librerías, pero la que más nos interesarán para este script son:

- UnityEngine: Importa las funcionalidades básicas de Unity, necesarias para cualquier script que maneje objetos.
- UnityEngine.SceneManagement: Importa el SceneManager, que es la clase de Unity utilizada para gestionar las escenas, permitiendo cargar, descargar y manipular escenas.

```
0 references
public class CambioDeEscena : MonoBehaviour
{
    0 references
    public void GoToStartScene(string NombreEscena)
    {
        SceneManager.LoadScene(NombreEscena);
    }
}
```

- Clase CambioDeEscena:

public class CambioDeEscena: MonoBehaviour: Esta línea define una clase pública llamada CambioDeEscena que hereda de MonoBehaviour. Esto significa que esta clase puede ser utilizada como un componente de un GameObject en Unity.

- Método GoToStartScene:
  - public void GoToStartScene (string NombreEscena): Es un método público que no retorna nada (void) y recibe un parámetro NombreEscena de tipo string. Este parámetro es el nombre de la escena que se quiere cargar.
  - SceneManager.LoadScene(NombreEscena): Este es el núcleo del método. Llama a LoadScene de la clase SceneManager para cargar una nueva escena especificada por el nombre que se pasa como argumento.

El script ya está diseñado. Pero ahora tendremos que realizar las configuraciones pertinentes desde la interfaz gráfica de Unity. Tendremos que crear una nueva escena. Una nueva escena solo para el menú. Y en esta, añadir los elementos de Unity necesarios para crear los botones de “Escena 1” y “Escena 2” que hemos visto en el diagrama de flujo. Nombres que tendremos que insertar en la interfaz gráfica, ya que como hemos visto en el Script, el cambió de escenas se realizará mediante el nombre que le hayamos asignado [29].

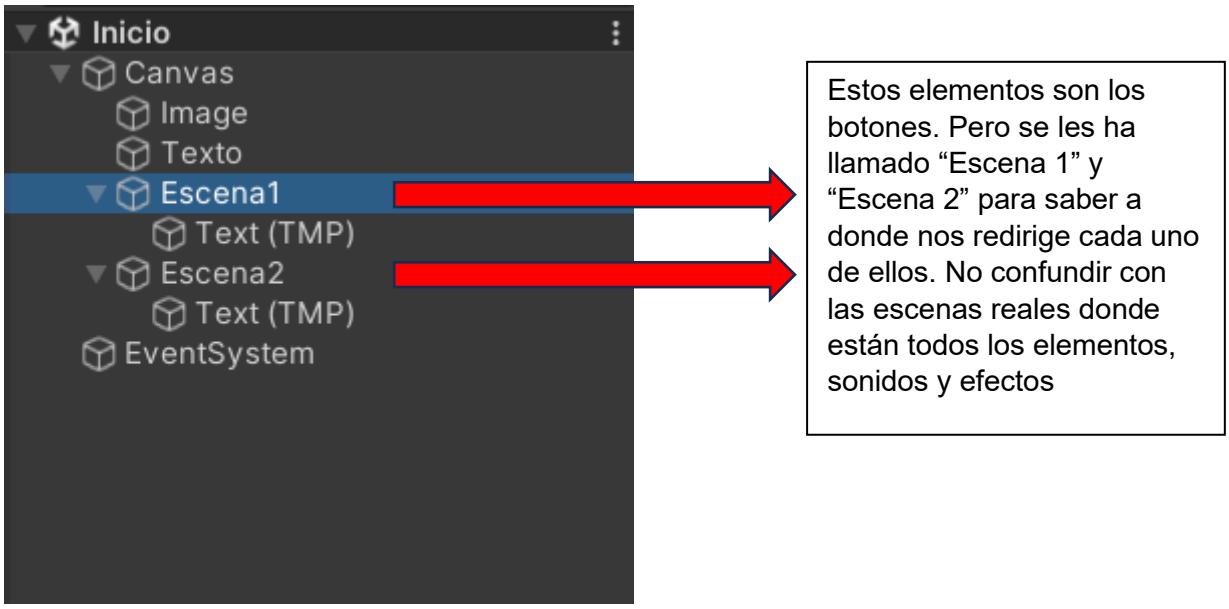


Figura 45: Escena de Ïnicio en Unity

Se creará un Canvas. El Canvas es un componente esencial en Unity para la creación de interfaces de usuario (UI). Es el área en la que se renderizan todos los elementos de la UI, como botones, imágenes, textos, etc. Todo lo que queramos que aparezca como parte de la interfaz del usuario en pantalla debe estar dentro de un Canvas.

Otro elemento imprescindible será EventSystem. Se encarga de manejar la interacción con la UI, como detectar clics en botones, movimientos del mouse, toques en pantalla en dispositivos móviles, etc.

Ahora tendremos que añadir los botones que harán la interacción de acceder a la Escena 1 o a la Escena 2 e incorporarle las funcionalidades. Estos botones funcionan como GameObjects, esto lo que quiere decir es que se le pueden añadir nuevos componentes. Por ello, se le añadirán el Script que hemos desarrollado de cambio de escena, para que puedan realizar la función que deseamos, y pondremos exactamente el mismo nombre de las escenas a las que queremos que nos redirijan. Por supuesto antes de esto, tendremos que haber creado las escenas 1 y 2 en el entorno de Unity.

Figura 46: Creación de todas las escenas de Unity

Dentro de la configuración del botón, arrastraremos el Script. Y en el nuevo campo que se ha creado escribiremos el nombre de la escena a la que queremos que nos redirija. Esto lo tendremos que hacer con los 2 botones existentes.

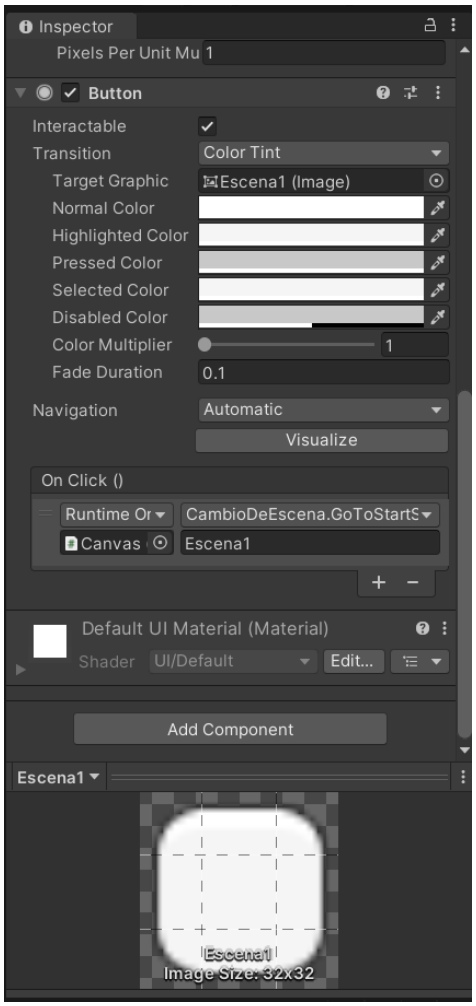


Figura 47: Opciones del botón de Unity

Con todos estos ajustes todo el Script del cambio de escena queda finalizado

4.2.5.2 Módulo de panning y detección de sonidos.

Tras iniciar cualquiera de las escenas, se cargarán todos los scripts y se iniciará la cámara. La aplicación no hará nada hasta que no se detecten primeramente las tarjetas de sonidos, que son los GameObjects que activarán las funcionalidades. Además de eso, dependiendo de su posición dentro del entorno físico, reproducirán el sonido en la izquierda, derecha o centro. Queda reflejado en la siguiente Figura del diagrama de flujo.

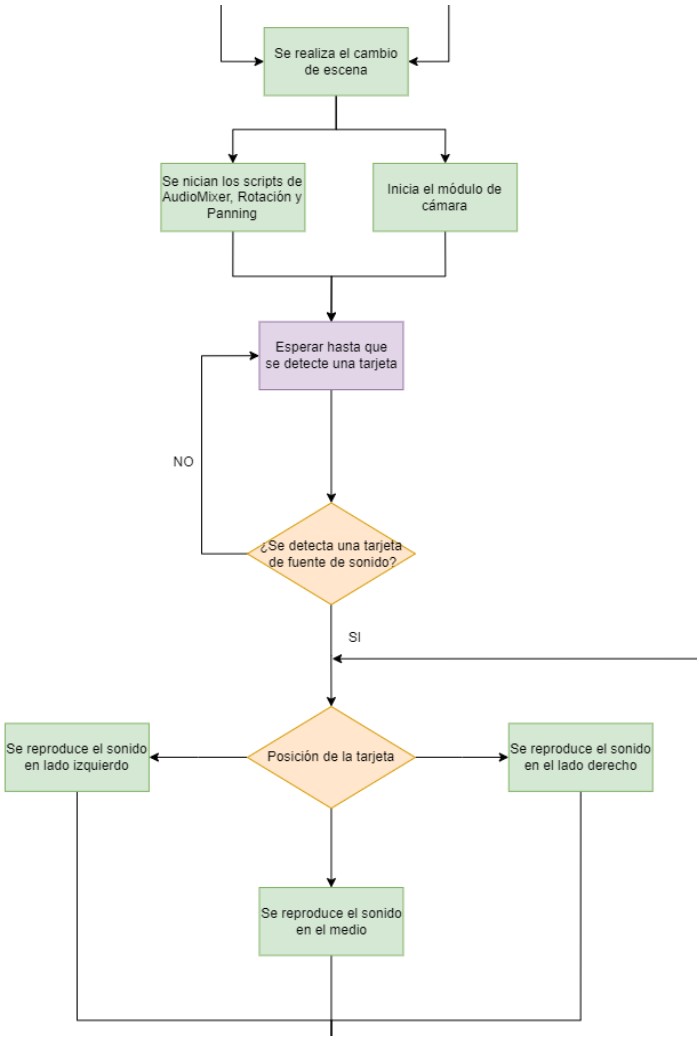


Figura 48: Zoom 2 Panning y detección de sonido Diagrama de flujo

La configuración para la detección del sonido ya la habíamos realizado desde los GameObjects en anteriormente, desde el Script y opciones proporcionadas por Vuforia. Al poner las opciones de On target lost y On target Found.

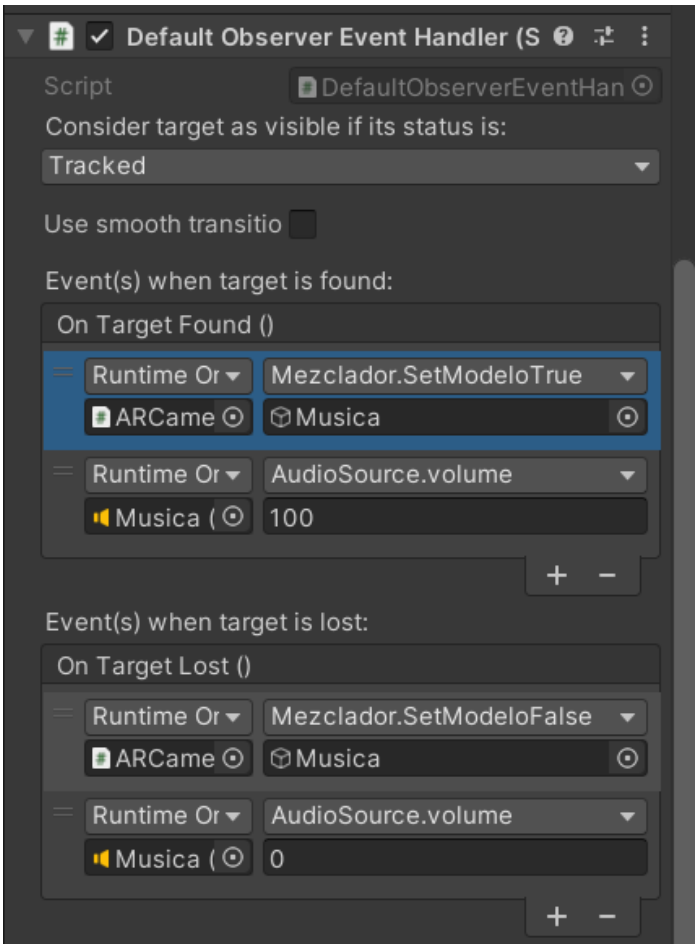


Figura 49: GameObjects de fuentes sonoras

Ahora lo que haremos será que ese sonido que se reproduce al detectar la tarjeta, sea reproducido en el lado físico en el que estén posicionadas. Daremos explicación mediante el script

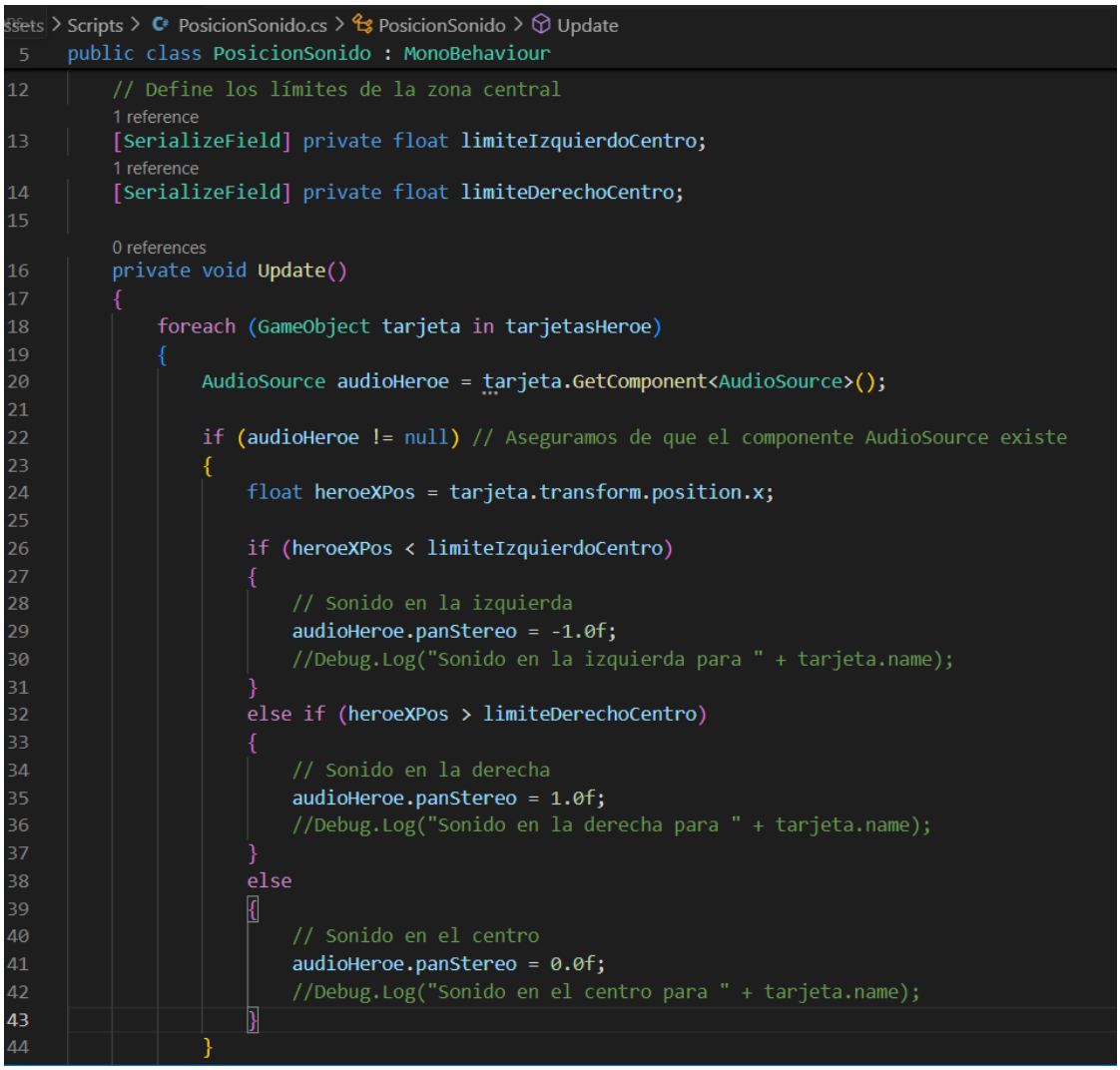


Figura 50: Script Panning

Variables Serializadas: Las variables marcadas con [SerializeField] permiten configurar los objetos y parámetros directamente desde el Inspector de Unity. En este caso, tarjetasHeroe es una lista de objetos que representan las tarjetas del héroe, y zonaIzquierda, zonaCentro, y zonaDerecha son referencias a zonas específicas en la que situaremos GameObjects vacíos que definirán las zonas que activarán el sonido en un lado u otro. Solo nos sirven para marcar la posición. Los límites limiteIzquierdoCentro y limiteDerechoCentro definen los limites de la zona central.

- Configuración del Audio en Función de la Posición:
  - Actualización de Audio: En el método Update, el script revisa continuamente la posición de cada tarjeta de héroe en la escena.
  - Acceso al Componente de Audio: Para cada tarjeta en la lista, el script obtiene el componente AudioSource asociado. Esto permite modificar cómo se escucha el sonido emitido por cada tarjeta.

- Determinación de la Posición y Ajuste del Pan Estéreo:
  - Posición Horizontal: Se obtiene la posición en el eje X de cada tarjeta para determinar en qué parte del área definida se encuentra.
  - Configuración del Balance del Audio: Dependiendo de la posición horizontal de la tarjeta en relación con los límites de la zona central, se

ajusta el balance del audio (pan estéreo) para que el sonido se escuche a la izquierda, a la derecha o centrado. Esto se traduce en un ajuste del panorama estéreo para simular la ubicación del sonido en el espacio 2D o 3D.

- Manejo de Errores
    - Comprobación de Existencia del Componente: Si una tarjeta no tiene un componente AudioSource, el script emite una advertencia en el log. Esto asegura que el script no intente ajustar el audio de un objeto que no tenga el componente necesario.

Ahora, desde la interfaz gráfica de Unity realizaremos la configuración final

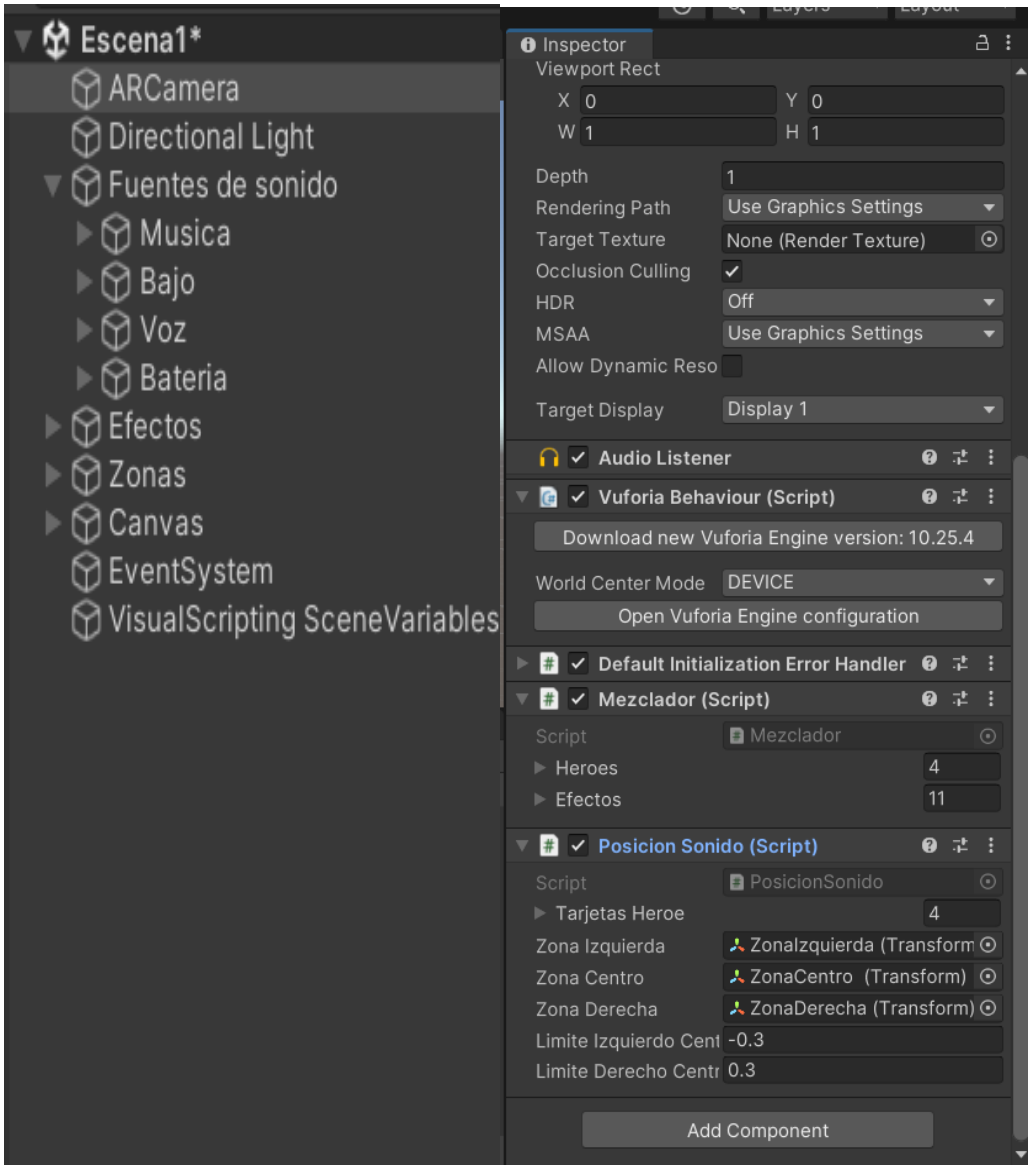


Figura 51: Configuración del ARCamera

Tendremos que arrastrar el Script creado al componente de ARCamera. El ARCamera es un componente específico dentro del sistema de Vuforia que actúa como la cámara principal en una aplicación de realidad aumentada. No es una cámara ordinaria de Unity, sino una cámara adaptada para manejar la realidad aumentada

mediante la integración de varios elementos y configuraciones proporcionadas por Vuforia.

Es muy importante arrastrar aquí el script y no a los GameObjects de las tarjetas sonoras, porque este elemento es el pilar toda la aplicación. Es desde este, donde comienza toda la app. Y por eso escribiremos en los apartados serializados el límite de las zonas físicas, siendo 0,3 y -0,3. Así terminaría la configuración del panning. [30]

4.2.5.3 Módulo asignación de efecto

Tras detectar una fuente sonora o varias, tendremos la posibilidad de modificarla aplicándole un efecto. Este proceso de asignación de efectos se hará enrutando la salida del sonido de las fuentes sonoras por subgrupos del AudioMixer que tienen el efecto ya incorporado.

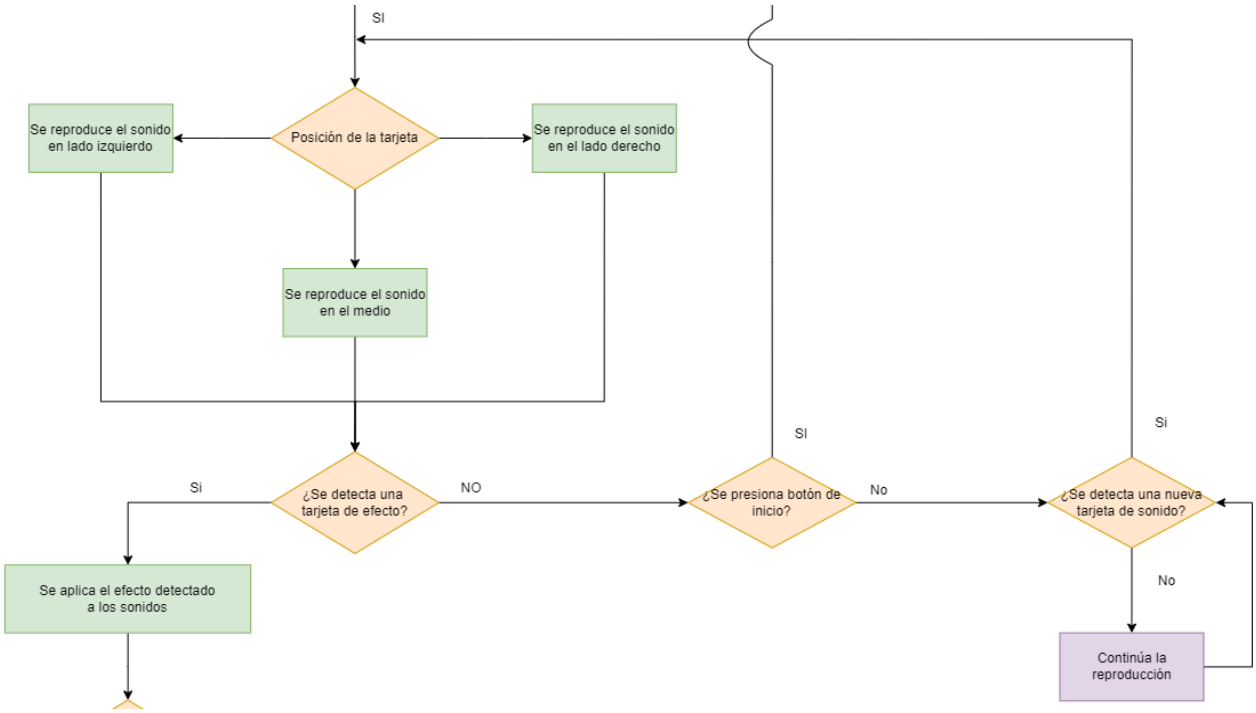


Figura 52: Zoom 3 Asignación de efectos Diagrama de flujo

Configuración Inicial y Estado de los Héroes

- Objetos y Efectos
  - Se definen dos arrays serializados: Heroes y Efectos, donde Heroes contiene los GameObjects correspondientes a los personajes o

elementos que tendrán efectos de audio, y Efectos contiene los distintos grupos de efectos de audio disponibles.

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.Audio;
using Vuforia;

0 references
public class Mezclador : MonoBehaviour
{

    13 references
    [SerializeField] private GameObject[] Heroes;
    33 references
    [SerializeField] private AudioMixerGroup[] Efectos;

    29 references
    private Dictionary<GameObject, bool> EstadoDeLosModelos = new Dictionary<GameObject, bool>();
        0 references
        void Start()
    {
        foreach (var Heroe in Heroes)
        {
            EstadoDeLosModelos[Heroe] = false;
            Debug.Log("Heroe: " + Heroe.name + "Estado: " + EstadoDeLosModelos[Heroe]);
        }
    }
```

Figura 53: Parte 1 Script asignación de efectos

Se inicializa un diccionario (EstadoDeLosModelos) que mantiene el estado de cada héroe (si está activo o no) en relación con los efectos de audio. Este diccionario es poblado en el método Start, donde a cada héroe se le asigna un estado inicial de false. Aplicación de Efectos de Audio

```
    0 references
    public void SetModeloFalse(GameObject Heroe){
        EstadoDeLosModelos[Heroe] = false;
        Debug.Log("Heroe: " + Heroe.name + "Estado: " + EstadoDeLosModelos[Heroe]);
    }

    0 references
    public void SetModeloTrue(GameObject Heroe){
        EstadoDeLosModelos[Heroe] = true;
        Debug.Log("Heroe: " + Heroe.name + "Estado: " + EstadoDeLosModelos[Heroe]);
    }
```

Figura 54: Parte 2 Script asignación de efectos

Hay varios métodos dedicados a aplicar diferentes efectos de audio a los héroes. Cada método sigue una estructura similar: verifica si el héroe está activo en el diccionario de estados y, si es así, le asigna un efecto de audio correspondiente a partir del array Efectos. Los efectos incluyen Chorus, Low Pass, High Pass, Distortion, Flanging, Pitch Shifter, Vibrato, Compressor, ParamEQ, y Echo.

El método VolverOriginal es utilizado para restaurar el efecto de audio original a los héroes que están activos. Esto podría ser útil para reiniciar el estado de los héroes a una condición inicial.

```
public void VolverOriginal(){

    foreach (GameObject Heroe in Heroes)
    {
        if(EstadoDeLosModelos[Heroe]){

            AudioSource AudioHeroe = Heroe.GetComponent<AudioSource>();
            AudioHeroe.outputAudioMixerGroup = Efectos[0];
a\Assets\Scripts\Rotacion\Compresor
            Debug.Log("Efecto aplicado: " + Efectos[0].name + " a " + Heroe.name + " Estado: " + EstadoDeLosModelos[Heroe]);
        }
        else{
            Debug.Log("Efecto " + Efectos[0].name + "NO aplicado a " + Heroe.name);
        }
    }
}

0 references
public void AsignacionEfectoChorus(){

    foreach(GameObject Heroe in Heroes)
    {
        if(EstadoDeLosModelos[Heroe]){

            AudioSource AudioHeroe = Heroe.GetComponent<AudioSource>();
            AudioHeroe.outputAudioMixerGroup = Efectos[1];

            Debug.Log("Efecto aplicado: " + Efectos[1].name + " a " + Heroe.name + " Estado: " + EstadoDeLosModelos[Heroe]);
        }
        else{
            Debug.Log("Efecto " + Efectos[1].name + " NO aplicado a " + Heroe.name);
        }
    }
}
```

Figura 55: Parte 3 Script Asignación de efectos

La asignación de efectos se puede observar que se ha hecho accediendo a la lista de Efectos definida en la primera línea y se le ha ido asignando un número a cada uno de ellos. Como en AsignaciónEfectoChorus se le asigna Efectos[1]. Esta cifra no es aleatoria, es la posición dentro del array que tiene el grupo del AudioMixer. Ya que como se ha mencionado antes la asignación se hace mediante el output del AudioMixer. En el código se ve referenciado mediante la función AudioHeroe.outputAudioMixerGroup.

Ahora, al igual que los anteriores scripts, tendremos que completar su funcionalidad desde la interfaz gráfica de Unity. Y esto lo haremos arrastrando este nuevo script hasta el componente ARCamera de nuevo.

Figura 56: ARCamera Script de Asignación de efectos

Aquí añadiremos a la lista de Heroes. Se le ha nombrado como “heroes” en referencia a las primeras tarjetas utilizadas en el proyecto de superhéroes. Tendremos que arrastrar todos los GameObjects de tarjetas de sonido a la lista. El orden en esta lista es irrelevante.

En cambio, en la lista de Efectos, tendremos que seleccionar cuidadosamente que efecto va a ir en esa posición del array. Ya que como se ha visto en el script, la asignación de los efectos va a estar basada por completo en estas posiciones. Una vez completas estas listas, el componente ARCamera habrá finalizado

Pero aún tendremos que configurar todos los GameObjects de efectos y los de fuentes sonoras.

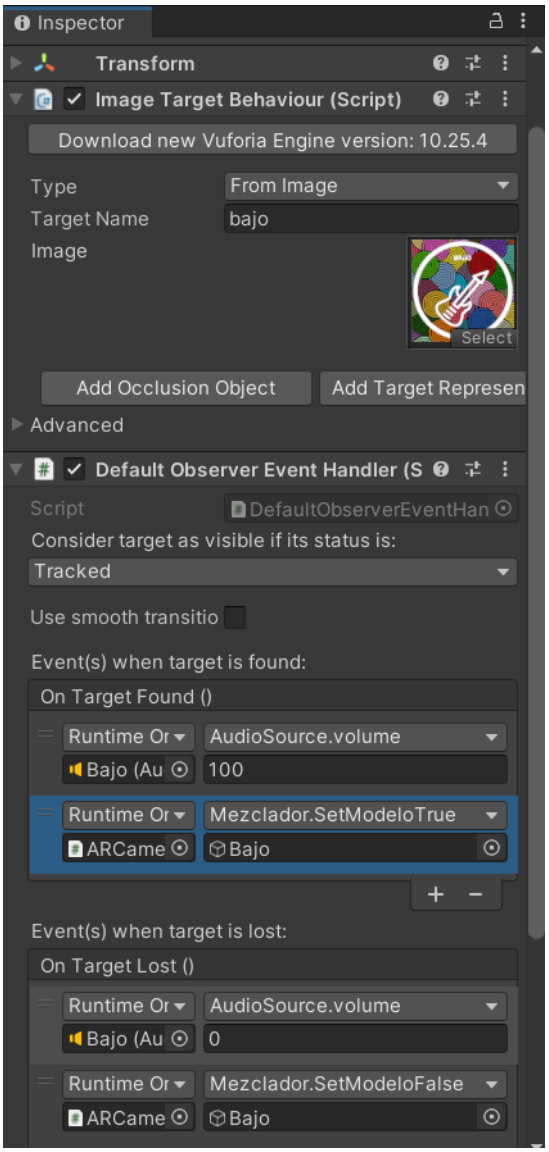


Figura 57: Configuración de las tarjetas de sonido con Script de Asignación de efectos

Para las tarjetas de fuentes sonoras, añadiremos el Script en cada uno de los apartados de eventos. En “On target Found”, añadiremos el script y buscaremos la función True que esta puesta pública justo para eso, para que se pueda acceder desde la interfaz gráfica. Esta configuración será activar en el elemento del objeto detectado en la cámara y ponerlo como True. Es decir, como activo. Para estas funcionalidades ha sido por lo que hemos definido los diccionarios.

En el apartado “On target Lost” haremos justo lo contrario. Añadiremos la función pública “False” del diccionario para desactivar todas las fuentes sonoras que no se encuentren en delante de la cámara.

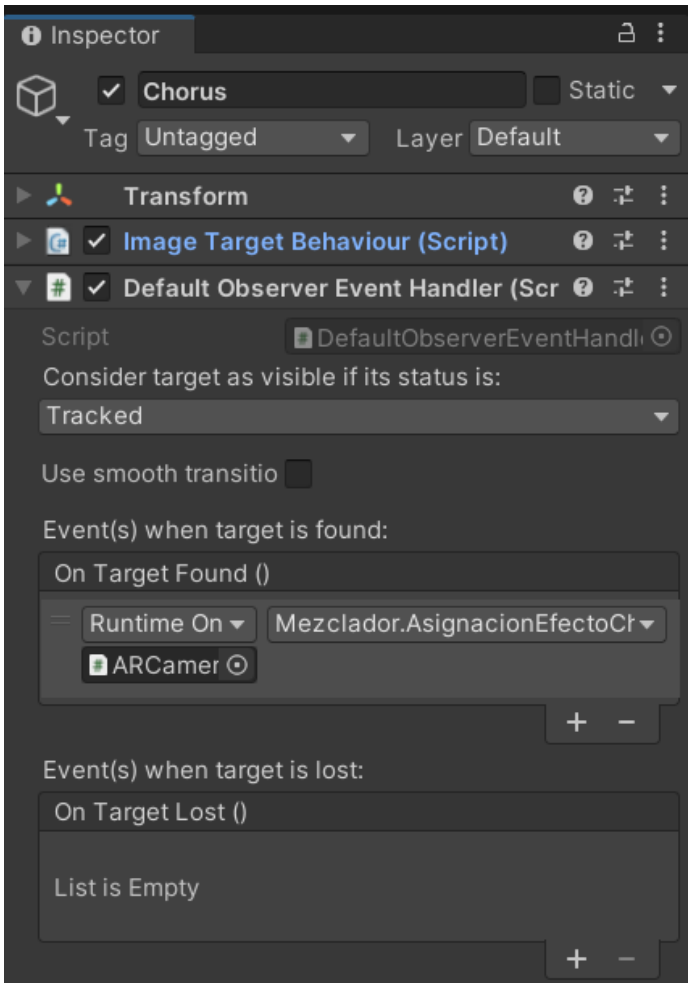


Figura 58: Configuración de las tarjetas de efectos con Script de Asignación de efectos

Con las tarjetas de efectos haremos una configuración similar. Arrastraremos el script de asignación y listo. Daría por finalizado todos los efectos

4.2.5.4 Módulo de rotación de efectos.

Tras detectar un efecto, podremos modificar sus valores mediante la rotación física de las tarjetas

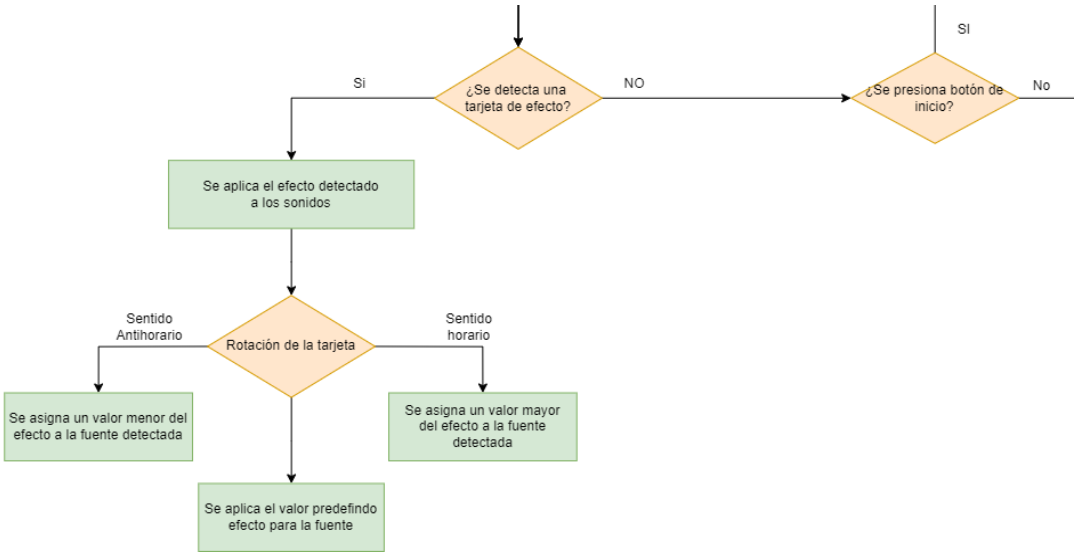


Figura 59: Zoom 4 Rotación de efectos Diagrama de flujo

Iremos relacionando ahora el diagrama de flujo con el código para entender su implementación.

```
4
    0 references
5   public class ChorusDelay : MonoBehaviour
6   {
        0 references
7       [SerializeField] private GameObject[] heroes;
        1 reference
8       [SerializeField] private AudioMixerGroup efectosGroup; // Referencia al grupo de efectos que contiene el efecto de coro
9       // [SerializeField] private string nombreParametroRate = "RateChorus"; // Nombre del parámetro del tono (pitch)
        1 reference
10      private float anguloMaximo = 350f; // Ángulo máximo de rotación
        1 reference
11      private float valorMinimo = 0f; // Valor mínimo del parámetro de tono
        1 reference
12      private float valorMaximo = 100f; // Valor máximo del parámetro de tono
13
        3 references
14      private GameObject targetObject; // Referencia al objeto target detectado por Vuforia
15
        0 references
16      void Start()
17      {
18          // Obtener el objeto target detectado por Vuforia
19          targetObject = GameObject.Find("Chorus");
20      }
21
```

Figura 60: Parte 1 Script Rotación

[SerializeField] private GameObject [] heroes: Declara un array de GameObject llamado heroes, que será visible en el Inspector de Unity, pero privado en el código. En este Array colocaremos las tarjetas de fuentes sonoras.

[SerializeField] private AudioMixerGroup efectosGroup: Declara una referencia a un AudioMixerGroup, que agrupa varios efectos de audio. Será visible en el Inspector. Aquí agruparemos cada uno de los efectos desde la interfaz gráfica.

En el método Start, se busca un objeto en la escena llamado "Chorus" y se almacena en targetObject. Y en el método Update, si el objeto detectado no es nulo, se realiza lo siguiente:

- Se obtiene el ángulo de rotación en el eje Y del objeto.
- Este ángulo se escala entre 0 y 1 basado en un valor máximo predefinido.
- Se utiliza esta escala para interpolar (escalar) un valor de tono entre los valores mínimo y máximo.

Finalmente, se aplica este valor de tono al parámetro ChorusDelay en el AudioMixerGroup, lo que ajusta dinámicamente el efecto de audio en función de la rotación del objeto.

```
0 references
void Update()
{
    if (targetObject != null)
    {
        // Obtener el ángulo de rotación del objeto target en el eje Y
        float anguloRotacionY = targetObject.transform.rotation.eulerAngles.y;

        // Escalar el ángulo de rotación entre 0 y el ángulo máximo
        float escala = Mathf.Clamp01(anguloRotacionY / anguloMaximo);

        // Calcular el valor del parámetro de tono en función de la escala
        float valorRate = Mathf.Lerp(valorMinimo, valorMaximo, escala);

        // Aplicar el valor del parámetro de tono al grupo de efectos
        efectosGroup.audioMixer.SetFloat("ChorusDelay", valorRate);
    }
}
```

Figura 61: Segunda parte Script rotaciones

El script está diseñado, pero tendremos que finalizar la configuración desde la interfaz gráfica inspeccionando las tarjetas de efecto.

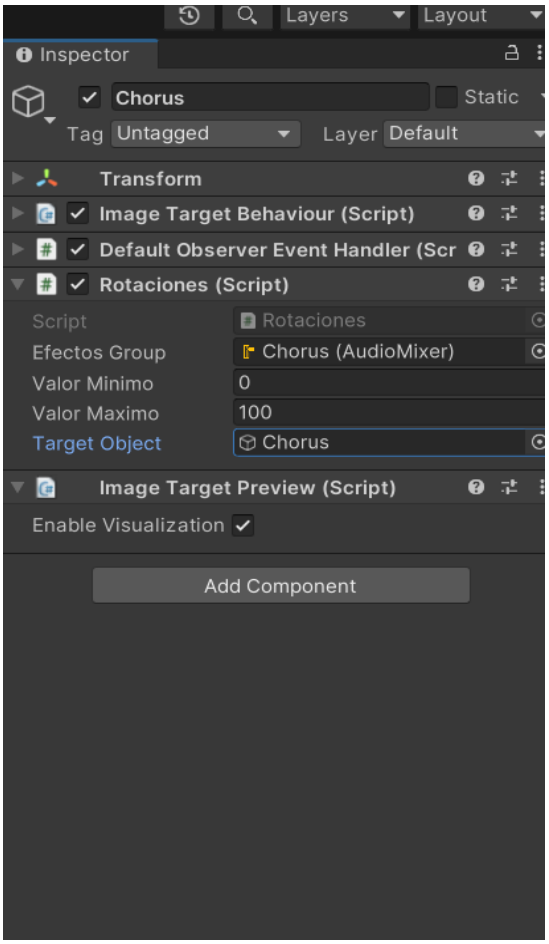
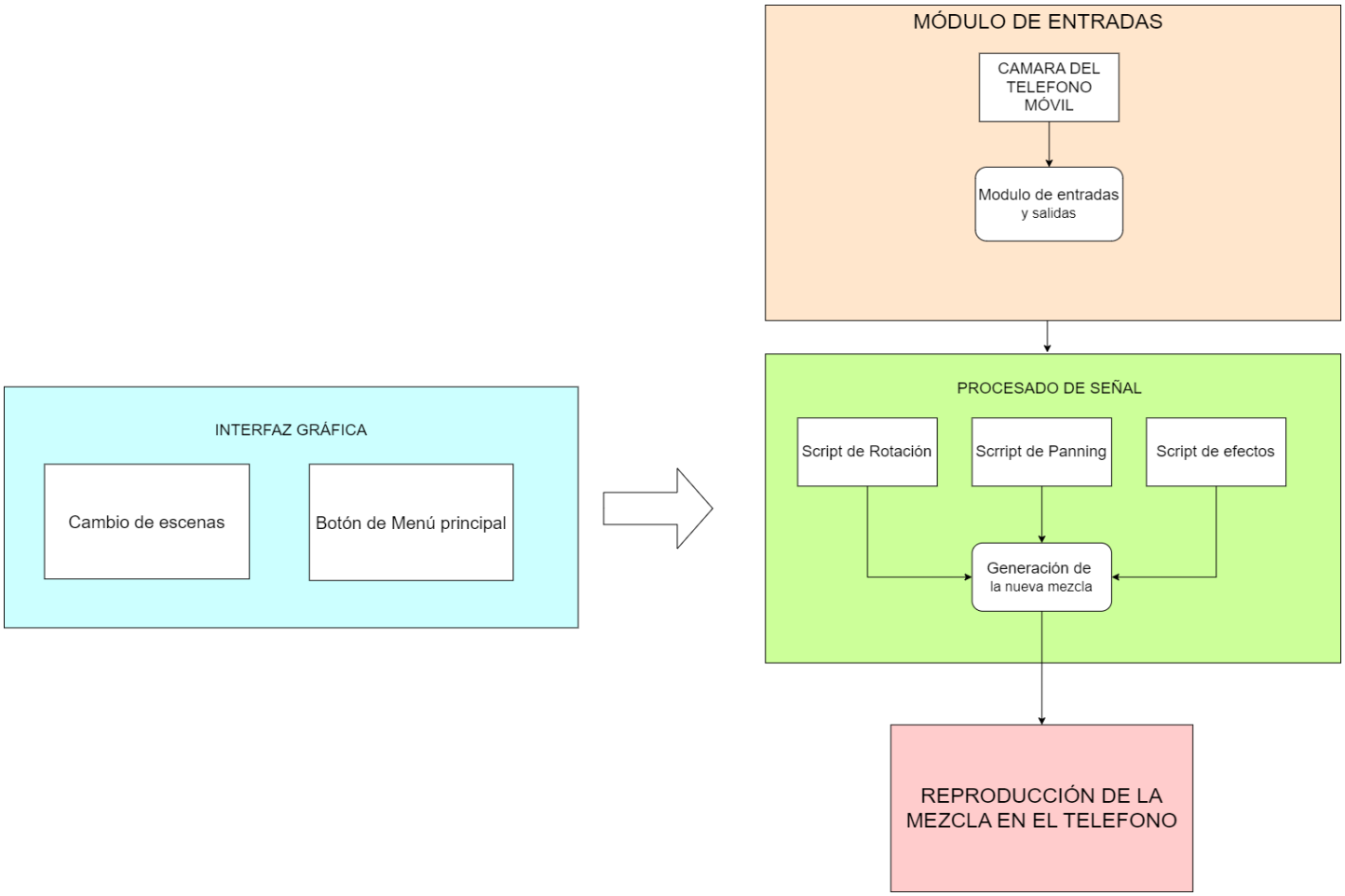


Figura 62: Configuración de efectos con el script de Rotación

Al arrastrar el script a las tarjetas de efecto, tendremos que configurar varios apartados. Tendremos que seleccionar que subgrupo del AudioMixer es el que asignará como output cada tarjeta. Asignar los valores máximos y mínimos y por último arrastrar el objeto.

La función principal del proyecto es crear nuestra mesa de mezclas en AR similar a la mesa de mezclas física. Por ello se puede concluir esta sección comparando la Figura 13 donde se mostraban las funciones y el flujo de la mesa de mezclas con todo nuestro software:



4.4 Pruebas y solución de problemas

Durante el desarrollo de la aplicación, la implementación del script presentó una serie de desafíos que requirieron una cuidadosa atención y resolución. A medida que avanzábamos en la integración de las funciones de mezcla de audio, surgieron problemas relacionados con la precisión de los efectos, reconocimiento de ellos mismos y optimización. Enfrentar y solucionar estos problemas fue fundamental para asegurar que la aplicación funcionase de manera eficiente y proporcionara una experiencia de usuario fluida y satisfactoria. A continuación, se detallan los principales problemas encontrados y las estrategias adoptadas para superarlos:

- **Limitaciones con la asignación dinámica de efectos:** Este problema estuvo presente durante todo el desarrollo. La función de aplicar el efecto es conceptualmente simple. Debemos de asignar a los sonidos el Audio Mixer del efecto que queramos aplicar y que solo se apliquen al sonido que esté en pantalla. Parcialmente se consiguió, pero no aplicaba al sonido que estaba mostrándose, si no a todos a la vez, estuvieran activos o no. Parcialmente era la función que se buscaba, aunque no funcionaba al 100%.
    - La primera solución fue la de establecer una lista en C# con todos los efectos utilizados y todas las fuentes sonoras. Además, crearemos una función pública para cada uno de los efectos. Esta reconocerá que efecto está activo, y enviaremos cada una de las fuentes sonoras que se

encuentran en la cámara por esa salida del Audio Mixer, consiguiendo así aplicarle el efecto deseado modificando su output. Ahora solo tendremos que llamar a la función que acabamos de crear cuando se invoque. Haciendo uso de la interfaz de Unity y de las herramientas de Vuforia. Gracias a que hemos definido anteriormente la función como pública, podemos acceder a ella desde otras clases y definir que cuando esta tarjeta cambie su estado a “detectada”, active la función del efecto correspondiente.

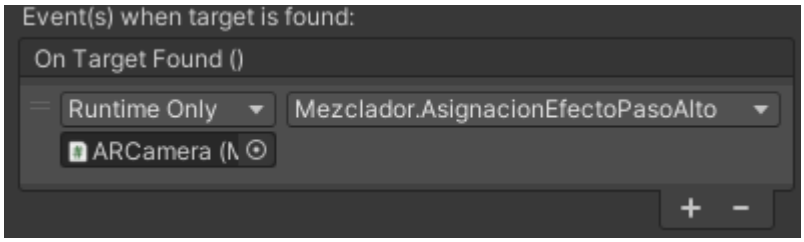


Figura 63: Función pública de asignación de Paso Alto

- Para solucionar el problema de la errónea asignación de efectos a todos los sonidos, haremos uso de un diccionario para las fuentes. Una estructura de datos que nos permitirá almacenar pares de clave-valor, proporcionando un acceso rápido y eficiente. Nos servirá para indicarle a la app cuales son los efectos que se están reconociendo en cámara, actuando como un boolean. La opción del boolean existía previamente desde la interfaz de Unity, pero no actuaba bien. Solo reconocía el objeto la primera vez que aparecía y las demás no lo invocaba, lo deshabilitaba. [31]

  El uso del diccionario lo visualizamos imprimiendo en la consola (debug) y conseguiremos diferenciar y visualizar si las tarjetas se encuentran activas “True” o las hemos sacado de la escena “False”.

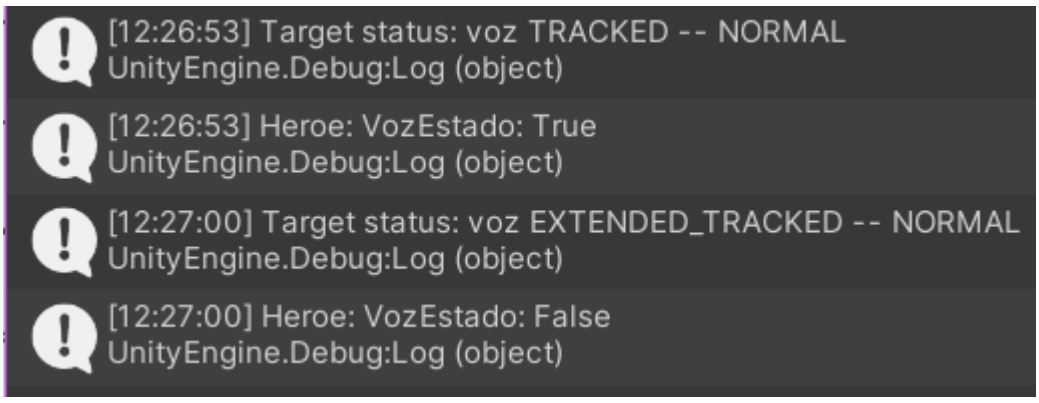


Figura 64: Consola de Unity

- **Ineficiencia de detección:** En varias ocasiones las tarjetas de los superhéroes utilizadas durante la demo daban problemas. Al sacarlas de cámara se seguían ejecutando, se proyectaban sobre el entorno físico sin estar presentes. Esto ha sido muy problemático a la hora de comprobar el correcto funcionamiento de la

aplicación. No nos permitía diferenciar los efectos aplicados ni escucharlos con claridad. Es un enorme problema como usuario que se nos apliquen efectos de manera aleatoria que hemos dejado de mostrar en cámara.

- La solución ha sido radical. Rediseñar por completo las tarjetas de sonido, ya que el código en esta parte no tenía falla y era de fácil configuración. Por lo que se dedujo que la raíz del problema serían las tarjetas. Se optó por diseñarlas mediante Adobe Illustrator, de la manera más profesional posible. Este software de diseño gráfico y dibujo vectorial principalmente usado por diseñadores gráficos y profesionales creativos, nos permitirá crear ilustraciones, logotipos e iconos a nuestra visión. Aprovecharemos sus gráficos vectoriales para mantener las imágenes diseñadas de las tarjetas nítidas y con máxima calidad. Conseguiremos así que la detección sea perfecta. El proceso de diseño se explicará detalladamente más adelante. [32]

Figura 65: Adobe Illustrator

- **Variación de los parámetros mediante giro:** Una de las funciones más complejas a la hora de plasmarlo en el código. El problema residía en el desconocimiento de fusionar la rotación de una tarjeta con un valor.

  - Para empezar, debemos pasar nuestro valor a C, será darle acceso al script a los parámetros. Haciendo clic derecho en la componente que hemos fijado antes como variable, solo tendremos que “mostrarla al Script”. A partir de ahora se marcará con una flecha y podremos acceder a ella desde el código. [33]

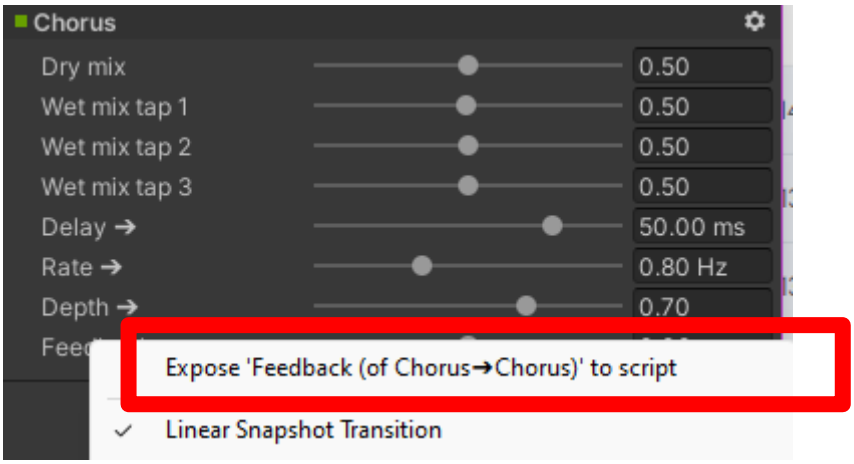


Figura 66: Acceso de parámetros al Script

- Ahora que se tiene acceso a los parámetros, tendremos que idear como variarlos. Nuestras tarjetas estarán sobre una superficie plana, como es una mesa, sabiendo que Unity trabaja con tres componentes: X, Y y Z, la mejor idea será cambiar los parámetros mediante el giro de estas. De esta manera podemos jugar con el eje Y obviando las demás componentes.

  Obtendremos el ángulo de rotación del objeto, lo normalizaremos entre 0 y un nuestro ángulo máximo (350º), y utilizamos este valor para interpolar entre un valor mínimo y máximo del parámetro que deseemos. Finalmente, aplicamos este valor al parámetro de audio expuesto anteriormente al script en un grupo de efectos, permitiendo que el efecto de audio cambie dinámicamente según la orientación.

- **Optimización del Panning en la espacialización de Sonido:** Durante el desarrollo enfrenté un problema significativo relacionado con el panning y los umbrales de audio. A pesar de mover las tarjetas a la izquierda o derecha, no lograba un panning adecuado, la distribución del sonido en el espacio estéreo no funcionaba correctamente. Este fallo nos impedía una correcta espacialización del audio para nuestra mesa de mezclas

  - Para resolver este problema, utilicé la propiedad “AudioSource.panStereo” de Unity, que permite reproducir un sonido en estéreo moviéndolo entre los altavoces izquierdo y derecho. Si la tarjeta estaba a la izquierda el sonido se reproducía completamente a la izquierda (panStereo = -1f). Si estaba a la derecha, el sonido se reproducía completamente a la derecha (panStereo = 1f).

  - Cuando finalmente se consiguió el panning, el sonido no se escuchaba en el centro; sólo se percibía en los extremos izquierdo y derecho. Por lo tanto, cree nuevas zonas límites. Asignando en realidad cuatro fronteras: Límite izquierdo, límite derecho, centro izquierda, centro derecha. Si la tarjeta estaba en el centro, el sonido se balanceaba equitativamente entre ambos altavoces (panStereo = 0f). Si la tarjeta sobrepasaba las zonas intermedias, se escucharía completamente en ese lado. Con este enfoque solucionamos el problema, permitiendo una correcta

espacialización del sonido y mejorando la experiencia auditiva en el entorno.

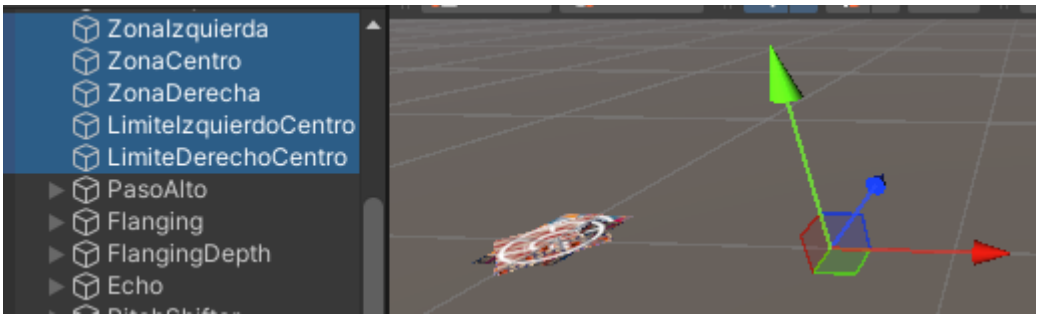


Figura 67: Fronteras del panning

4.5 Proceso de creación de tarjetas

Durante el desarrollo del proyecto ha sido necesario el diseño de dos conjuntos de elementos: Los elementos 3D flotantes sobre las tarjetas y las propias tarjetas

Como se mencionó en el capítulo anterior, el diseño de las tarjetas no ha sido por mera estética. Tiene la función de facilitar a nuestra aplicación la detección de los sonidos y efectos. Diseñarlas nosotros mismos mediante un software profesional, nos da la posibilidad crearlas lo más reconocibles posibles y adecuadas para el efecto que este aplicando

Para el diseño de las tarjetas hemos recurrido al software de Adobe Illustrator. Además de darle un toque más visual, otro objetivo desarrollando estas tarjetas, es que sean intuitivas para cualquier usuario. Desde el principio se tuvo en cuenta que éstas, debían de ser entendibles de un golpe de vista.

En primer lugar, se hicieron pruebas de tamaños, hasta decidir que las dimensiones ideales de estas tarjetas eran de 6 centímetros y de forma cuadrada. Así se consigue una manejabilidad óptima.

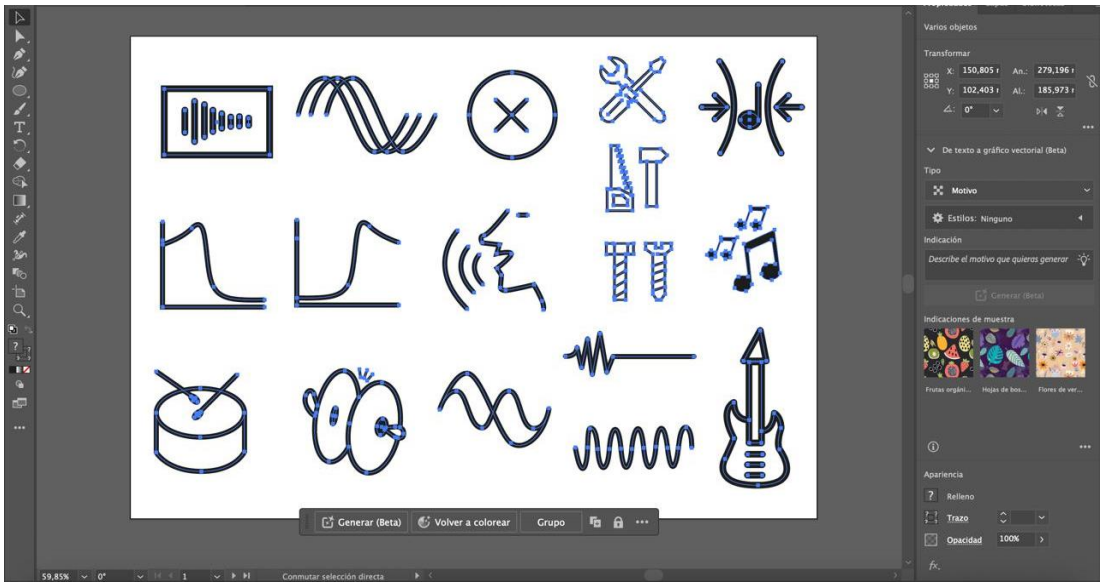


Figura 68: Iconos vectorizados en Illustrator

En segundo lugar, se recopilaron todos los nombres de los sonidos y efectos para asignarle a cada uno de ellos su respectivo icono. Una vez se bocetaron las primeras ideas, se pasó a manejar el software de diseño (Adobe Illustrator) donde se vectorizaron cada uno de los iconos junto con el nombre de cada una de las tarjetas. A continuación, una vez que los iconos están establecidos, se les asigna un patrón o estampado de color como fondo de las tarjetas. Se aplicaron fondos de formas y colores muy variados con el fin de que no hubiera semejanzas entre tarjetas y todas se diferenciasen entre sí. Estos fondos fueron sacados de Pinterest [34]

Figura 69: Fondo de las tarjetas

De esta forma, se evita que Vuforia diese errores de aplicar los efectos cuando era pertinente.

Por último, para hacer las tarjetas tangibles, se mandaron a imprenta para su impresión y se montaron sobre un papel de alto gramaje (cartulina).

Figura 70: Tarjetas impresas

Por otro lado, el diseño de los elementos flotantes sobre nuestras tarjetas es fundamental para proporcionar al usuario una experiencia inmersiva y conocimiento sobre los efectos que está aplicando. A nivel de software no aporta nada a nuestra aplicación. No ejecuta ningún código ni tampoco ayuda a la detección del objeto en sí. Pero será necesario cuidar la perspectiva del usuario y darle el control junto con las mayores facilidades de uso.

Para el diseño de los elementos 3D había multitud de opciones. Hoy en día la oferta de aplicaciones de modelado 3D es enorme, aunque algunas de ellas muy específicas para desarrollo de videojuegos o complejas. Por lo tanto, para esta parte la mejor opción ha sido usar la plataforma de Onshape.

Onshape es una plataforma de modelado 3D basada en la nube que permite crear, editar y compartir diseños CAD (Computer-Aided Design) de manera colaborativa y en tiempo real. Nos será ideal ya que no requerirá de licencias ni descargas. Únicamente necesitaremos registrarnos. [35]

Desde la plataforma podremos crear bocetos seleccionando primeramente uno de los planos. En nuestro caso será el “Top”. Y mediante las herramientas disponibles, crearemos un texto con el tamaño que deseemos. Cabe destacar que el tamaño realmente es irrelevante, ya que dentro de Unity se puede reescalar.

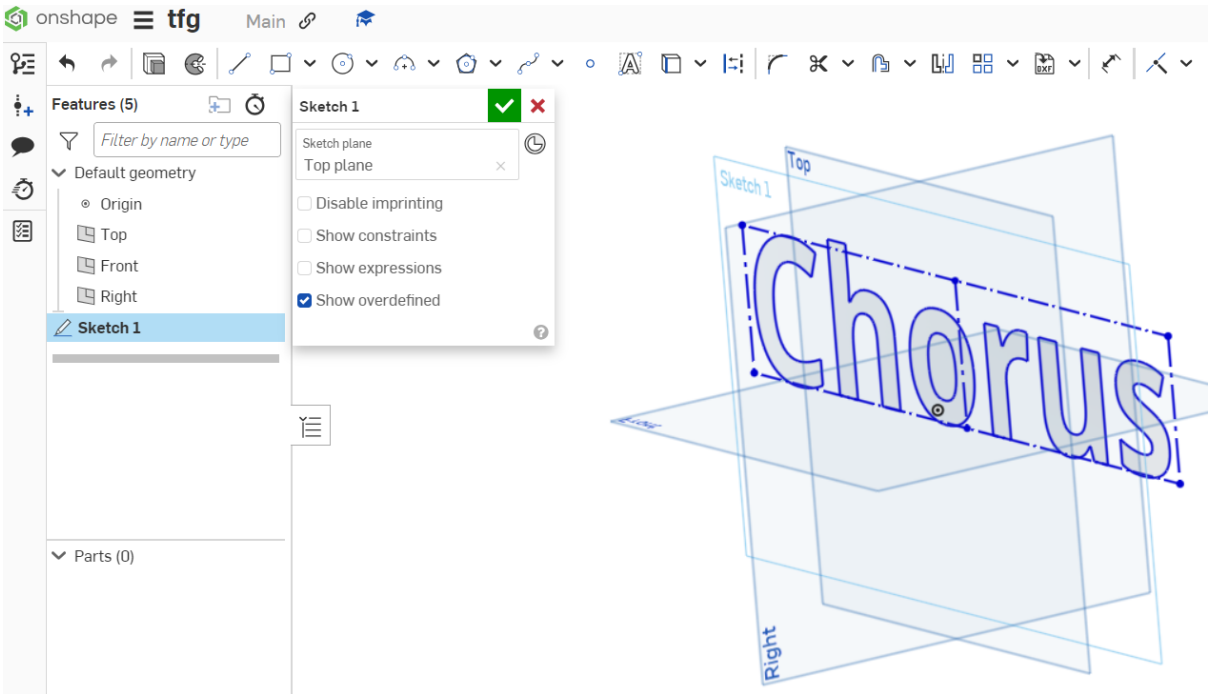


Figura 71: Objeto en OnShape 2D

Inicialmente nuestro boceto estará solamente en el plano 2D. Tendremos que utilizar la herramienta de “Extrudir” para darle la forma que buscamos. Seleccionando la dirección y la profundidad

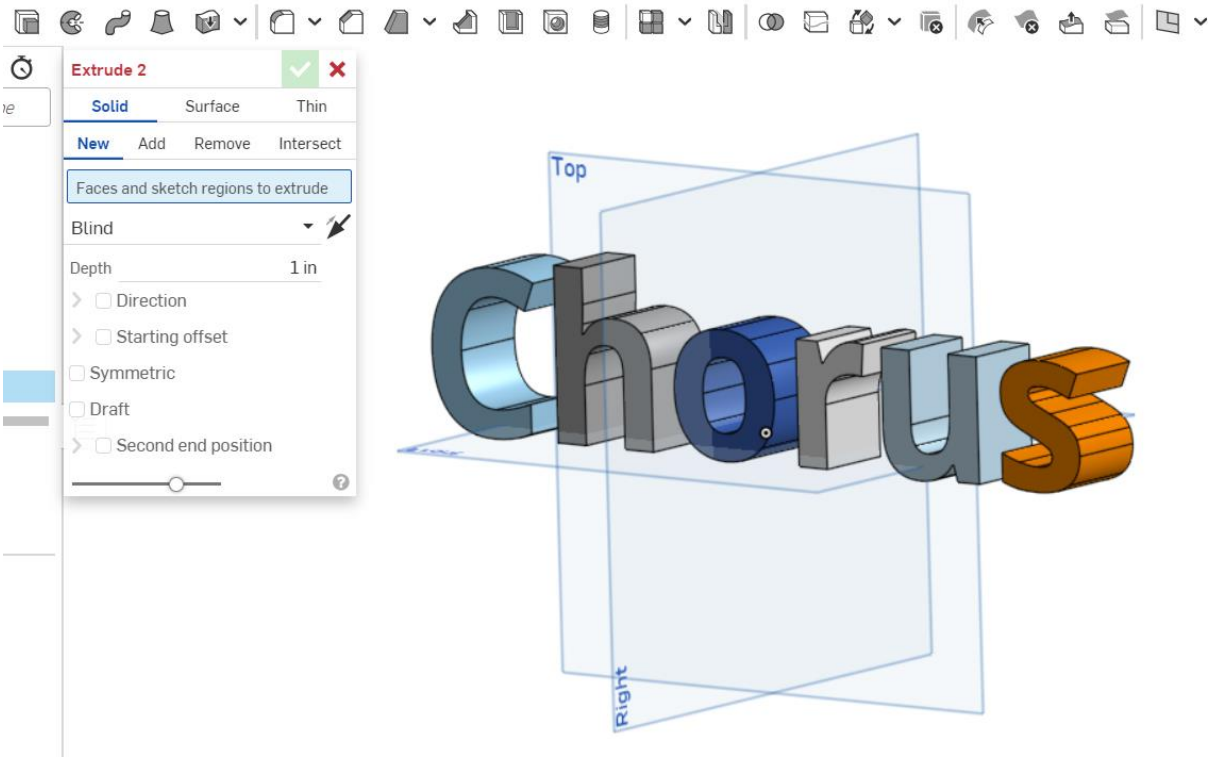


Figura 72: Objeto en OnShape 3D

Obteniendo como resultado la palabra queramos plasmar en Unity. Ahora será necesario unificar cada una de las letras de nuestra nueva palabra como un único objeto.

De este modo será exportable para Unity como un elemento individual, en vez de ser tratado letra a letra.

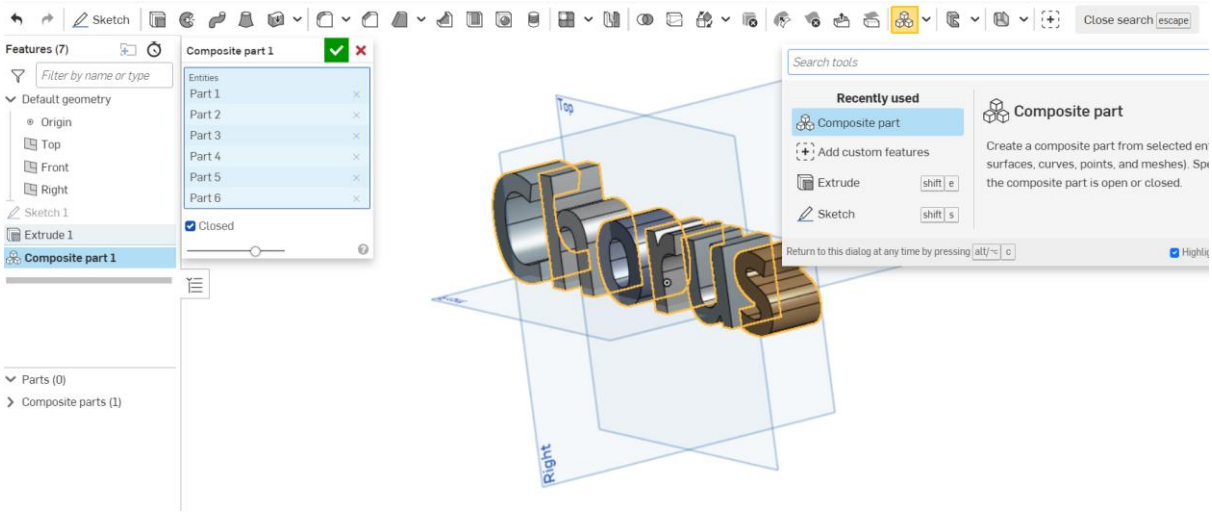


Figura 73: Transformación a un único objeto

Con la opción de “Composite part”, terminaremos el proceso de creación. Todas las letras han sido seleccionadas y transformadas a un único objeto. Finalizando con la exportación del boceto como un Asset para Unity

5. Discusión y conclusiones.

Durante este capítulo se presentará el resultado final del proyecto, nuevas implementaciones respecto a la interfaz gráfica, más opciones para el usuario, limpieza y depuración del código.

5.1 Interfaz Gráfica

Con la idea de hacer una sencilla de usar para el usuario, se ha añadido una interfaz gráfica. Se han usado componentes de Unity para construir interfaces gráficas, como botones, paneles, menús desplegables y barras de progreso, se han podido personalizar fácilmente para adaptarse a nuestra mesa de mezclas.

Adicionalmente han incorporado 3 nuevas escenas:

- **Pantalla de inicio:** Se carga siempre como primera pantalla, una vez iniciamos la aplicación. En ella tendremos 2 botones interactivos que nos permitirán acceder a 2 conjuntos de fuentes sonoras distintas precargas. Con la finalidad de que el usuario tenga varias posibilidades de mezclas y pruebas.

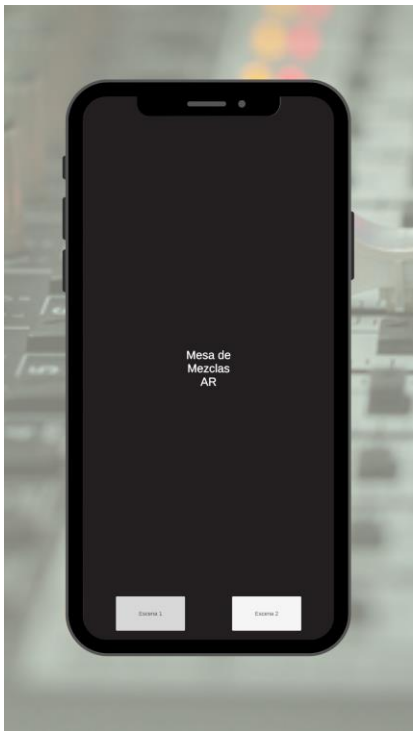


Figura 74: Primera versión Pantalla de inicio

- **Pantalla de fuentes 1:** Será una de las escenas que tendremos después de presionar el botón izquierdo del inicio. Será un entorno visualmente igual a la pantalla 2, pero el código interno será diferente, ya que esta escena tendrá precargadas un conjunto de canciones diferentes al de la pantalla 2. En este entorno tendremos un botón inferior que nos permitirá volver al menú principal.

- **Pantalla de fuentes 2:** Se accederá a este entorno desde el botón derecho de la pantalla de inicio. Igualmente descrito antes, esta pantalla solo se diferenciará con la pantalla 1 en las fuentes precargadas. Todos los efectos y componentes serán los mismos.

5.2 Resultado de las tarjetas finales

Se desarrollaron varias opciones para elegir como tarjetas. Todas comparten un fondo y nombre personalizado. Pero las diferencias de cada opción son:

- Primera opción: Es una figura circular con un rectángulo blanco exterior y un contorno negro rodeando el patrón de fondo.
- Segunda opción: El patrón ocupa todo el diseño rectangular de la tarjeta. Eliminamos el contorno y solo insertamos el diseño vectorizado del efecto
- Tercera opción: El patrón vuelve a ocupar toda la tarjeta. Pero esta vez se rodea la figura vectorizada de una circunferencia con puntas blancas, para graduar así la intensidad del efecto rotándose.

Figura 75: Primeras versiones de las tarjetas

.

Aquí podría entrar la decisión subjetiva de cada persona, pero la finalidad del diseño de estas, era la de facilitar el reconocimiento a Vuforia, por lo que para salir de dudas de cual seleccionar, usaremos la herramienta que nos proporciona la misma plataforma, el Target Manager.

Esta herramienta nos permitirá crear y gestionar bases de datos de targets de AR, que pueden incluir imágenes, objetos 3D y otros tipos de datos que Vuforia puede reconocer y rastrear en tiempo real. Las imágenes serán procesadas para evaluar su calidad y generar datos de rastreo. Una vez procesadas, se organizan en bases de datos que pueden ser descargadas e integradas en nuestra aplicación de Unity. Por esta razón, las tarjetas fueron exportadas desde Adobe Illustrator en formato JPG, consiguiendo que el software de Vuforia las procesara.

Pero realmente lo que más nos interesa de esta herramienta es su función para proporcionar una calificación de la calidad de las imágenes subidas, basada en su capacidad para ser detectadas y rastreadas de manera efectiva por el motor de la app.

Figura 76: Diseño final de todas las tarjetas

Tras varias pruebas se llegó a la conclusión de que el mejor diseño era el último. Añadiendo este diseño en todos los bocetos se comprueba que prácticamente todas las tarjetas finales importadas a la plataforma reciben una calificación de 5/5

Target Manager › TFGLuisOrtega

TFGLuisOrtega Edit Name

Type: Cloud
License Key: TFG

Targets (21) | Database Access Keys

Add Target

Search

Image	Target Name	Rating	Recos	Status	Date Modified
	VFlangingDepth	★★★★★	0	Active	Jun 13, 2024
	PasoBajo	★★★★★	0	Active	Jun 12, 2024
	Original	★★★★★	0	Active	Jun 12, 2024
	Chorus	★★★★★	0	Active	Jun 12, 2024
	VcompresorRatio	★★★★★	0	Active	Jun 10, 2024
	Musica	★★★★★	0	Active	Jun 10, 2024
	ParamEQ	★★★★★	0	Active	Jun 10, 2024
	Echo	★★★★★	0	Active	Jun 10, 2024

Figura 77: Calificación de Vuforia.

5.3 Funcionamiento de la aplicación

Una vez tratadas todas las pantallas y diseños, el aspecto final de la aplicación en funcionamiento será muy simple y eficiente. Las fuentes de audio que hayamos incorporado desde Unity, estarán sonando continuamente, aunque inicialmente sin volumen, hasta que aparezcan en cámara. La escena de Unity, con las tarjetas cargadas, se verá limpia y sencilla, garantizando una experiencia de usuario intuitiva y directa.

Al momento en que uno de nuestros diseños de Illustrator aparezca en la cámara, la aplicación lo detectará y proyectará sobre esta tarjeta el objeto 3D creado desde OnShape. Esto permitirá al usuario visualizar en tiempo real qué efecto y sonido está aplicando. La interacción se hace fluida y dinámica, ya que, al retirar la tarjeta de fuente, el efecto aplicado se quedará guardado en la configuración de la aplicación. Esto permite al usuario aplicar diferentes efectos a cada sonido de manera continua, combinándolos para crear nuevas mezclas musicales originales y creativas sin interrupciones.

Con la existencia de la tarjeta “Original”, restableceremos todos los sonidos a su valor original, pudiendo volver a crear una nueva mezcla sin necesidad de reiniciar la aplicación ni volver a la pantalla de inicio. Y gracias a la vectorización de los diseños, no tendremos problema con supuestos objetos detectados inexistentes dando una experiencia perfecta.

Gracias a la vectorización de los diseños, la aplicación minimiza los problemas de detección de objetos inexistentes. Esto asegura que la experiencia sea precisa y fiable, evitando errores comunes en aplicaciones de realidad aumentada y proporcionando una experiencia perfecta para el usuario. La robustez del sistema de detección se complementa con la capacidad de la aplicación para manejar múltiples tarjetas y efectos simultáneamente, manteniendo una interfaz de usuario clara y respondiendo rápidamente a las interacciones.

5.4 Rendimiento y código.

En las primeras fases de desarrollo de la aplicación, se encontraron varios desafíos relacionados con la organización del código y la estructura de la escena de Unity. Inicialmente, cada GameObject tenía un script dedicado para rotar las tarjetas y cambiar sus parámetros. Este enfoque, aunque funcional, resultó en una gran cantidad de scripts dispersos, lo que complicaba el mantenimiento y la comprensión del código.

Para mejorar la limpieza y eficiencia, todos estos scripts individuales se unificaron en un solo script central. Este nuevo es el responsable de manejar las rotaciones y cambios de parámetros de todas las tarjetas. La unificación no solo ha simplificado la estructura del proyecto, sino que también ha reducido la redundancia del código, facilitando su mantenimiento y actualización. Ahora, con un solo script manejando múltiples tareas, el código es más limpio y más fácil de entender, lo que también mejora el rendimiento de la aplicación, ya que se reducen las llamadas y referencias innecesarias a múltiples scripts, optimizando el uso de recursos del sistema.

Otro desafío significativo fue la complejidad de la escena de Unity. Inicialmente, la escena contenía numerosos GameObjects individuales para cada efecto y fuente de sonido, lo que hacía difícil su gestión y comprensión. Cada efecto y fuente de sonido estaban separados, resultando en una estructura desorganizada y difícil de manejar desde el punto de vista del desarrollo.

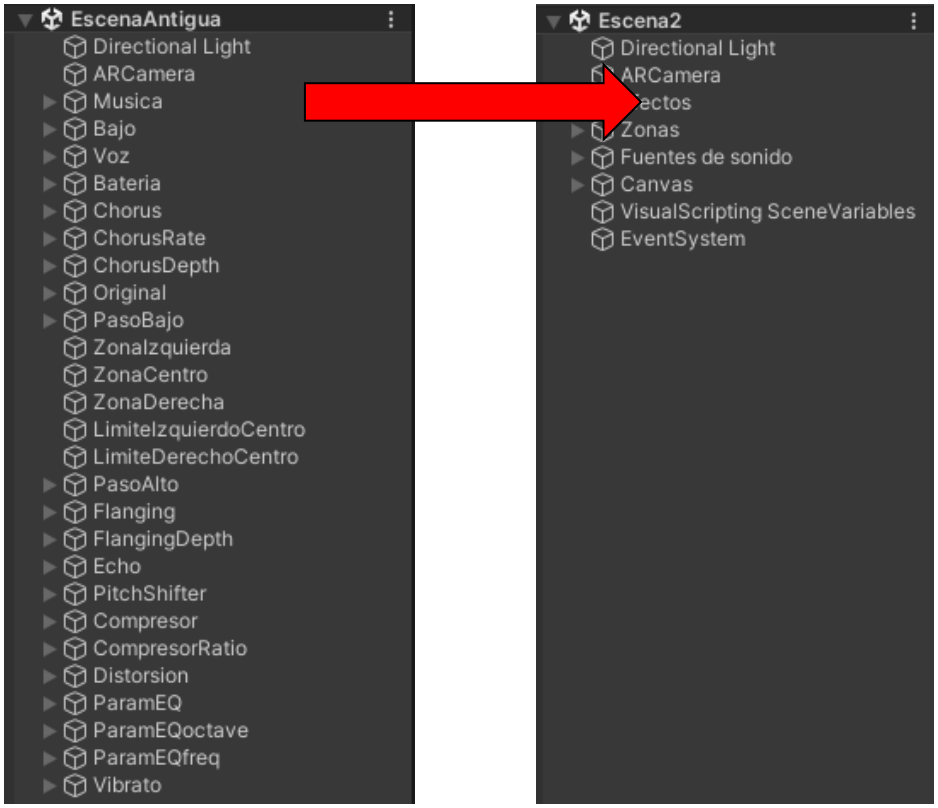


Figura 78: Estructura de las escenas de Unity

Para abordar este problema, se realizó una reestructuración completa de la escena. Todos los efectos y sonidos se unificaron y organizaron en subgrupos lógicos. Esta reorganización no sólo mejoró la claridad visual de la escena, sino que también facilitó la gestión y modificación de los elementos dentro de Unity. Ahora, la escena es más limpia y mucho más fácil de navegar, permitiendo a cualquier desarrollador localizar y modificar componentes específicos con mayor rapidez y eficiencia.

Con estas mejoras en la estructura del código y la organización de la escena se ha conseguido un rendimiento mejorado de la aplicación. Además, la nueva organización permite una mayor escalabilidad, facilitando la incorporación de nuevas funcionalidades y efectos sin comprometer la claridad y el rendimiento del código existente.

5.4 Líneas de futuro

Tras finalizar el desarrollo de nuestra aplicación, se plantean diferentes opciones con la intención de mejorar la experiencia de usuario y el rendimiento.

- Se plantea la posibilidad de incorporar una mayor cantidad de efectos adicionales actualmente inexistentes, ofreciendo así más posibilidades al usuario. Estos nuevos efectos podrían incluir modulaciones avanzadas o reverberaciones especializadas que permitan una mayor creatividad y personalización en las mezclas de audio.

- Se plantea incorporar tarjetas comodín. Estas tarjetas tendrían la función de poder aplicar cualquier efecto de nuestra base de datos y cambiarlo con botones extra en la pantalla. Esto nos daría la ventaja de no necesitar tantas tarjetas en uso, lo cual podría llegar a ser confuso para el usuario. La simplificación de la interfaz mediante el uso de tarjetas comodín mejoraría significativamente la usabilidad y la experiencia del usuario.

Figura 79: Tarjeta comodín.

- Mejoras en la interfaz de Usuario: Se podría mejorar la interfaz de usuario (UI) mediante la implementación de diseños más intuitivos y accesibles. Esto incluiría la introducción de tutoriales interactivos para nuevos usuarios, botones y controles mejorados para la manipulación de efectos, y una visualización en tiempo real de los cambios de parámetros. Además, la interfaz podría ser personalizada según las preferencias del usuario para una experiencia más individualizada.

6. Planos y anexos.

6.1 Anexo I: Diseño corporativo

Enfocar el Trabajo de Fin de Grado desde una perspectiva empresarial permite vislumbrar el potencial y alcance de la aplicación desarrollada. La idea fundamental detrás de esta aplicación es ofrecer una mesa de mezclas virtual integrada en el móvil, eliminando la necesidad de equipos voluminosos y costosos. Este enfoque democratiza el acceso a la producción musical, permitiendo que cualquier persona pueda experimentar como DJ o llevar la tecnología a entornos profesionales como pubs y discotecas.

La aplicación tiene un alcance global, permitiendo que personas de todo el mundo, independientemente de su situación económica, puedan iniciarse en el mundo de la producción musical. Los recursos necesarios para comenzar en este campo suelen ser elevados, por lo que nuestra aplicación representa una solución accesible y práctica. Al reducir los costos y la necesidad de equipos físicos, facilitamos que más usuarios puedan explorar y desarrollar habilidades musicales, fomentando la creatividad y la innovación.

Para fortalecer la identidad corporativa y destacar la aplicación en el mercado, se ha diseñado un nuevo icono y una pantalla de inicio que siguen una línea visual coherente y atractiva.

El icono de la aplicación se ha rediseñado para ser más colorido y característico. Este nuevo icono no solo mejora la estética visual, sino que también facilita la identificación de la aplicación entre las demás en el escritorio del usuario. El logo representa, con una serie de cuadrados de colores, los diseños utilizados en las tarjetas. Este patrón colorido y distintivo permite que nuestra aplicación destaque y sea fácilmente reconocible para los usuarios.

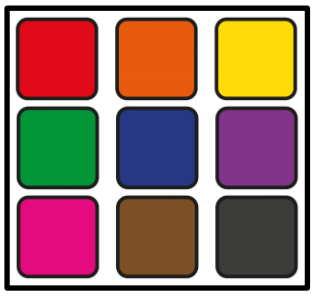


Figura 80: Menú de diseño corporativo.

Continuando con esta identidad corporativa, se ha diseñado un nuevo menú de inicio que sigue la misma línea visual del icono. La pantalla de inicio incorpora los mismos colores, proporcionando una experiencia cohesiva desde el primer momento en que el usuario abre la aplicación. Este diseño no solo mejora la estética, sino que también refuerza la marca y facilita una navegación intuitiva.

Figura 81: Menú de diseño corporativo.

La propuesta de vender una mesa de mezclas dentro del móvil presenta varias ventajas empresariales. En primer lugar, elimina las barreras de entrada para nuevos productores musicales, permitiendo un mayor mercado potencial. Además, la portabilidad de la aplicación permite su uso en diversos entornos, desde el hogar hasta eventos profesionales, ampliando su utilidad y atractivo.

En el ámbito de los negocios, la aplicación puede ser comercializada como una herramienta esencial para pubs y discotecas, permitiendo a estos locales ofrecer experiencias musicales innovadoras sin necesidad de invertir en equipos costosos. Esto puede abrir nuevas oportunidades de ingresos y mejorar la competitividad de dichos negocios en el mercado del entretenimiento nocturno.

La integración de un diseño corporativo sólido y una propuesta empresarial clara no solo mejora la usabilidad y atractivo de la aplicación, sino que también posiciona el proyecto en un contexto comercial viable y prometedor. La aplicación no solo se presenta como una herramienta funcional, sino también como un producto comercialmente atractivo y accesible para una amplia audiencia.

6.2 Anexo II: Manual de usuario

La aplicación móvil “Mesa de Mezclas en AR” es una app para android que se plantea como una alternativa a las mesas de mezclas convencionales, para dar la oportunidad a cualquier persona a realizar su propia música a partir de efectos variables y fuentes sonoras, accesible a todos los públicos.

Este manual tiene como objetivo proporcionar instrucciones claras y detalladas sobre cómo utilizar la aplicación de realidad aumentada, donde su funcionamiento se basará en 22 tarjetas físicas. Cuatro de ellas serán las fuentes de sonido y las dieciséis restantes serán efectos y variaciones de estos.

A continuación, se describen los pasos para instalar, configurar y utilizar la aplicación, así como la resolución de problemas comunes.

6.2.1 Preparación del sistema

Primeramente, debemos de activar las opciones de desarrollador de nuestro teléfono móvil. Esto lo haremos entrando en la pestaña de “Información de Software” a la que accederemos desde “Ajustes”. Desde aquí encontraremos el número el número de compilación de nuestro dispositivo, y solo tendremos que tocar sobre este apartado 7 veces para activarlo.

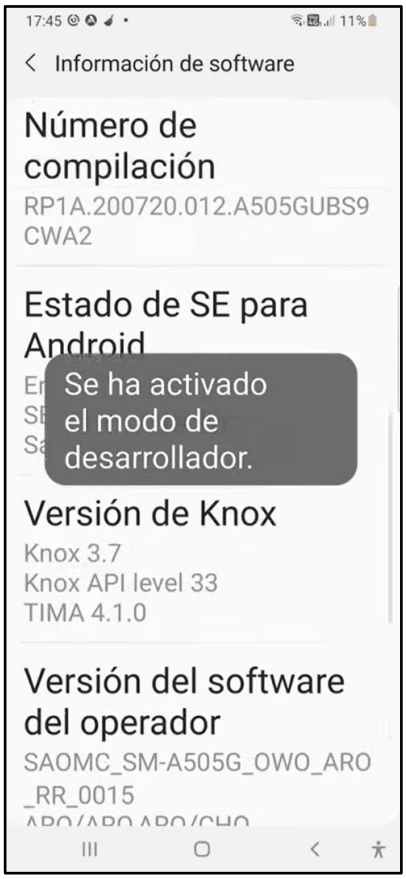


Figura 82: Activación de opciones de desarrollador.

Una vez realizado este paso, tendremos que asegurarnos de que tenemos alrededor de 100MB para realizar la instalación de nuestra APK, y que tenemos la cámara del teléfono operativa.

Debemos de imprimir todas las imágenes que nos servirán como targets para posteriormente poder usarlas en nuestras combinaciones musicales

Y por último, la aplicación solicitará permisos para acceder a la cámara y almacenamiento. Estos son necesarios para la correcta visualización de los objetos en realidad aumentada y para guardar configuraciones personalizadas.

6.2.2 Funcionamiento y fuentes de sonido.

Una vez instalada la aplicación, encontraremos su icono en nuestro escritorio. Solo tendremos que tocar sobre este para inicializarla. Tras una espera de unos segundos, encontraremos la siguiente pantalla de inicio:

Figura 83: Pantalla de inicio final

Donde tendremos para esta demo dos opciones:

· Escenario 1: Encontraremos los sonidos de la canción que hayamos seleccionado desde el entorno de desarrollador de Unity, en este caso será “Me Maten” del artista “C. Tangana”

· Escenario 2: Encontraremos sonidos de otra canción distinta que hayamos seleccionado anteriormente desde el entorno de desarrollador de Unity, en este caso la canción “COMO HAS ESTAU?” del artista Mora.

Ambos escenarios tienen el mismo conjunto de tarjetas, y los efectos funcionarán de la misma manera. La única diferencia es que el conjunto de sonidos: voz, batería, música y bajo, estarán adaptados a la canción del escenario que hayamos seleccionado.

Al seleccionar el escenario que deseemos, entraremos en una nueva pantalla. La cuál accederá a la cámara de nuestro teléfono y nos incorporará un botón en la parte inferior de la pantalla, que nos servirá para volver al menú principal.

Figura 84: Botón “Volver al menú”

Aquí comenzará el proceso de creación de nuestra propia mezcla musical. Todo el funcionamiento se basa en las tarjetas. Si no enfocamos ninguna en la cámara, no ocurrirá nada. Por lo tanto, comenzaremos mostrando en cámara cualquiera de las cuatro tarjetas de fuentes de sonido. En ese momento, comenzará a sonar en nuestro

dispositivo el sonido de la tarjeta que hayamos puesto en pantalla, además se proyectará una imagen 3D que nos ayudará a identificar cual es el sonido que se está reproduciendo.

Figura 85: Tarjeta de sonido con 3D.

Si retiramos la tarjeta de la pantalla o la ponemos boca abajo, el sonido dejará de reproducirse. Ya que solo funciona como hemos dicho anteriormente, reconociendo mediante la cámara, la imagen de la tarjeta, como si fuera un QR. Podremos poner tantos sonidos en la cámara como deseemos, haciendo nuestras propias combinaciones. Pero hay que recalcar que todas las tarjetas, sin importar el tipo que sean, deben de posicionarse una al lado de la otra.

Figura 86: Tarjetas de sonido con 3D.

Todos los sonidos irán en sintonía al mismo tiempo de la canción, para no desincronizar ninguno de ellos, aunque no se estén enfocados en la cámara.

6.2.3 Panning.

El panning es una técnica de producción de audio que consiste en distribuir un sonido entre los canales izquierdo y derecho de un sistema de audio estéreo. Se trata de controlar la posición percibida de un sonido en el campo estéreo, moviéndolo hacia la izquierda, la derecha, o cualquier punto intermedio. Para conseguir una mejor experiencia, el usuario podrá colocar cada fuente de sonido en una posición. Existen 3 zonas distintas: medio, izquierda y derecha.

Tendrá que enfocar con la cámara los sonidos y estando visibles, desplazarlos hacia un extremo de la pantalla u otro. Los efectos son independientes de la posición, posicionarlos en un lado u otro no cambia nada de la aplicación

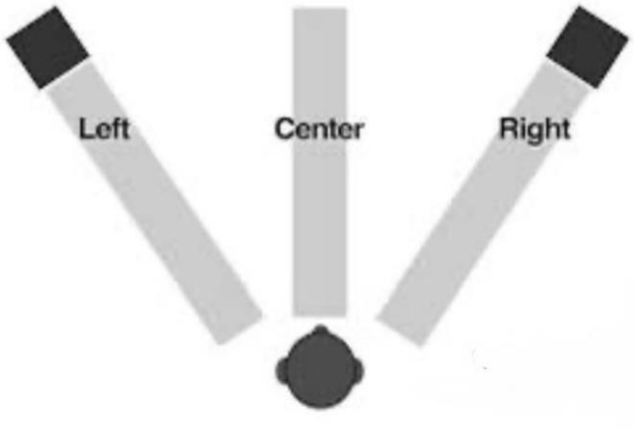


Figura 87: Panning.

6.2.4 Efectos y variantes.

Ahora comenzaremos con el uso de las tarjetas de efectos y sus variantes. Algunas tarjetas de efectos, tienen variaciones. Esto quiere decir, que todas las tarjetas de la misma familia de efectos, aplicarán la modulación principal, como en este caso “Chorus”, pero las tarjetas de variaciones, modifican parámetros distintos.

Ya que, todas las tarjetas de efectos tienen una nueva funcionalidad, el giro. Estas dieciséis tarjetas, dependiendo de su rotación, modificarán el parámetro de su efecto en mayor o menor medida. Siendo la posición normal, el valor más bajo del parámetro que modifique su efecto, y a mayor giro hacia las agujas del reloj, mayor será el valor aplicado. Volviéndose al menor valor de nuevo, cuando restablecemos el giro de la tarjeta a su posición original

Figura 88: Familia de efectos “Chorus”

Todas las tarjetas de la familia “Chorus” aplicarán el efecto, pero cada una de ellas modificará un parámetro distinto, pudiéndose colocar varias a la misma vez para modificar varios parámetros simultáneamente

Para aplicar cualquier efecto, primero debemos de colocar la tarjeta de fuente de sonido que queremos modificar. Puede ser una o varias fuentes, pero el efecto se aplicará exclusivamente a las fuentes de sonoras que tengamos en la cámara. Y deben de colocarse en pantalla simultáneamente con los sonidos. Nunca posicionándose encima de las tarjetas de sonido. Si no, a un lado.

Figura 89: Aplicación de efectos

Podremos modificar sus parámetros rotando la tarjeta del efecto. Una vez retiradas las fuentes sonoras de la pantalla, la modificación que hayamos hecho, se quedará guardada en memoria. Si volvemos a enseñar una tarjeta de efecto, a un sonido que hayamos modificado anteriormente, se sumarán todas las modulaciones.

Existe la tarjeta “Original”, la cual restablecerá todos los valores de los sonidos que se encuentren en pantalla a sus valores originales.

6.2.5 Resultado final

Esta aplicación explota al máximo las posibilidades de creación, colocando una tarjeta de sonido junto con el efecto que deseemos, lo podremos rotar para atenuar o agravar este último, y lo sacaremos de la pantalla habiéndose guardado su configuración. Haciendo este proceso uno a uno, tendremos cada sonido con una configuración diferente, y podremos finalmente poner en pantalla todas las fuentes sonoras, sin ninguna tarjeta de efecto, para poder escuchar la mezcla que nosotros mismo hayamos diseñado. Podremos resetear la configuración de cada tarjeta aplicando “Original” para volver a crear o por si hemos cometido errores.

6.2.6 Solución de Problemas Comunes

- Si la aplicación no reconoce la tarjeta, asegúrese de que la tarjeta esté bien iluminada y dentro del campo de visión de la cámara.
- La configuración que hayamos hecho de efectos, solo se guardará hasta el momento en el que cerremos la aplicación o presionemos el botón de “Volver al menú”. No se guardará entre distintos escenarios.
- No poner las tarjetas encima. Si ponemos las tarjetas encima, las taparemos y solo se reproducirá la que esté visible arriba.
- Si tiene problemas de sonido, asegúrese de que el volumen del dispositivo esté activado y no esté en modo silencio.
- Si coloca una tarjeta de efecto sin ninguna de fuentes sonoras, no se reproducirá ningún sonido.

7. Bibliografía

[1] Jose. (2024b, abril 20). *¿Cómo surgen las primeras mesas de mezclas?* Hifi Center. https://www.hificenter.es/blog-hificenter-como-surgieron-primeras-mesas-mezclas/

[2] Innovae. (s. f.-b). Realidad aumentada. Innovae. https://www.innovae.com/la-realidad-aumentada/

[3] El pasado, presente y futuro de la realidad aumentada - Xnova360. (s. f.-b). Xnova360. https://xnova360.com/historia-de-la-realidad-aumentada/

[4] *Utilizar la app Medidas en el iPhone, iPad o iPod touch - Soporte técnico de Apple (ES)*. (s. f.-b). Apple Support. https://support.apple.com/es-es/102468

[5] App Store. (2024, 14 febrero). *Piano: Just play music!* https://apps.apple.com/us/app/piano-just-play-music/id6477355606

[6] Núñez, I. B. (2021). La realidad aumentada como recurso didáctico en la enseñanza superior. https://www.redalyc.org/journal/3783/378370462010/html/

[7] Stageverse en Meta Quest. (n.d.). Oculus. https://www.meta.com/es-es/experiences/5119490974759740/

[8] Sintetizador AR - Google Arts & Culture. (n.d.). Google Arts & Culture. https://artsandculture.google.com/story/7AUBadCIL5Tnow

[9] Sound On Sound. (2015). The History Of Recording – The Mixing Console. Recuperado de https://www.soundonsound.com/techniques/history-recording-mixing-console

[10] Universal Audio. (2020). Mixing Console Evolution: From Analog to Digital and Beyond. Recuperado de https://www.uaudio.com/blog/mixing-console-evolution/

[11] MANUAL DE SONIDO – 08 LA MESA DE MEZCLAS | Estudio Marhea. (n.d.). https://www.estudiomarhea.net/manual-de-sonido-08-la-mesa-de-mezclas/

[12] MTG - Music Technology Group - Reactable. (n.d.). MTG - Music Technology Group. https://www.upf.edu/web/mtg/reactable

[13] ROTOR |. (2022, October 6). Reactable Legacy. https://reactable.com/rotor/

[14] Plataforma de desarrollo en tiempo real de Unity | Motor de 3D, 2D, VR y AR. (n.d.). Unity. https://unity.com/es

[15] draw.io - free flowchart maker and diagrams online. (n.d.). diagrams.net. https://app.diagrams.net/

[16] Technologies, U. (n.d.). Unity - Manual: Android environment setup. https://docs.unity3d.com/Manual/android-sdksetup.html

[17] Licenses | Engine Developer Portal. (n.d.). https://developer.vuforia.com/develop/licenses

[18] Camera | Vuforia Library. (n.d.). https://developer.vuforia.com/library/platform-support/camera

[19] Technologies, U. (n.d.). Unity - Scripting API: GameObject. https://docs.unity3d.com/ScriptReference/GameObject.html

[20] Image Targets | Vuforia Library. (n.d.). https://developer.vuforia.com/library/objects/image-targets

[21] Unity API reference: ImageTargetBehaviour Class reference. (n.d.). https://developer.vuforia.com/library/sites/default/files/references/unity/classVuforia_1_1ImageTargetBehaviour.html

[22] Aislar los instrumentos de una canción. (n.d.). https://vocalremover.org/es/splitter-ai

[23] Technologies, U. (n.d.). Unity - Scripting API: AudioSource. https://docs.unity3d.com/ScriptReference/AudioSource.html

[24] Technologies, U. (n.d.). Unity - Scripting API: AudioSource.Volume. https://docs.unity3d.com/ScriptReference/AudioSource-volume.html

[25] Technologies, U. (n.d.). Audio Mixer (Mezclador de Audio) - Unity Manual. https://docs.unity3d.com/es/2019.4/Manual/AudioMixer.html

[26] Technologies, U. (n.d.). Unity - Manual: Audio Effects (Efectos de audio). https://docs.unity3d.com/es/530/Manual/class-AudioEffectMixer.html

[27] colaboradores de Wikipedia. (2024, June 26). Panning. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/Panning

[28] Technologies, U. (n.d.). Escenas - Unity Manual. https://docs.unity3d.com/es/2018.4/Manual/CreatingScenes.html

[29] Technologies, U. (n.d.). Unity - Scripting API: SceneManager. https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html

[30] Technologies, U. (n.d.). AudioSource-PanStereo - Unity Scripting API. https://docs.unity3d.com/es/2019.4/ScriptReference/AudioSource-panStereo.html

[31] Nivardo. (2024, April 11). Diccionarios en C#. Oregoom.com. https://oregoom.com/c-sharp/diccionarios/

[32] Adobe Illustrator | Software de Figura digital y vectores gráficos. (n.d.). https://www.adobe.com/es/products/illustrator.html

[33] Audio mixer exposed parameters - Unity Engine - Unity Discussions. (2015, June 30). Unity Discussions. https://discussions.unity.com/t/audio-mixer-exposed-parameters/587342

[34] Pinterest. (n.d.). Pinterest. https://es.pinterest.com/

[35] Onshape, a PTC Business. (n.d.). parallax model loop alt (slower) [Video]. https://www.onshape.com/en/