UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

REALIDAD AUMENTADA PARA LA CREACIÓN COLABORATIVA

Pablo Garijo Contreras

Tutor: Jorge Sastre Martínez

Cotutor: Adolfo Muñoz García

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2018-19

Valencia, 5 de septiembre de 2019

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG NO. ES-CV- 000030

Resumen

La aplicación Soundcool desarrollada en la propia UPV por el Instituto de Telecomunicaciones y Aplicaciones Multimedia (iTEAM) con colaboradores del Instituto de Diseño para la Fabricación y Producción Automatizada (IDF) y del Instituto de Instrumentación para Imagen Molecular (I3M) se concibió como un sistema abierto de creación musical y audiovisual colaborativo que ya ha llegado a diferentes tipos de dispositivos. Además de en smartphones, tablets, PC y Mac, lo podemos encontrar en dispositivos de realidad aumentada como las HoloLens de Microsoft.

La tecnología de la realidad aumentada es relativamente reciente. Desde los últimos años, la eficiencia y potencia del hardware ya comienza a permitir que la realidad aumentada llegue a dispositivos tanto comunes como específicamente diseñado para ella. Este es el caso de las HololLens, que ya fueron utilizadas en Soundcool por primera vez para una performance en el MEVArt de 2018. A partir de esta misma aplicación surge la idea de trasladarla a dispositivos móviles, con vistas a poder utilizarlos como ventana para visualizar en tiempo real los elementos que conforman el entorno virtual generado por las gafas. Se desarrollará una aplicación con Unity3D, que es una de las herramientas estándar para desarrollos de este tipo. La aplicación permitirá crear un espacio virtual en un entorno de realidad aumentada en donde los objetos envíen mensajes OSC en tiempo real al interactuar con ellos. OSC es un protocolo de comunicaciones pensado para compartir información musical entre diferentes dispositivos multimedia sobre una red inalámbrica, y Soundcool lo utiliza para comunicar smartphones y tablets con la aplicación que funciona desde el ordenador. La librería ARFoundation de Unity3D envuelve las librerías de realidad aumentada de Google y Apple para poder desarrollar simultáneamente para plataformas iOS y Android. Por esto se acometerá el proyecto utilizando esta librería para llevar a cabo un desarrollo multiplataforma.

A lo largo del desarrollo de este proyecto se comprobará cómo éste es un campo constantemente cambiante al que uno debe adaptarse continuamente. Esta tecnología está en proceso de maduración, lo cual, por otro lado, añade enjundia e interés a llevar a cabo un proyecto de este ámbito en este momento.

Resum

L'aplicació Soundcool desenvolupada en la mateixa UPV per l'Institut de Telecomunicacions i Aplicacions Multimèdia (iTeam) amb col·laboradors de l'Institut de Disseny per a la Fabricació i Distribució automatitzada (IDF) i del Institut d'Instrumentació per a la Imatge Molecular (I3M) es va concebre com a un sistema obert de creació musical i audiovisual col·laboratiu que ja ha arribat a diversos tipus de dispositius. A més de en smartphones, tablets, PC i Mac, la podem trobar en dispositius de realitat augmentada com ho són les HoloLens de Microsoft.

La tecnologia de la realitat augmentada es relativamente recent. Des d'els últims anys, la eficiència i potència del hardware comença a permetre que la realitat augmentada arribe a dispositius tant comuns com específicament disenyats per a ella. Aquest és el cas de les HoloLens, que ja foren introduïdes en Soundcool per primera vegada per a una performance en el MEVArt de 2018. A partir d'aquesta aplicación sorgix la idea de traslladar-la a dipositius mòbils, de manera que es puguen utilitzar com a finestra per a visualitzar en temps real els elements que composen l'entorn virtual generat per les ulleres. Es desenvoluparà una aplicació mitjantçant Unity3D, que es una de les eines estàndart per a projectes d'aquest tipus. L'aplicació permetrà crear un espai virtual en un entorn de realitat augmentada on els objectes envien missatges OSC en temps real al interactuar amb ells. OSC es un protocol de comunicacions pensat pero a compartir informació musical entre diferents dispositius multimèdia sobre una xarcia inalàmbrica, i Soundcool l'utilitza per a comunicar smartphones i tablets amb l'aplicació que funciona des de l'ordinador. La llibreria ARFoundation de Unity3D embolica les llibreríes de realitat augmentada de Google i Apple per a poder programar alhora per a plataformes iOS i Android. És per això que s'amprarà aquesta llibreria, per a dur a terme un desenvolupament multiplataforma.

Al llarg del desenvolupament d'aquest projecte es comprobarà com un deu adaptar-se constantment a aquest camp d'estudi que sempre es troba canviant. Aquesta tecnologia està encara en un procés de maduració, el qual, per altra banda, afegix contingut i interés a dur a terme un projecte d'aquest àmbit en aquest moment.

Abstract

The Soundcool application, which was developed in the UPV by the Institute of Telecommunications and Multimedia Applications (iTEAM) with the collaboration of the Institute for Automated Production and Fabrication (IDF) and the Institute of Instrumentation for Molecular Imaging (I3M) was conceived as an open system for collaborative music and audiovisual creation that has already been released for different types of devices. Along with smartphones, tablets, PC and Mac computers, we can find it in augmented reality devices such as Microsoft HoloLens.

The augmented reality technology is relatively recent. The efficiency and power hardware has got during the last couple of years is starting to allow augmented reality to be implemented in common devices as well as in those specially designed for it. That's the case for HoloLens, which were already used by Soundcool for the first time for a performance in the 2018 edition of MEVArt. From this application the idea came up to the team to port it to mobile devices, foreseeing its usage as a window to visualize the elements in the virtual space created by the glasses in real time. An app will be developed with Unity3D, which is a industry standard for this kind of projects. The application will allow its user to create a virtual space in an augmented reality environment where the objects will send OSC messages when interacting with them. OSC is a communications protocol created to share musical information between different devices via wireless network, and Soundcool uses it to set communications between smartphones and tablets and the main Soundcool app working in the computer. ARFoundation is a Unity3D library that wraps Google's and Apple's augmented reality libraries, which allows simultaneous mobile development for iOS and Android. That's the reason why the project will be using this library, to develop a multiplatform application.

Along the development process we will find out how this is a study field where changes are steadily taking place and that demands continuous adaptation. This technology is still in a maturation process, which is also interesting for a project like this and gives more value to it in this moment of time.

Índice

Capítulo 1. Introducción

1.1 La Realidad aumentada en el contexto actual

Gracias al avance en términos de hardware en los últimos años, las tecnologías de realidad virtual y aumentada han ido entrando poco a poco en las vidas tanto de usuarios como de creadores de nuevos contenidos e invenciones. En lo que respecta a esta segunda, ha tenido una penetración más fuerte a nivel comercial dada la posibilidad de utilizarla en dispositivos comunes como smartphones y tablets. Esto supone una ventaja respecto a la realidad virtual, para la que son necesarias unas gafas que no todo el mundo puede permitirse y que todavía sufren ciertas limitaciones como el espacio y los mareos.

Entendemos la realidad aumentada como la creación de un espacio virtual en base al entorno real, y en donde los elementos virtuales se mezclan y pueden interactuar con la realidad. La diferencia esencial con la realidad virtual es que en ésta se introduce al usuario en un entorno totalmente virtual, por lo tanto, aunque pueden parecer tecnologías parecidas, las problemáticas que se afrontan son diferentes. En realidad aumentada, que es la tecnología que se explora en este proyecto, es importante detectar con precisión puntos de referencia y superficies y no perderlos de vista para mantener el entorno virtual anclado al real.

A pesar de haber llegado a dispositivos de más bajo nivel que mucha gente ya posee, también se han diseñado dispositivos específicos de realidad aumentada, como las HoloLens de Microsoft (https://www.microsoft.com/es-es/hololens). Ahora bien, estos instrumentos tienen un enfoque más orientado a aplicaciones laborales, educativas e industriales, mientras que los de realidad virtual se han estado centrando más en el mundo del entretenimiento y los videojuegos. Esto se puede detectar a simple vista, únicamente comparando los vídeos promocionales de dispositivos como HoloLens y HTC Vive (tráiler HTC Vive Pro https://youtu.be/jNjmmLLBGHs, tráiler HoloLens 2 https://youtu.be/eqFqtAJMtYE).

En los últimos tiempos los componentes físicos se han reducido en tamaño y abaratado, los softwares han sido optimizados y recientemente están llegando tecnologías de comunicación tan avanzadas como el 5G. Debido a ello, ésta es una época en la que se están destinando muchos recursos y tiempo al desarrollo de la realidad aumentada y se producen enormes avances y cambios continuamente. Por esto el proyecto ha encontrado algún cambio en el poco tiempo que ha durado su desarrollo. El más importante vino de parte de la propia Microsoft, que presentaba en febrero de 2019 su segunda versión de las HoloLens en el Mobile World Congress de Barcelona, anunciando también un cambio radical en el framework para desarrollar para ellas y que va en la dirección de hacer a todos sus dispositivos más compatibles entre sí. Como la aplicación de Soundcool para HoloLens se desarrolló para la primera versión, esto suponía poco menos que tener que desarrollar la aplicación desde cero para la segunda.

Con todos estos nuevos avances introducidos y sin la posibilidad de adquirir a tiempo un kit de desarrollo de la nueva versión de HoloLens por no estar aún disponible, se hizo un reenfoque hacia una aplicación de realidad aumentada, pero para dispositivos móviles, que pudiera complementar la aplicación “HoloDance” desarrollada por el equipo de Soundcool para la performance HoloSound con estas gafas. Esta performance se preparó como demostración para el MarketLab del Sonar+D del Festival de Música Electrónica Sonar de Barcelona 2018, y se representó también en el festival MEVArt de Música Electrónica y Video Arte de la UPV en su edición de 2018, en la apertura del acto del 50 aniversario de la UPV, y en 2019 en el World Science Festival en Nueva York. Véase como ejemplo el vídeo del 50 aniversario de la UPV en https://youtu.be/V-B1gE448tw.

Figura 1. Aplicación HoloDance en la performance HoloSound del World Science Festival 2019 en Nueva York.

Figura 2. Ejemplo de implementación de HoloLens 2.

1.2 Soundcool

Soundcool es una plataforma para la creación musical, sonora y audiovisual colaborativa creada por la UPV en colaboración con otras universidades e instituciones. El sistema puede descargarse gratuitamente desde su página web (http://soundcool.org/descargas), y consiste en un conjunto de módulos como instrumentos virtuales, reproductores (players) de audio o video, entradas de audio (micro) o video (cámara) en directo, efectos de audio y video, mesas de mezclas, etc. que funcionan en ordenadores Mac y Windows. Dichos módulos se abren desde la aplicación de la figura 3 y pueden interconectarse entre sí. La ventaja principal de Soundcool es que cada uno de estos módulos puede controlarse con los móviles o tablets de los participantes posibilitando la creación colaborativa, con aplicaciones tanto educativas como profesionales. Puede verse una Introducción a Soundcool en https://youtu.be/zoZaVK7ysRM. La comunicación entre los dispositivos y el ordenador se realiza mediante el protocolo OSC u Open Sound Control. El protocolo OSC se trata de un protocolo de

comunicación entre dispositivos que permite, entre otras cosas, enviar mensajes de control. Podría considerarse una evolución de la tecnología MIDI.

Figura 3. Aplicación Soundcool para PC o Mac: módulos de audio a la izq. y de video a la dcha.

La aplicación de Soundcool es muy versátil y para controlarla se pueden utilizar las propias HoloLens. Inicialmente también se podía utilizar el sistema Kinect de Xbox360, pero este dispositivo se haya ya en desuso. Como se comentó anteriormente, al comienzo del proyecto sobrevino el inconveniente del anuncio de las HoloLens 2. Todavía no están disponibles las nuevas gafas pero tampoco valía la pena continuar el desarrollo para el modelo antiguo porque no sería compatible con el nuevo. Por lo tanto, se decidió adelantar trabajo por otro frente desarrollando una aplicación móvil con realidad aumentada y haciendo una versión de HoloDance para móvil. Ésta servirá para complementar una eventual versión de la aplicación para HoloLens 2.

El prototipo que será explicado en este documento puede describirse en una frase como una versión en realidad aumentada de la aplicación Soundcool OSC que permite controlar los módulos de Soundcool en el ordenador mediante el móvil o tablet. Se trata de la versión móvil de la aplicación Soundcool, que envía mensajes OSC a través de un puerto y a una IP definidas por el usuario.

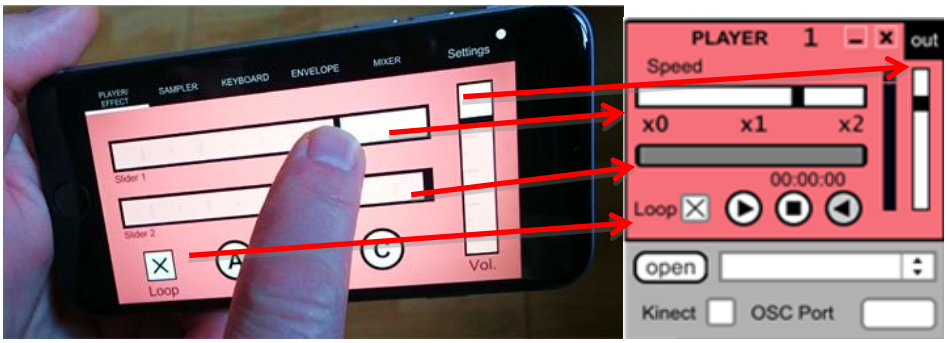

Figura 4. Ejemplo de interfaz táctil del móvil/tablet (izq.) para control wifi del módulo player o reproductor de audio de Soundcool en el ordenador (dcha.)

La versión en realidad aumentada permitirá colocar los objetos de control de la nueva aplicación HoloDance en el espacio y además configurar los mensajes OSC de Soundcool que enviarán cada uno de los objetos al activarlos. Además, añadirá la cualidad de poder enviar mensajes OSC cualesquiera, lo que la hará compatible con cualquier aplicación OSC con su propio mapa de mensajes. También permitirá actuar como las gafas HoloLens (modo Play) para activar los objetos como si fueran las propias gafas.

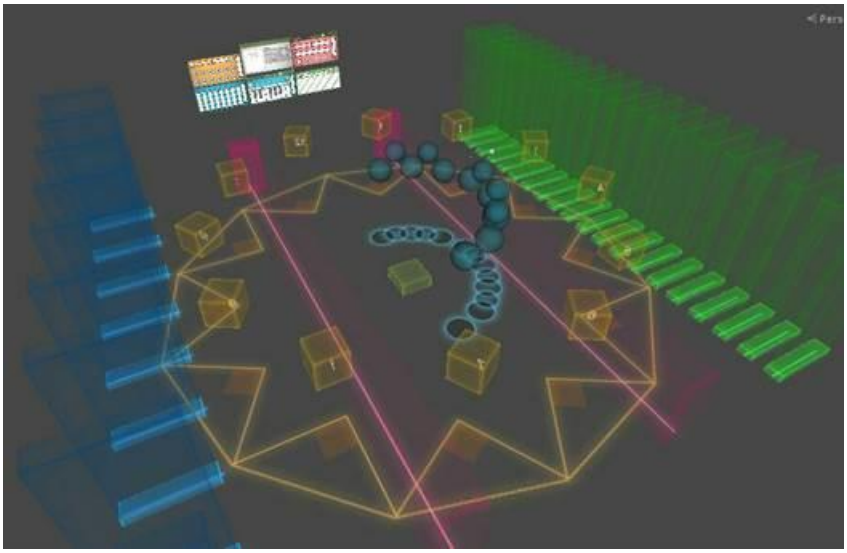

Figura 5. Controles virtuales AR de la aplicación HoloDance original de control de Soundcool: control de sliders (ej. barra verde), teclados de instrumentos virtuales (esferas azules), y reproducción de muestras (cubos amarillos).

Se desarrollará mediante Unity3D y su librería ARFoundation, que permite desarrollar simultáneamente para los dos principales sistemas móviles, iOS y Android.

Figura 6. Muestra de HoloDance para Soundcool en RA (ensayo para el World Science Festival 2019)

1.3 Unity3D y ARFoundation

Unity3D es un motor gráfico para desarrollo de videojuegos creado por Unity Technologies. Es, a día de hoy, uno de los estándares de la industria y tiene detrás una comunidad y un equipo enormes. Eso, sumado al hecho de que es gratuito y permite desplegar cualquier desarrollo en hasta 25 plataformas diferentes lo convierten en una opción realmente sólida para llevar a cabo un proyecto de este tipo.

El lenguaje usado para el proyecto es C#, aunque Unity3D también permite JavaScript y Boo (un lenguaje propio de Unity)

La propia Unity lanzaba la librería ARFoundation en 2018. Como se comentaba previamente, su función es envolver las librerías de realidad aumentada de Apple y Google (ARKit y ARCore, respectivamente) para poder desplegar proyectos en ambas plataformas con el mismo código. Sendas empresas estadounidenses lanzaron a su vez sus propias librerías tan solo en 2017, por lo que este es uno de los factores que hace que en este ámbito haya constantes cambios y mejoras. Sin ir más lejos, la función de anclas en la nube (cloud anchors) todavía no ha sido implementada en ARFoundation. Las anclas en la nube permiten a un dispositivo subir los datos del entorno virtual a la nube de forma que otros puedan bajarlas en tiempo real. Que no estén implementadas limita la continuidad del desarrollo de este proyecto por el momento, porque uno de los objetivos finales es una aplicación para HoloLens que use anclas en la nube para que smartphones o tablets en la sala puedan ver en tiempo real el espacio creado en las gafas. De todos modos, las funciones que ya hay implementadas son más que suficientes para poner en marcha algo de estas características.

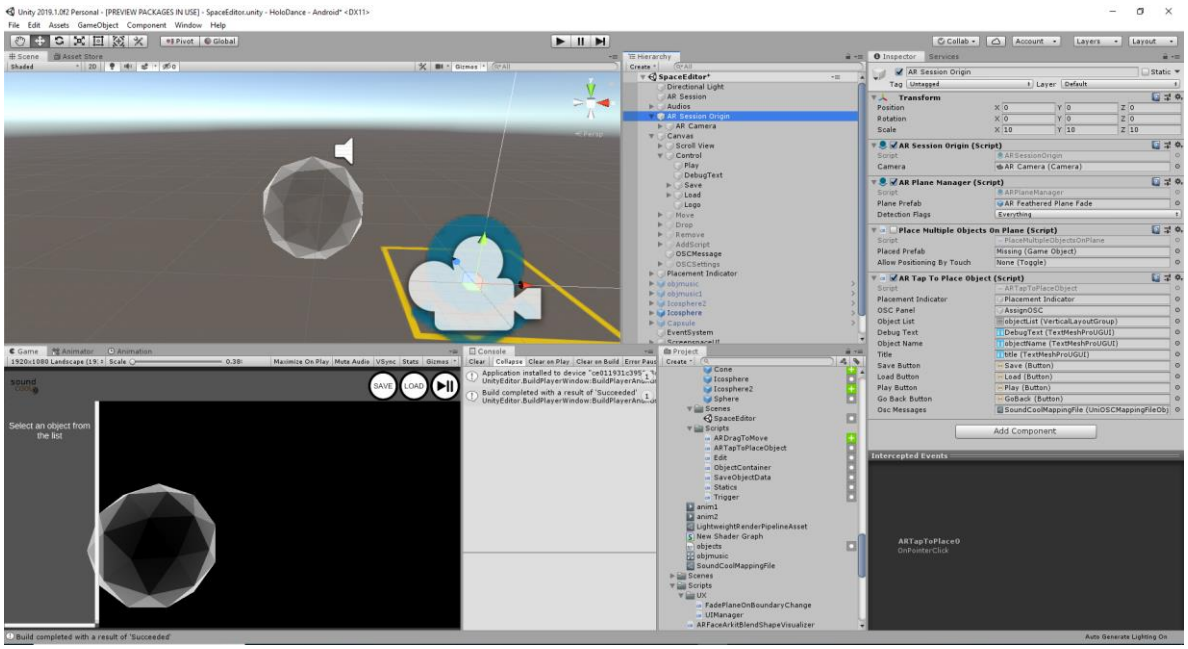

Figura 7. Editor de Unity3D.

Capítulo 2. Objetivos

La finalidad principal de este trabajo es hacer un editor de espacios en realidad aumentada con Unity3D. La versión original de HoloDance que se llevó al festival Sonar tenía un conjunto de objetos fijos en el espacio virtual, y en el marketlab del festival los productores mostraron interés en tener sus propios espacios para sus espectáculos. Esto llevó directamente a la idea de implementar un espacio configurable. Este espacio creado por el usuario se podrá guardar, cargar y modificar, y la aplicación se dividirá en dos modos principales: el de creación del espacio en sí mismo (modo Edit) y el de juego (modo Play). En el primero se podrán colocar objetos sobre los planos detectados tocando sobre la pantalla, seleccionarlos y asignarles un mensaje OSC. El segundo será virtualmente idéntico al HoloDance original, donde una bolita que sigue a la cámara activa el envío de mensajes OSC al tocar los objetos del entorno creado por el usuario. La diferencia es que en la aplicación original los mensajes OSC se corresponden con los de Soundcool y en esta versión serán totalmente configurables, de forma que serán compatibles con cualquier sistema que reciba mensajes OSC.

A otro compañero se le adjudicó la tarea de trasladar la aplicación HoloDance a ARFoundation tal cual para su trabajo final, lo cual incluye implementar en móvil todas las funcionalidades OSC que utiliza Soundcool en las gafas. Esta parte se complementará con este proyecto en tanto en cuanto el modo Play se basará en su trabajo. El proyecto se puede dividir en tres grandes bloques principales:

- Desarrollar una aplicación de realidad aumentada. Como ya se ha comentado, es una tecnología incipiente y con mucho futuro cuyo aprendizaje resulta interesante. También el desarrollo con herramientas muy usadas actualmente como Unity3D.
  Además, esta aplicación de realidad aumentada será un editor de espacios en 3D, de forma que se puedan colocar objetos en el entorno al gusto del usuario. Para esto se contará con la ayuda del Light Weight Render Pipeline (LWRP por sus siglas), que ayuda a la optimización del procesado de la imagen para que no consuma excesivos recursos, lo cual puede acortar la duración de la batería, sobrecalentar el procesador, etc., algo que en un smartphone puede ser muy común. Constará de un modo de edición y un modo de juego y los objetos que el usuario coloque en el entorno podrán ser movidos, borrados y asignados a mensajes OSC en el modo edición. Este modo podría también dividirse en dos submodos: el de colocar objeto (PlaceObject) y el de editar objeto (MoveObject). El primero tendrá una lista de interfaz de usuario con los distintos objetos disponibles y el segundo la sustituirá por un cuadro de entrada de texto para introducir el mensaje OSC deseado.
- Poder guardar un conjunto de objetos y cargarlos más adelante. Mediante una librería de serialización de datos de Microsoft [5], se guardará la información necesaria de los objetos que se encuentran en el escenario en un fichero XML, como son sus coordenadas, nombre y mensaje OSC asociado. También se podrán cargar los objetos desde el fichero para situarlos en el entorno tal y como se habían guardado.
- El modo Play será virtualmente idéntico en su funcionamiento a HoloDance y enviará los mensajes OSC a una dirección IP y un puerto de la aplicación Soundcool en el ordenador, o de cualquier aplicación que pueda recibir mensajes OSC. Aparecerá frente a la cámara del móvil o tablet una bolita con un colisionador que activará el mensaje OSC de cada objeto al chocar con él. Se podrá seleccionar la IP y el puerto por el que se envía el mensaje a través de un panel de interfaz de usuario. Puede verse el funcionamiento del HoloDance oroginal en el vídeo rodado con motivo del 50 aniversario de la UPV referido anteriormente (https://youtu.be/V-B1gE448tw). Se hará uso del desarrollo realizado paralelamente por el otro compañero para que las funcionalidades OSC se implementen de la misma manera, pero en un espacio y con unos mensajes totalmente customizables.

Como objetivo secundario, evidentemente, se busca acumular ciertas bases y conocimientos sobre tecnologías, herramientas, métodos y nuevos dispositivos en un entorno constantemente cambiante con mucho futuro, también de cara al mundo laboral. La familiarización con Unity3D y sus diversas funciones, desarrollar para dispositivos móviles y en realidad aumentada y ver más de cerca cómo

trabajan las HoloLens y evoluciona la industria aportan conocimientos muy útiles para el futuro más allá de este proyecto.

Capítulo 3. Metodología

En este apartado se tratará de explicar cómo se ha decidido gestionar el proyecto, así como de desglosar el desarrollo en diferentes tareas, desde la puesta en marcha hasta la redacción de este documento. La bibliografía utilizada consta de las diferentes documentaciones online de Unity3D y Soundcool, además de los tutoriales proporcionados para UniOSC por la web del desarrollador y XMLSerializer por una wiki de Unity3D. UniOSC es una librería para la implementación de OSC en Unity3D [4] y XMLSerializer permite guardar datos del juego en un fichero XML y también cargarlos. Para la resolución de dudas y errores in situ se consultó distintos foros de desarrolladores como Reddit, StackOverflow o los foros de Unity vía Google. Estos foros son comúnmente utilizados por programadores, desarrolladores, etc. del mundo entero para compartir conocimientos, consultar dudas y pedir ayuda en línea, lo cual constituye un canal de comunicación muy potente en este contexto.

3.1 Gestión del proyecto

Para realizar el grueso del proyecto, que es el desarrollo en Untiy3D, se creó un repositorio en GitHub al que su subió la aplicación HoloDance. El control de versiones mediante Git permite tener en la nube una copia de seguridad de los datos, además de permitir volver a estados anteriores del proyecto. Gracias a su sistema de hitos, Git también permite bifurcar el proyecto en distintas ramas para poder llevar el desarrollo de distintas funcionalidades o por distintos programadores en paralelo sin que se estorben entre sí. A partir de ahí se crearon ramas para desarrollar diferentes bloques del desarrollo para Soundcool, éste concretamente se sitúa en la del editor de espacios (Android_AR_LWRP2019_SpaceEditor). Aunque puede realizarse mediante línea de comandos, hoy en día hay extensiones e interfaces gráficas variadas para el control de versiones mediante Git, en este caso concreto se utilizó SourceTree (figura 8). Desde aquí se pudo trabajar tomando como referencia el otro trabajo realizado paralelamente para trasladar toda la lógica OSC a este modo Play.

A la hora de establecer la comunicación entre los miembros del equipo y los tutores se ha optado por utilizar Microsoft Teams, para mantener un contacto más fluido y poder dejar constancia de las tareas y compartir contenido de forma más sencilla. Teams es una aplicación desarrollada por Microsoft y enfocada al ámbito corporativo que permite una comunicación rápida entre usuarios desde su puesto de trabajo, ofreciendo llamadas de voz y vídeo, compartir archivos, etc. También permite el uso de metodologías de trabajo ágiles mediante la sincronización con tableros de tipo SCRUM para la división del trabajo en tareas como Trello.

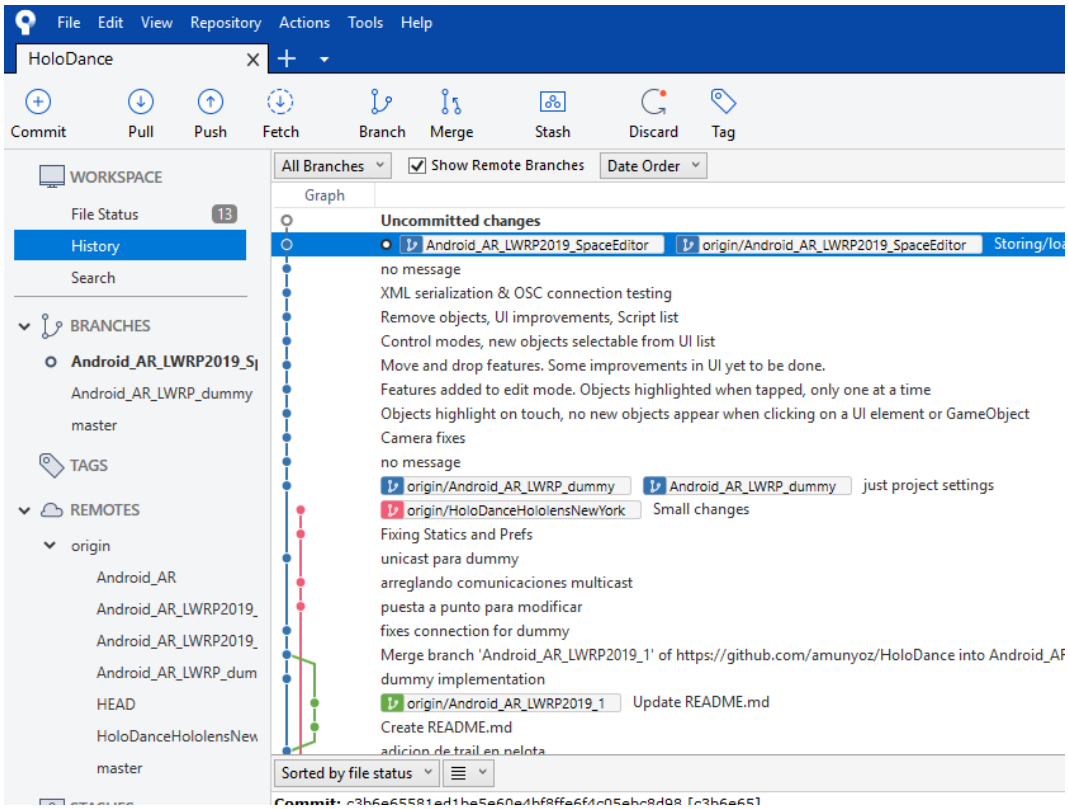

Figura 8. Interfaz de SourceTree.

3.2 Distribución en Tareas

La totalidad del desarrollo de este proyecto puede dividirse en tres grandes bloques de trabajo. Éstos son la división del propio trabajo y la fijación de los objetivos de éste; el desarrollo de la aplicación en Unity3D; y, finalmente, la recopilación de datos y comprobación de resultados y próximos pasos, para la redacción de este documento.

3.2.1 Objetivos y reparto del trabajo

En una primera reunión se determinó en qué va a consistir exactamente el proyecto, se repartieron las tareas y se acordó la metodología de trabajo, que es la mencionada en el punto anterior.

Desafortunadamente, el momento de la puesta en marcha de este proyecto se encontró en un periodo de transición entre la primera y la segunda versión de las HoloLens, para la que además se van a introducir una enorme cantidad de mejoras. Como no se disponía de un kit de desarrollo para HoloLens 2, se acordó cambiar el objetivo del desarrollo a dispositivos móviles. En cuanto al reparto de tareas, este documento trata el bloque de creación de espacios en un entorno de realidad aumentada, con vistas a un desarrollo para la futura versión de las gafas de Microsoft. Es importante el hecho de que unas gafas HoloLens cuestan unos 3500$, mientras que ya hay móviles y tablets compatibles con realidad aumentada por menos de 300€. Hacer un desarrollo enfocado a dispositivos móviles supone abrir el uso de esta aplicación a casi cualquier usuario, ya sea en un contexto educativo o profesional.

3.2.2 Desarrollo en Unity3D

Como ya ha quedado expuesto en el apartado de objetivos, el desarrollo en Unity puede dividirse en tres bloques principales de trabajo: La creación del entorno virtual, la capacidad de guardar el entorno actual y cargar uno guardado, y la de enviar mensajes OSC en tiempo real al interactuar con los objetos en el modo Play. Específicamente, estos tres apartados se llevarán a cabo de la siguiente manera:

- La creación del espacio propiamente dicho, que incluirá colocar y manipular los objetos para el control mediante mensajes OSC. Se comenzará por poner en marcha una aplicación capaz de detectar planos en la estancia real. Para esto se utilizará ARFoundation y previamente se practicará con ejemplos proporcionados por Unity para familiarizarse con la librería y su funcionamiento. Más adelante, se deben poder colocar objetos, seleccionarlos de forma que se active una animación para resaltarlos, borrarlos, moverlos y asociarlos a un mensaje OSC, todo lo cual constituirá el modo Edit. Este modo se divide en dos submodos, uno para colocar objetos y otro para trabajar sobre el objeto seleccionado. Notar que en este modo no habrá más de un objeto seleccionado al mismo tiempo. En el prototipo se han usado unos cuantos objetos con formas geométricas muy sencillas, pero en este sentido la aplicación es escalable: podrán colocarse en la carpeta del editor tantos objetos como se quiera y éstos se listarán en la interfaz.

  En este modo tendremos también un panel de interfaz de usuario con una lista de objetos desde donde se podrá seleccionar cuál se va a colocar. Aparecerán algunas frases de guía para el usuario y botones de guardado, carga y Play. Una vez se seleccione un objeto, aparecerá un cuadro de entrada de texto donde se introducirá el mensaje OSC que se le quiera asociar de cara al modo Play y un botón para añadírselo. También se mostrarán botones para moverlo y borrarlo (figura 9).

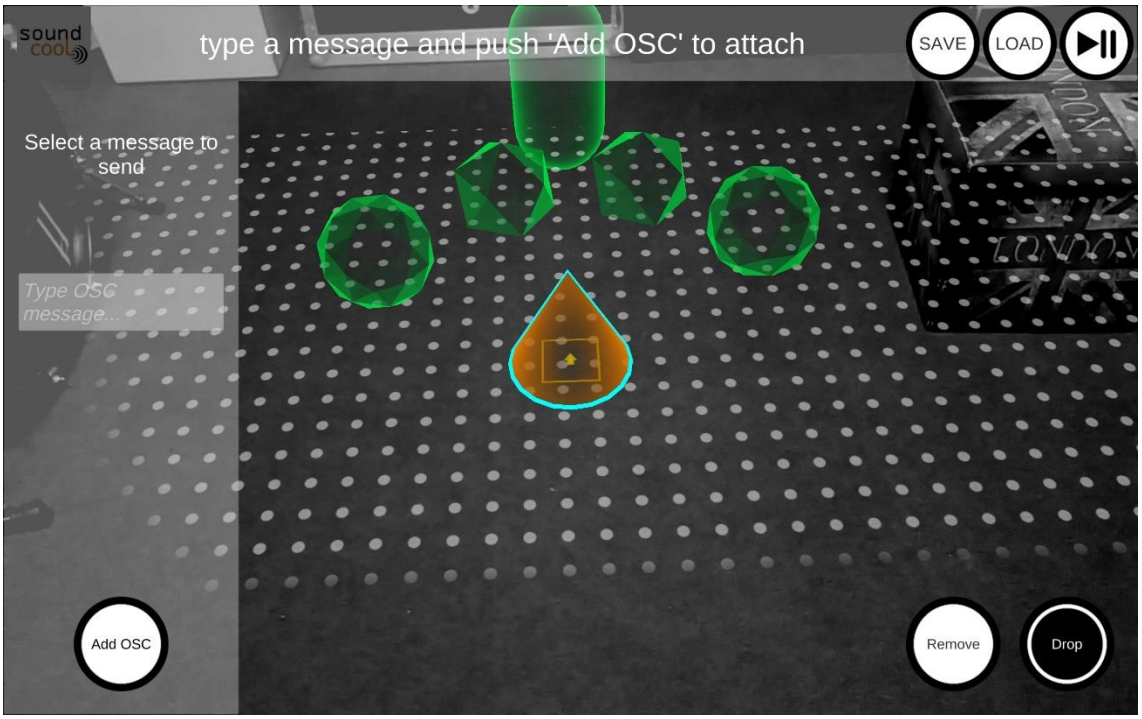

Figura 9. Ejemplo con un cono seleccionado.

- Tras conseguir una aplicación donde se pueden colocar objetos y seleccionarlos, se guardarán y cargarán datos de un fichero XML: las coordenadas espaciales, el nombre y el mensaje OSC de cada objeto [5]. Los botones Load y Save realizarán las funciones de guardado y carga de la configuración, y el botón Play también realizará la función de carga para comenzar este modo con los objetos que haya guardados en ese momento. Los datos se tienen que poder cargar para recolocar los objetos de nuevo en el espacio y añadirles su mensaje OSC. Evidentemente, al volver a colocar los objetos, éstos serán manipulables tal y como se explica en el punto anterior.

- A partir de la aplicación base se creará el modo Play. El punto más importante de este apartado es gestionar la conexión y los mensajes OSC. Siguiendo los tutoriales de la librería UniOSC [4], de la que se sirve Soundcool OSC, se aprenderá su funcionamiento e implementación en Unity3D. Al entrar en el modo Play aparecerá una bolita que se moverá

con la cámara del móvil o tablet y activará la animación de los distintos objetos al colisionar con ellos, enviando a su vez el mensaje OSC que se les hubiese asignado anteriormente. Aparecerá a modo de guía el último mensaje y datos enviados en la esquina superior derecha. En el panel de interfaz de usuario aparecerán dos cuadros de texto que nos permitirán elegir tanto la dirección IP como el puerto por el cual se enviarán los mensajes, y otros dos botones, uno para activar la conexión OSC y otro volver al modo Edit (figura 10, esquina inferior derecha).

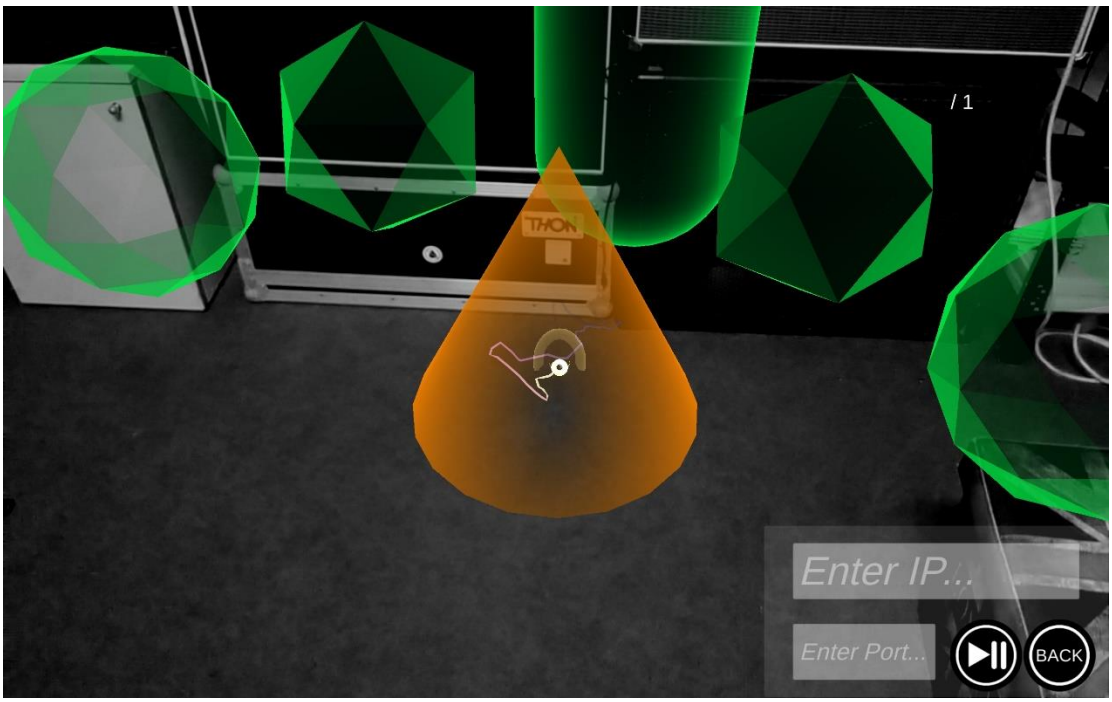

Figura 10. Ejemplo modo Play.

3.2.3 Redacción de la memoria

Tras finalizar el desarrollo de la aplicación y llegar a un prototipo funcional, se recopilará toda la información que fue necesaria para llevarlo a cabo, como las librerías utilizadas, hitos, la metodología seguida y problemáticas encontradas a lo largo del camino, entre otras. Todo con el objetivo de redactar este documento que sirva de guía de consulta para cualquier nuevo desarrollador que continúe con este proyecto. En el apartado de desarrollo se detallarán elementos del código y el editor de Unity3D que proporcionarán la información técnica básica para poder recurrir a este documento ante posibles dudas.

3.3 Diagrama temporal

La distribución temporal del trabajo ha sido bastante sencilla, ya que al ser sólo una persona y ser el desarrollo tan lineal y estar dividido en bloques bastante independientes, no ha sido necesaria una distribución del tiempo muy elaborada. Los primeros meses estuvieron enfocados casi en su totalidad a estudiar las librerías de realidad virtual de Google y Apple y cómo ARFoundation trabaja con ellas, así como a aprender a programar en Unity3D y a manejar el entorno de desarrollo. El siguiente paso fue el desarrollo del primer bloque, que comprende poner en marcha la aplicación base de creación de espacios. Cuando ésta ya estuvo lo bastante avanzada, se introdujeron los otros dos bloques, que realmente son independientes el uno del otro y podían desarrollarse paralelamente.

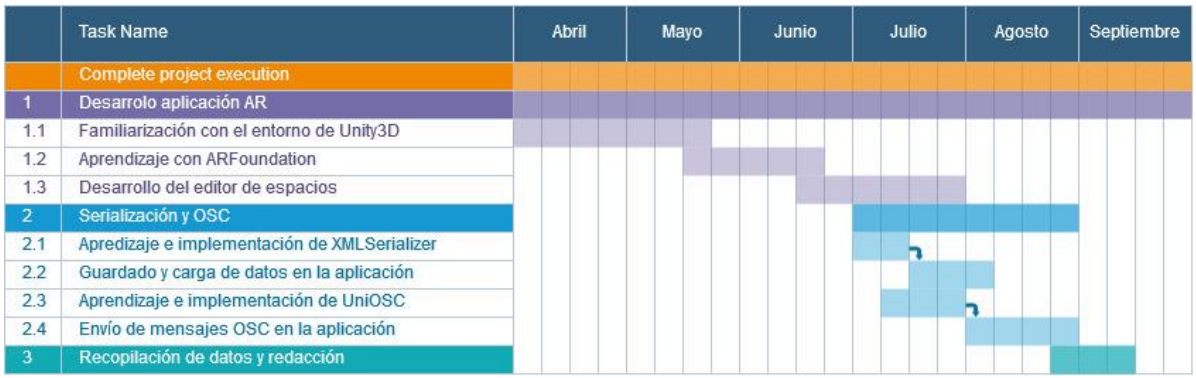

Figura 11. Diagrama temporal.

Capítulo 4. Desarrollo y resultados

Se ha llevado a cabo un prototipo de realidad aumentada con Unity3D con el flujograma que se muestra en la figura 12. Se pueden observar los dos submodos del modo Edit y el modo Play en azul, junto con las funcionalidades disponibles en cada uno de ellos en verde y los botones que las desencadenan en blanco. A lo largo de este apartado se desglosará cada una de las funcionalidades entrando en el código y la lógica implementada a través del editor de Unity3D.

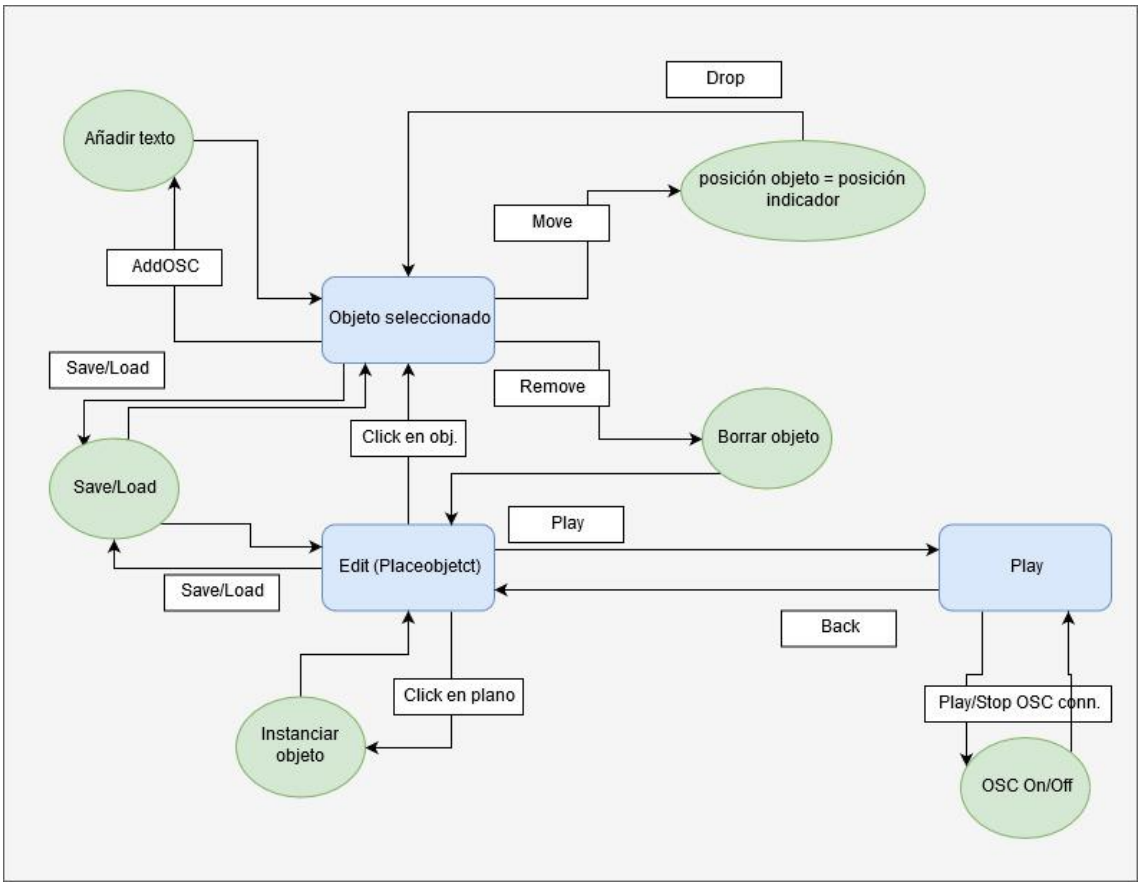

Figura 12. Flujograma de la aplicación

4.1 Primeros pasos con Unity3D y ARFoundation

La puesta a punto para poner en marcha el proyecto empieza con la instalación de los distintos programas que utilizaremos para llevarlo a cabo. El principal de ellos es Unity3D que, como se mencionó anteriormente, se trata de un motor gráfico para desarrollo principalmente de videojuegos, y que está hoy por hoy estandarizado en este tipo de desarrollos. Más concretamente, se ha utilizado la versión 2019.1 de Unity3D sobre Windows. Al tratarse de una plataforma Windows, Visual Studio complementará al motor gráfico para escribir el código en C# y depurarlo a lo largo del desarrollo.

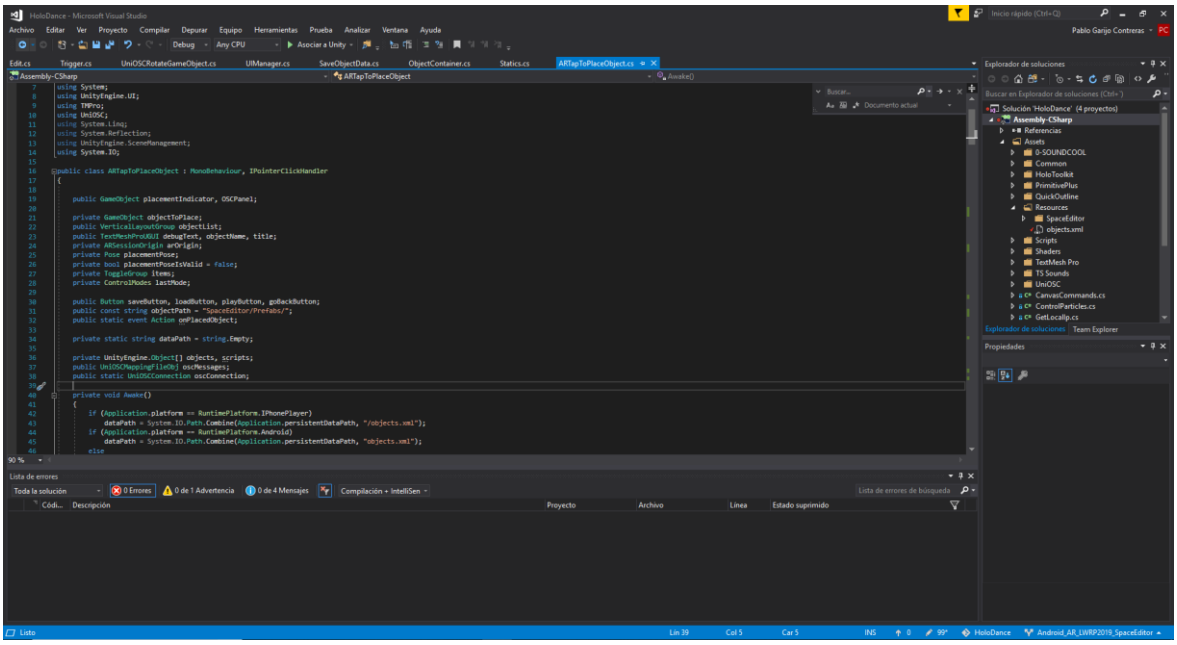

Figura 13. Entorno de Visual Studio.

Una vez puesto en marcha lo más básico, estudiamos guías y realizamos tutoriales para familiarizarnos con el manejo tanto del motor como del entorno de desarrollo y el lenguaje C#. En la puesta a punto fue necesario instalar en Unity los distintos paquetes necesarios para permitir los siguientes pasos. El primero y más importante de todos ellos es ARFoundation, que es la librería de realidad aumentada de Unity para móviles. ARFoundation recoge las librerías de realidad aumentada de Apple y Google de manera que se permiten desarrollos multiplataforma ahorrando la mitad del desarrollo y por lo que decimos que la aplicación debe tener un comportamiento idéntico tanto en dispositivos Android como iOS [3].

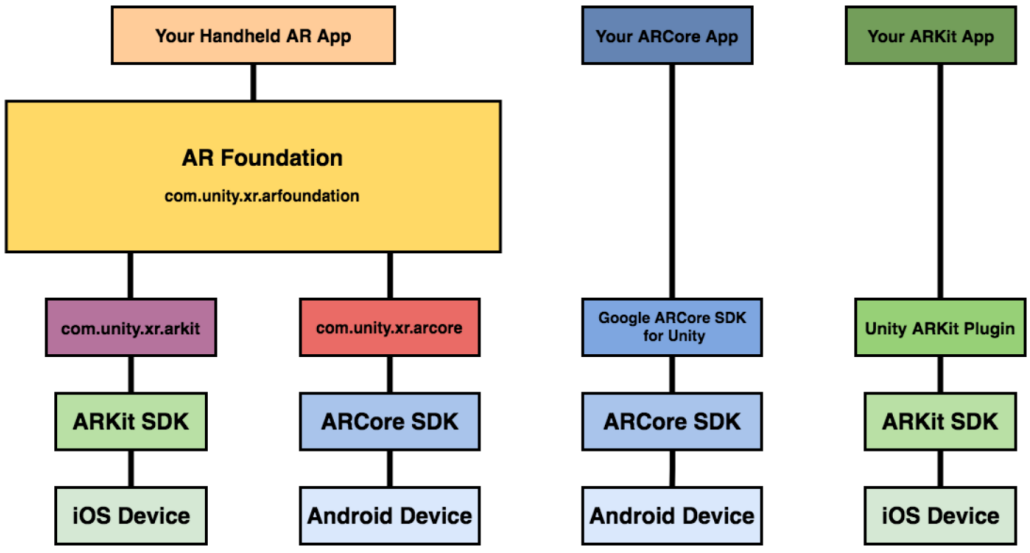

Figura 14. Cómo funciona ARFoundation

Los primeros pasos con ARFoundation pasan por consultar la documentación y seguir pequeños ejemplos y guías para aplicaciones sencillas de detección de superficies, manipulación de objetos, búsqueda de 'feature points' o puntos de referencia (los puntos de la estancia que utiliza ARFoundation para mantener fijo el espacio virtual respecto a ésta), etc. Esto puede acabar resultando muy útil para establecer un punto de partida para empezar a crear el editor de espacios propiamente

dicho, como fue el caso al dar con un vídeo tutorial muy simple que consiste en poner en marcha un programa sencillo que detecta superficies, proyecta un rayo desde el centro de la pantalla (raycast) y muestra un indicador si éste interseccionа una de ellas y, por último, coloca un objeto sobre el indicador al tocar la pantalla [7].

Este pequeño programa ha sido el punto de partida para la creación del editor básico de espacios. Tanto es así que se ha mantenido el diseño del indicador de posición. Con unos retoques, se introduce en la script **ARTapToPlaceObject**, y consta de dos funciones que se llaman en cada fotograma para determinar si el raycast está interseccionando un plano y para situar el indicador en la intersección en caso afirmativo. La script ARTapToPlaceObject se encuentra adherida a ARSessionOrigin y en ella se controla casi la totalidad de la lógica de juego, por lo que aparecerá en más ocasiones. El código de estas funciones prácticamente no ha sido alterado, salvo en un par de detalles que se exponen a continuación:

```
void Update()
    {
        UpdatePlacementPose();
        UpdatePlacementIndicator();

                . . .

    }
```

Tras cada actualización de fotograma de la cámara se llama a las funciones por este orden, ya que la primera determina si la segunda mostrará o no el indicador de posición. Anteriormente también se manejaba la lógica para colocar objetos desde Update, pero se optó por hacer lo mismo a través de un interfaz IPointerClickHandler por comodidad. Se comentará su implementación más adelante.

```
private void UpdatePlacementPose()
    {
        Var screenCenter = Camera.main.ViewportToScreenPoint
                                        (new Vector3(0.5f, 0.5f));

        var hits = new List<ARRaycastHit>();

        arOrigin.Raycast(screenCenter,hits,TrackableType.PlaneWithinPolygon);

        placementPoseIsValid = hits.Count > 0;

        if (placementPoseIsValid)
        {
            placementPose = hits[0].pose;

            var cameraForward = Camera.main.transform.forward;

            var cameraBearing = new Vector3(
                 cameraForward.x, 0, cameraForward.z+1).normalized;

            placementPose.rotation = Quaternion.LookRotation(cameraBearing);
        }

    }

private void UpdatePlacementIndicator()
    {
        if (ModeControl.CONTROL_MODE == ControlModes.Play)
        {
            placementPoseIsValid = false;
        }
        if (placementPoseIsValid)
        {
            placementIndicator.SetActive(true);

            placementIndicator.transform.SetPositionAndRotation
                                        (placementPose.position,
```

```
                                            placementPose.rotation);
        }
        else
        {
            placementIndicator.SetActive(false);
        }
    }
```

UpdatePlacementPose no se ha tocado en absoluto respecto a la versión original del programa, mientras que a UpdatePlacementIndicator se le ha añadido un condicional para situar la posición como no válida en el modo Play.

```
public void OnPointerClick(PointerEventData eventData)
    {
        onPlacedObject();

        if (placementPoseIsValid && !EventSystem.current.
                  IsPointerOverGameObject() && ModeControl.CONTROL_MODE
                   == ControlModes.PlaceObject)
        {
            var newItem = Instantiate(objectToPlace, placementPose.position,
                 placementPose.rotation);

            newItem.name = objectToPlace.name;
       }

    }
```

Al utilizar este interfaz es necesario tocar en un plano detectado, pero supone una gran ayuda a la hora de gestionar cuándo el toque debe hacer unas cosas y no otras según el momento. Se ejecuta OnPlacedObject para detener la animación guía cuando el usuario toque por primera vez. Las tres condiciones para instanciar un objeto en la posición del indicador son que éste esté activo, que no se esté tocando sobre otro objeto de la escena y que el modo en el que se encuentre el jugador sea el submodo de colocar objeto. El cambio de nombre al objeto instanciado sirve para quitarle la coletilla '(Clone)' al instanciarlo, ya que supone cierto problema llegados a la parte de guardado y carga.

4.2 Light Weight Render Pipeline (LWRP) y modo Edit

Todo el proceso de detección de planos, guardado de puntos de referencia e interacción con el entorno en tiempo real supone un gran consumo de recursos de hardware, sobre todo del procesador y más en un smartphone o tableta, por lo que Unity lanzó este "canal de renderizado ligero" (o LWRP por sus siglas en inglés). Esta pipeline nos permite optimizar el rendimiento del renderizado específicamente en dispositivos móviles, de forma que dispositivos de menor rendimiento puedan mover fluidamente las aplicaciones y evitando recalentamientos o costes energéticos exagerados. A efectos de experiencia de usuario sólo se notará que la cámara captura las imágenes reales en blanco y negro, no así los objetos del espacio virtual.

Tras descargar e instalar el paquete de LWRP lo primero que hay que tener en cuenta es añadir el renderizador de fondos para que la cámara del dispositivo funcione con la aplicación. Lo siguiente es conocer que se deben utilizar shaders compatibles con LWRP, así que hemos creado un shader simple para el prototipo de la aplicación y, además, un animador para poder resaltar los objetos al interactuar con ellos o seleccionarlos. Más adelante también se ha añadido un script que perfila los objetos al tocarlos, también para identificar selecciones (Outline). Llegados a este punto y con unas cuantas líneas de código ya tenemos una aplicación que coloca objetos que pueden ser seleccionados o deseleccionados con un toque y que se animan y resaltan en consecuencia.

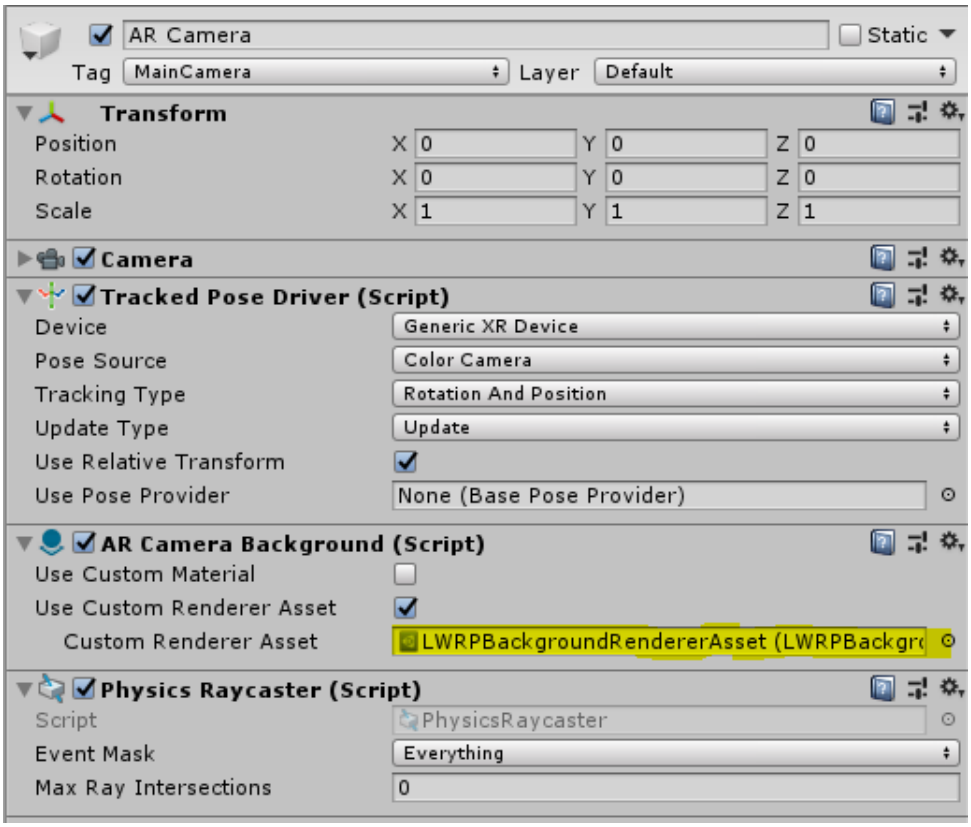

Figura 15. Vista del inspector de ARCamera.

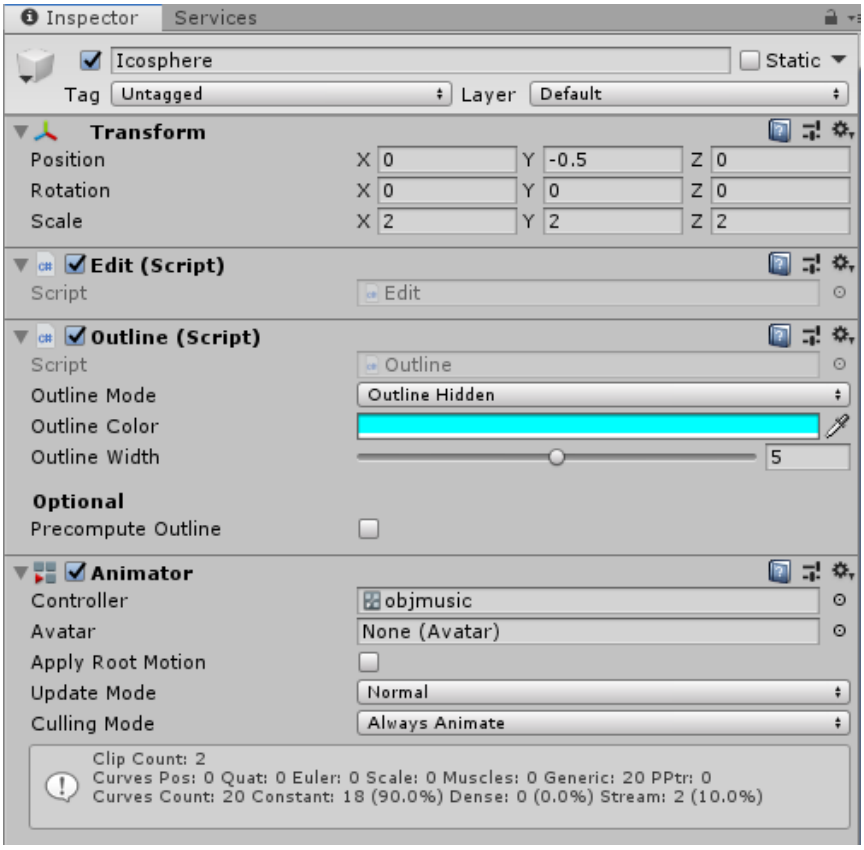

Figura 16. Vista del inspector de cualquiera de los objetos usados

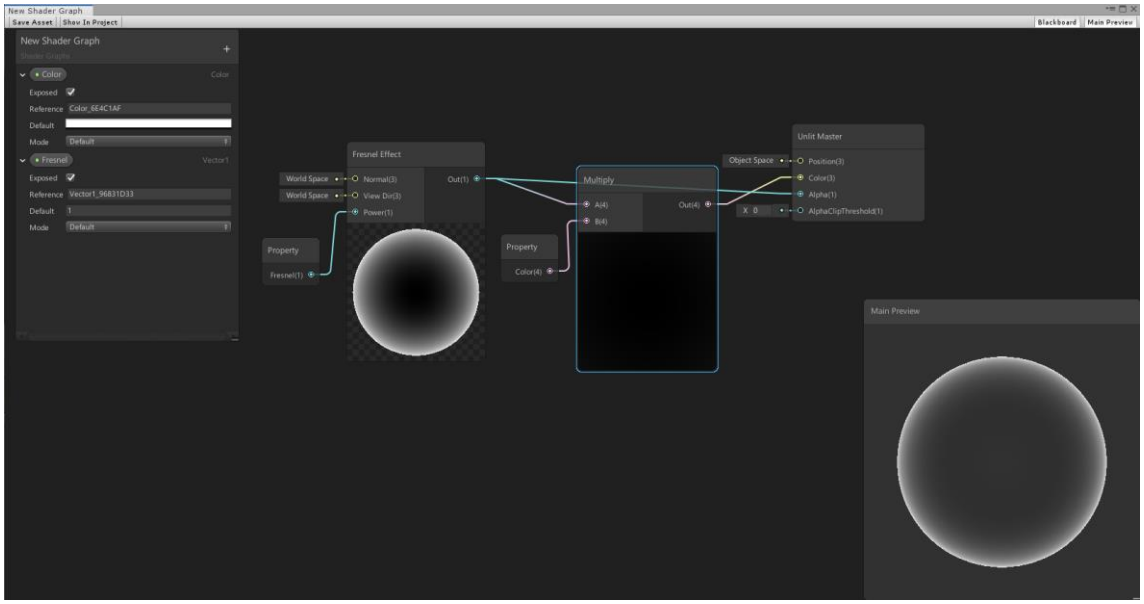

Figura 17. Vista del editor de shaders con el nuevo shader creado.

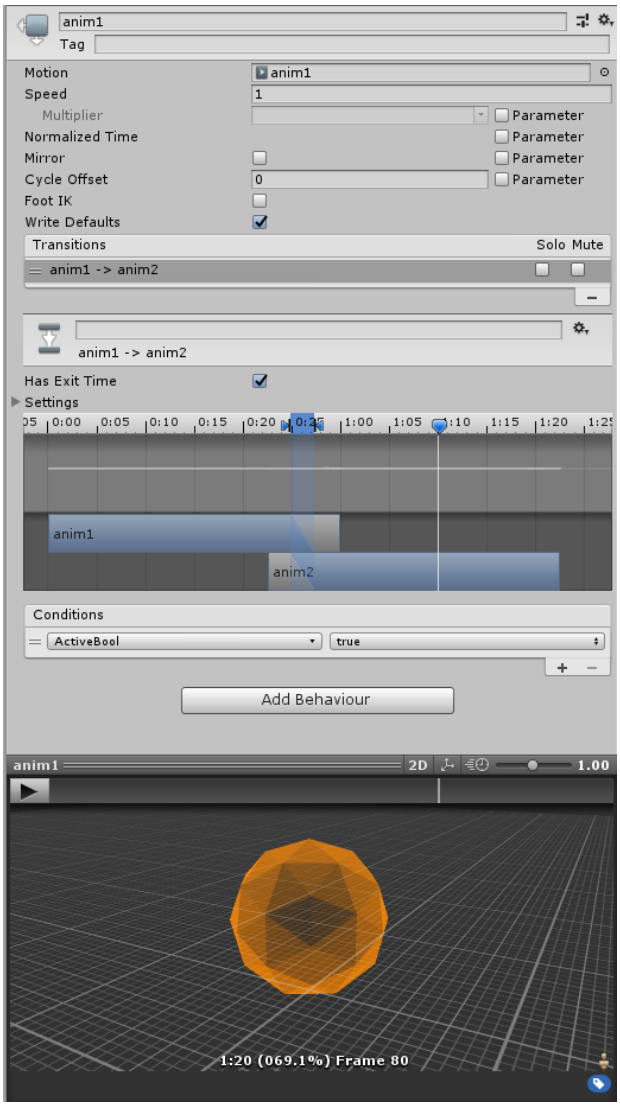

Figura 18. Vista del inspector de una animación del animador objmusic.

4.2.1 Modos de juego

En el fichero **Statics** se puede encontrar, entre otras cosas, un enumerador que se utiliza a lo largo de todo el proyecto para determinar el modo en el que se encuentra el jugador y actuar en consecuencia. Conceptualmente están los modos Edit y Play, pero programáticamente se ha optado por utilizar tres modos, como se ha dado a entender anteriormente:

- **Colocar objeto**. Éste es el modo en el que se podrá seleccionar un objeto de la lista y colocarlo al tocar en la pantalla. Cuando cualquiera de ellos se seleccione se cambiará al modo de editar objeto
- **Editar objeto**. Estará activo cuando un objeto se encuentre seleccionado, y mostrará los botones para moverlo, borrarlo y añadirle el mensaje OSC. También se cambiará la lista por un cuadro de entrada de texto.

Estos dos modos se engloban en lo que se ha definido desde el principio como modo Edit, ya que se conserva el panel superior de interfaz de usuario desde donde se pueden guardar y cargar entornos y pasar al modo Play.

- **Modo Play**. Se esconden todos los paneles de interfaz y botones de la pantalla y se muestra la bolita con colisionador y el panel de control de OSC, que contiene dos cuadros de entrada de texto (IP y puerto) y dos botones (iniciar conexión y volver)

```
public static class ModeControl
{
    public static ControlModes CONTROL_MODE = ControlModes.PlaceObject;
}

public enum ControlModes : int
{
    PlaceObject = 0, EditObject = 1, Play = 2
}
```

4.2.2 La script EDIT

La script Edit va asignada a cada objeto en particular, y será la encargada de realizar todas las interacciones de éste con el usuario. Como puede verse, en Start se inicializan las variables correspondientes al texto del mensaje OSC y a los botones de mover, colocar, borrar y añadir mensaje y se determina el comportamiento de éstos:

```
void Start()
    {
        arOrigin = FindObjectOfType<ARSessionOrigin>();
        name = gameObject.name;

        moveButton = FindObjectOfType<Canvas>().transform.Find("Move")
                .GetComponent<Button>();

        dropButton = FindObjectOfType<Canvas>().transform.Find("Drop")
                .GetComponent<Button>();

        removeButton = FindObjectOfType<Canvas>().transform.Find("Remove")
                .GetComponent<Button>();

        addScriptButton = FindObjectOfType<Canvas>().transform.Find("AddScript")
                .GetComponent<Button>();

        oscInput = FindObjectOfType<VerticalLayoutGroup>().transform
                .Find("AssignOSC/OSCInput/OSC").GetComponent<Text>();

        removeButton.onClick.AddListener(() => {
            if (gameObject.GetComponent<Outline>()
                .OutlineMode != Outline.Mode.OutlineHidden)
            {
```

```
                gameObject.SetActive(false);
                ModeControl.CONTROL_MODE = ControlModes.PlaceObject;
                moveButton.gameObject.SetActive(false);
                dropButton.gameObject.SetActive(false);
                removeButton.gameObject.SetActive(false);
                addScriptButton.gameObject.SetActive(false);
            }
        });

        addScriptButton.onClick.AddListener(() => {
            if (gameObject.GetComponent<Outline>()
                 .OutlineMode != Outline.Mode.OutlineHidden)
            {
                var script = gameObject.GetComponent<Text>() ?? gameObject
                          .AddComponent<Text>();
                script.text = oscInput.text;
            }
        });
    }
```

Se puede apreciar también que a los botones de mover y colocar (moveButton y dropButton) no se les ha asignado ningún comportamiento. Eso es porque sólo se consultará que estén activos, y esto basta con asignarlo desde el editor. Estos dos botones trabajan como si fueran el mismo, pero al pulsar el que está activo, éste se desactiva y muestra el inactivo.

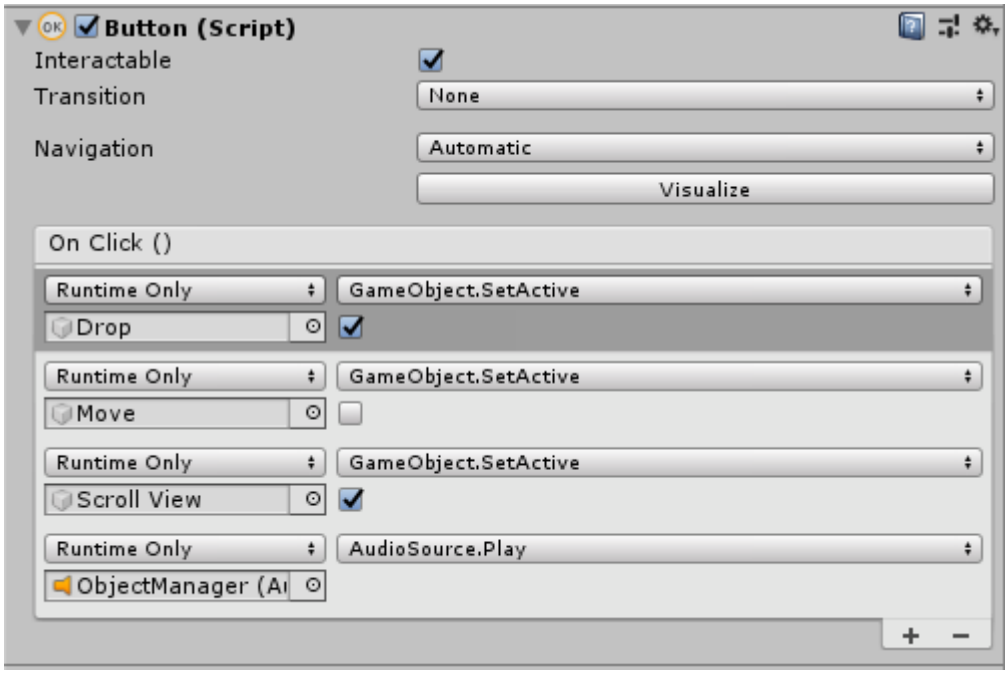

Figura 19. Ejemplo de vista del inspector del botón Move.

En la función Update se observa claramente para qué sirve inicializar estos dos botones en la script. El condicional determina que se ejecutará el código que contiene si el botón de colocar está activo (lo cual quiere decir que se ha pulsado el botón de mover) y, además, el objeto está seleccionado y el jugador en el submodo de modificar objeto. El código muestra que mientras estas condiciones se cumplan el objeto seleccionado seguirá al indicador de posición hasta que se pulse de nuevo el botón de colocar y éste se esconda y muestre de nuevo el de mover:

```
void Update()
    {
        if(dropButton.IsActive() && highlight.OutlineMode
                        != Outline.Mode.OutlineHidden
                        && ModeControl.CONTROL_MODE == ControlModes.EditObject)
        {
            transform.position = placementIndicator.transform.position;
            transform.rotation = placementIndicator.transform.rotation;
```

```
        }

    }
```

Para controlar todo lo relacionado con el estado en el que se encuentra el jugador y las animaciones y resaltados de los objetos, se añade un IPointerClickHandler igual que en ARTapToPlaceObject, de forma que se ejecute el código al tocar sobre un objeto. Se obtienen los componentes de la animación y el resaltado, que en el primer caso es un booleano, y se incializa el objeto que contiene el indicador de posición.

```
public void OnPointerClick(PointerEventData eventData)
    {
        bool B = GetComponent<Animator>().GetBool("ActiveBool");
        highlight = GetComponent<Outline>();

        placementIndicator = GameObject.Find("/Placement Indicator");
        if (!moveButton.isActiveAndEnabled && ModeControl.CONTROL_MODE ==
                ControlModes.PlaceObject)
        {
            GetComponent<Animator>().SetBool("ActiveBool", !B);
            highlight.OutlineMode = Outline.Mode.OutlineAll;
            ModeControl.CONTROL_MODE = ControlModes.EditObject;
            moveButton.gameObject.SetActive(true);
            removeButton.gameObject.SetActive(true);
            addScriptButton.gameObject.SetActive(true);
        }
        else if (B)
        {
            GetComponent<Animator>().SetBool("ActiveBool", !B);
            highlight.OutlineMode = Outline.Mode.OutlineHidden;
            ModeControl.CONTROL_MODE = ControlModes.PlaceObject;
            moveButton.gameObject.SetActive(false);
            dropButton.gameObject.SetActive(false);
            addScriptButton.gameObject.SetActive(false);
            removeButton.gameObject.SetActive(false);
        }

    }
```

Si el botón de mover se encuentra inactivo y el jugador se encuentra en modo de colocar, querrá decir que no hay ningún objeto seleccionado, por lo que el que se acaba de tocar se selecciona. Se pasa a modo de editar objeto y se muestran todos los botones.

En caso de haber ya un objeto seleccionado que no sea el que se ha tocado, no se hace nada, pero si lo es, se deselecciona, se esconden los botones del modo de editar objeto y se vuelve al modo de colocar.

4.3 Interfaz de usuario

Este apartado redundará brevemente en los modos de juego, esta vez centrándose más en el comportamiento de los elementos de interfaz de usuario y mostrando algunas capturas para apreciarlo de forma más sencilla visualmente.

Al inicializar la aplicación se muestra una pequeña animación que guía al usuario para detectar algún plano y colocar un objeto sobre él. Como puede observarse en las diferentes capturas del prototipo, hay una franja en la parte superior de la pantalla que contiene un cuadro de texto con mensajes de ayuda (y que también ha sido de ayuda para depurar código en el dispositivo) y los botones de guardar, cargar y pasar al modo Play. La funcionalidad de los dos primeros tendrá que ver con el apartado sobre serialización, y el segundo esconderá todos los paneles del modo Edit y mostrará el del modo Play y la bolita con colisionador [1].

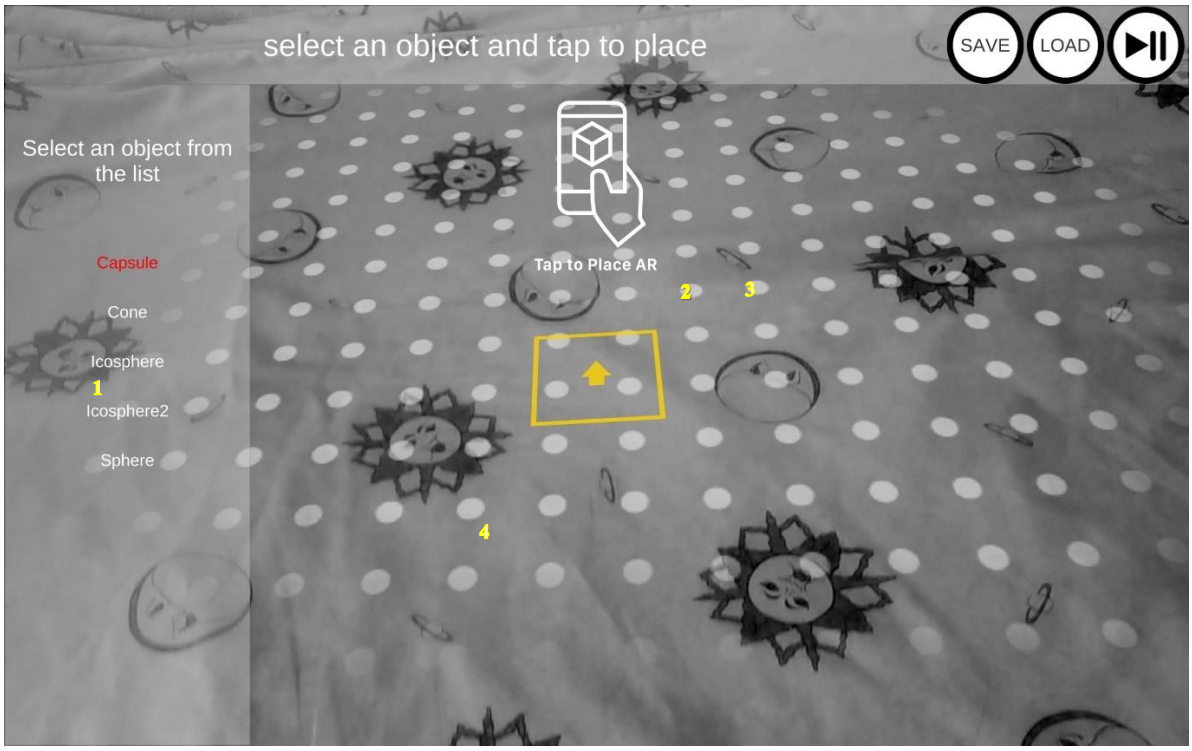

Figura 20. Modo colocar

En la parte izquierda de la pantalla también se coloca un panel con un cuadro de texto a modo de título y otro número indefinido de cuadros que se poblarán programáticamente según el momento. A estas alturas del desarrollo cargará los nombres de los objetos que se encuentren guardados en la ruta especificada en el código y que son, en este caso, los que se podrán colocar seleccionando de la lista.

En el momento en el que se toque uno de los objetos que ya hay en la escena éste cambiará su opacidad y color y su contorno se resaltará. Cuando esto ocurra, aparecerán en las esquinas inferiores derecha e izquierda los botones de mover y borrar, y añadir mensaje, respectivamente. El primero hará que el objeto seleccionado siga al marcador de posición cuando se pulse y, cuando vuelva a ser pulsado, se quedará donde estuviera en ese momento. El botón de borrar quita el objeto de la escena y el de añadir mensaje asociará al objeto seleccionado el texto que haya en el cuadro de entrada que aparecerá en lugar de la lista cuando haya un objeto resaltado.

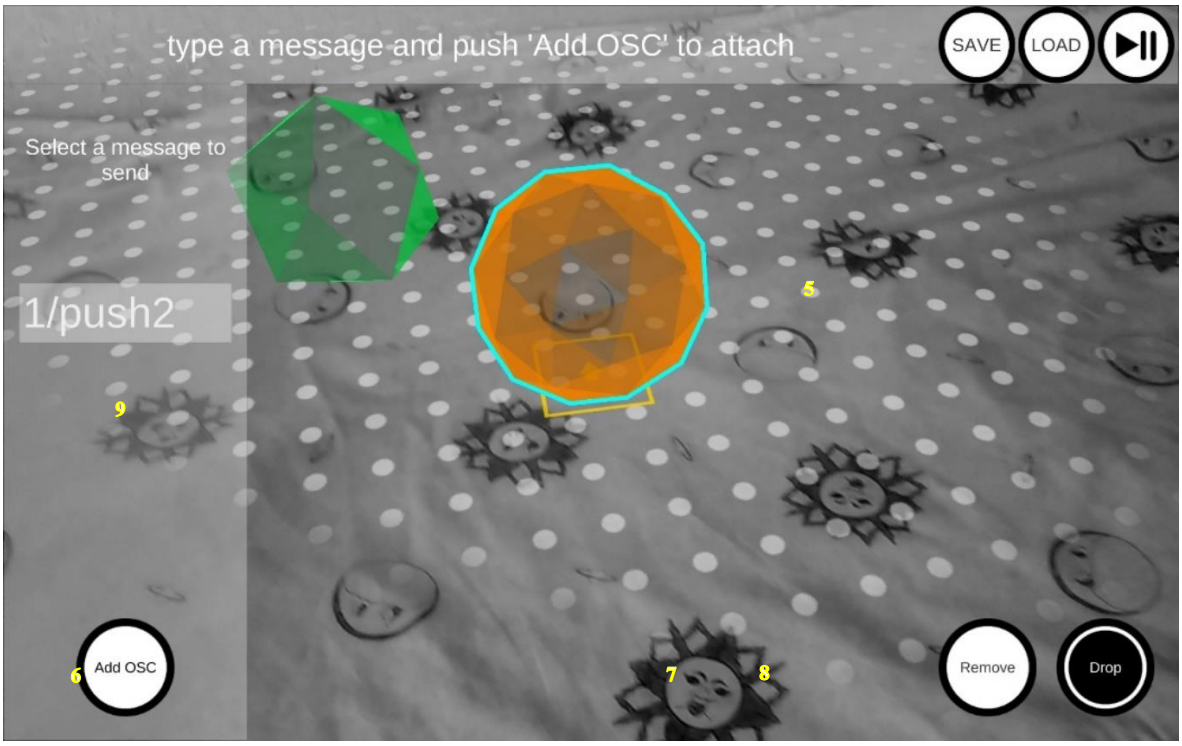

Figura 21. Modo editar con el objeto seleccionado siendo movido.

Por último, habrá un panel en la esquina inferior derecha que será visible únicamente durante el modo Play y que contiene dos cuadros de entrada de texto para introducir una dirección IP y un puerto de salida, y dos botones, uno para inicial y finalizar la conexión OSC y otro para volver al modo Edit que esconderá este panel y mostrará los descritos anteriormente.

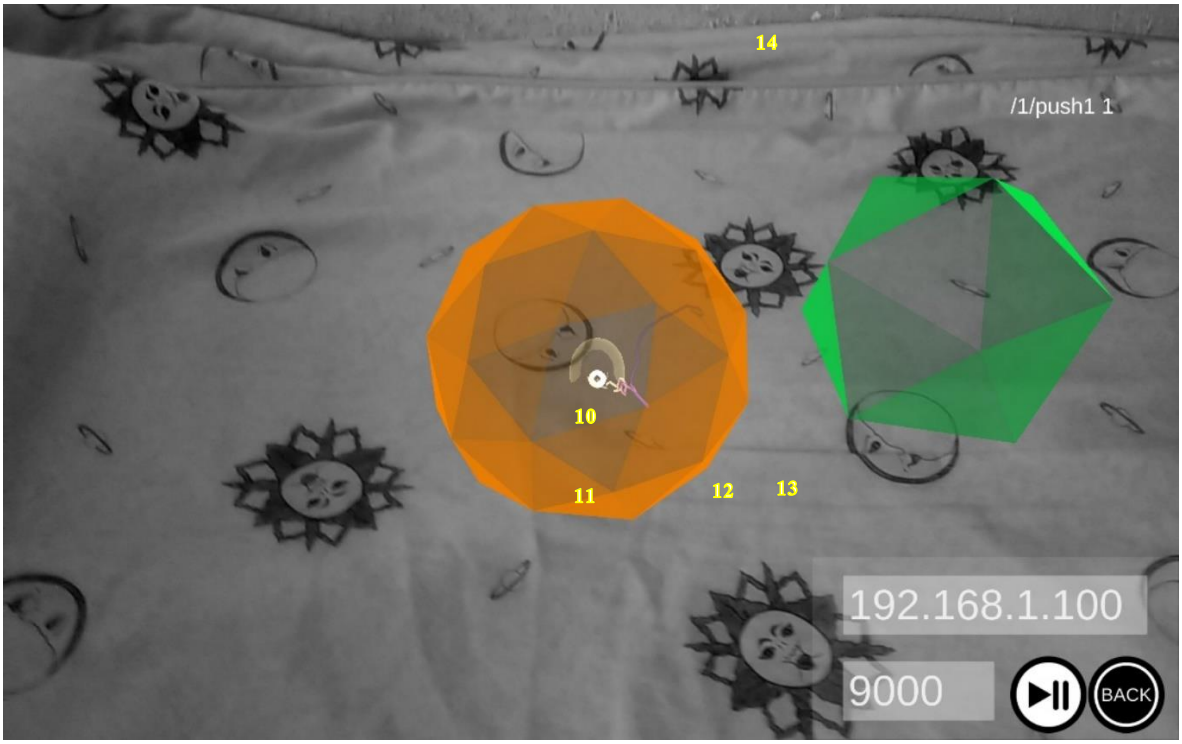

Figura 22. Modo Play

Los elementos numerados en las figuras 20, 21 y 22 son los elementos que acaban de ser descritos en este texto. Por orden nominal:

1. Lista de objetos. La ruta a la carpeta que los contiene se puede definir en el código de ARTapToPlaceObject.
2. Botón de guardado. El apartado que trata la serialización de datos entrará más en profundidad a los métodos que se activan al presionarlo.
3. Botón de carga. Igual que el de guardado.
4. Plano detectado e indicador de posición. Como ya se ha explicado, si el raycast desde el centro de la pantalla intersecciona un plano detectado, el indicador se muestra.
5. Botón Play. Parecido al botón de carga (3) pero además pone en marcha el modo Play.
6. Botón de añadir mensaje. Añade una componente de texto (9) al objeto con el mensaje que llevará al ser instanciado para el modo Play.
7. Botón de borrar. Quita el objeto seleccionado del espacio.
8. Botón de mover/colocar. El botón mover hace que el objeto seleccionado siga al indicador de posición y el de colocar cancela la acción y deja el objeto donde se encontrase.
9. Mensaje OSC. Cuadro de entrada de texto, al presionar el botón de carga (6) se añade como componente al objeto seleccionado.
10. Dirección IP. Determina la IP a la que la Conexión OSC envía los mensajes.
11. Puerto OSC. Determina el puerto a través del cual la conexión OSC envía los mensajes.
12. Botón para niciar/detener conexión OSC.
13. Botón para volver al modo Edit.
14. Último mensaje OSC enviado.

Tras haber seguido todos estos pasos se consigue una aplicación funcional con la que se puede colocar, mover y borrar objetos y añadirles una cadena de texto como mensaje OSC. A partir de aquí se puede continuar paralelamente con la gestión de los mensajes OSC por un lado y con el guardado y la carga de los datos por otro.

4.4 UniOSC y XMLSerializer

Soundcool OSC, la aplicación móvil de Soundcool, se sirve de la librería UniOSC para gestionar el envío de mensajes y las conexiones OSC[4]. Como la primera intención del programa desarrollado es trasladar Soundcool OSC a realidad aumentada con algunas mejoras, también se utilizará esta librería, aunque sólo se hará uso de la clase UniOSCEventDispatcherButton a modo de ejemplo, que actúa como un pad de un secuenciador. A pesar de que el ejemplo del prototipo en términos de OSC será muy simple, es un buen punto de partida para hacer a esta aplicación compatible con cualquier dispositivo OSC, puesto que se podrá introducir cualquier texto como ya se ha venido viendo a lo largo del desarrollo.

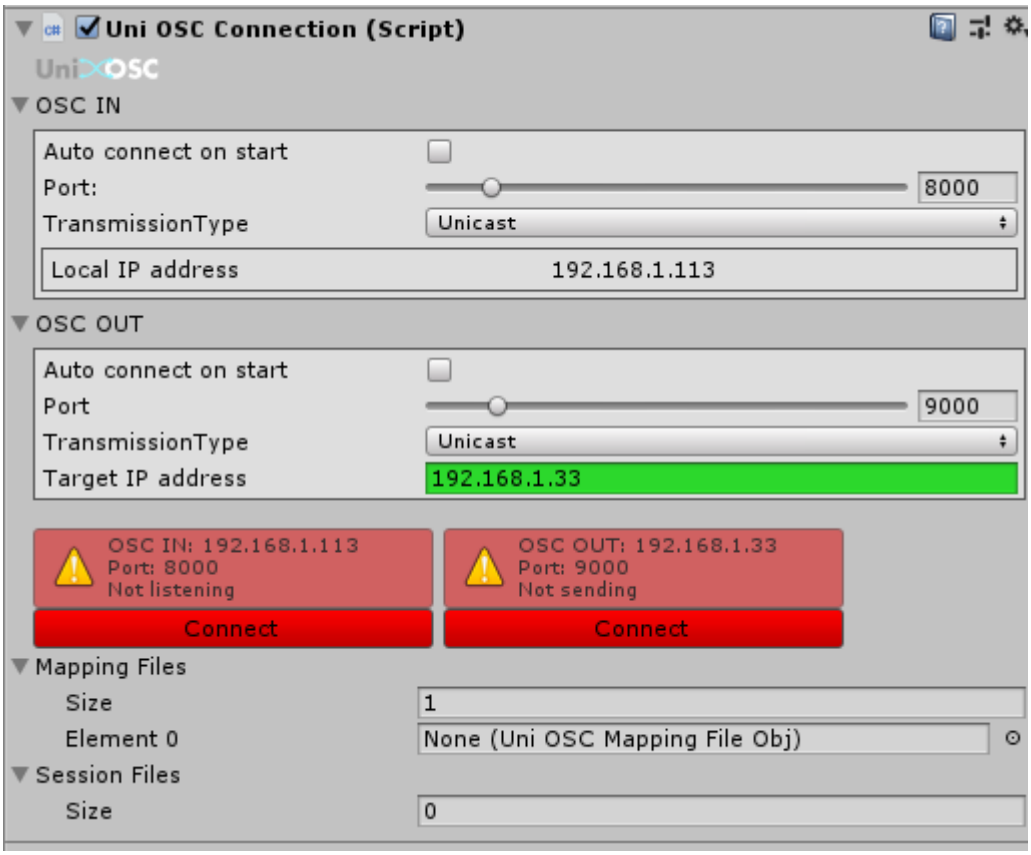

Figura 23. Conexión OSC con UniOSC en el editor de Unity3D

4.4.1 La script TRIGGER

Esta sencilla script implementa la detección de colisiones. Como la bolita que aparece ante la cámara en el modo Play está configurada como colisionador, se ejecutará el código para el objeto con el que ésta colisione. La script se encuentra agregada al objeto 'objmusic', que es como se llama el objeto hijo de los distintos prefabs.

```
void Start()
    {
        button = GetComponentInParent<UniOSCEventDispatcherButton>();
        message = FindObjectOfType<Canvas>().transform.Find("OSCMessage")
                .GetComponent<TextMeshProUGUI>();
        active = GetComponentInParent<Animator>();

private void OnTriggerEnter(Collider collision)
    {
        button.SendOSCMessageDown();
        message.text = button.oscOutAddress +" "+ button.downOSCDataValue;
        active.SetBool("ActiveBool", true);

    }
private void OnTriggerExit(Collider collision)
    {
        button.SendOSCMessageUp();
        message.text = button.oscOutAddress +" "+ button.upOSCDataValue;
        active.SetBool("ActiveBool", false);
    }
```

Al entrar en colisión, el objeto se anima y su mensaje OSC se envía. El valor de los datos equivale a presionar el pad de un secuenciador. Al salir de la colisión la animación termina y se envían los datos equivalentes a soltar el pad.

4.4.2 Scripts para seralización XML

XMLSerializer es una librería del framework .NET de Microsoft que permite guardar y cargar datos en ficheros XML, como su propio nombre indica. Siguiendo el sencillo tutorial para guardar y cargar datos mediante XMLSerializer en Unity3D [5] fue relativamente sencillo implementar código que guardase las coordenadas de los objetos de la escena, su nombre y su mensaje OSC asociado:

```
[XmlRoot("ObjectCollection")]
public class ObjectContainer
{
    [XmlArray("Objects")]
    [XmlArrayItem("Object")]
    public List<ObjectData> objects = new List<ObjectData>();
}
public class ObjectData
{
    [XmlAttribute("name")]
    public string Name;

    [XmlElement("posX")]
    public float posX;

    [XmlElement("posY")]
    public float posY;

    [XmlElement("posZ")]
    public float posZ;

    [XmlElement("oscMessage")]
    public string oscMessage;
}
```

La Script SaveObjectData ha permanecido sin cambio alguno, así como la función de colocar objeto desde el XML, donde simplemente se ha añadido el componente de UniOSC. También se ha añadido la lógica que realizan los botones de guardar y cargar, todo esto en la script ARTapToPlaceObject:

```
void Start()
{
                . . .
         saveButton.onClick.AddListener(delegate {
             SaveObjectData.Save(dataPath, SaveObjectData.objectContainer);
             debugText.text = "SAVED";
         });
         loadButton.onClick.AddListener(delegate {
             SaveObjectData.Load(dataPath);
             debugText.text = "LOADED";
         });
                . . .
}
public static Edit PlaceObject(ObjectData data, string path, Vector3 position,
         Quaternion rotation)
    {
        GameObject prefab = Resources.Load<GameObject>(path + data.Name);
        GameObject go = Instantiate(prefab, position, rotation) as GameObject;
        go.name = prefab.name;

        button = go.AddComponent<UniOSCEventDispatcherButton>();
        button.useExplicitConnection = true;
        button.explicitConnection = oscConnection;
        button.oscOutAddress = data.oscMessage;

        Edit item = go.GetComponent<Edit>() ?? go.AddComponent<Edit>();

        item.objectData = data;
        return item;
}
```

4.4.3 Ruta de datos

Es importante hacer saber al programa el formato de la ruta donde se encontrará y guardará el fichero de datos según en qué dispositivo esté corriendo, ya que ésta cambiará en función de si corre en formato escritorio, iOS o Android [5].

```
private void Awake()
    {
        if (Application.platform == RuntimePlatform.IPhonePlayer)
            dataPath=System.IO.Path.Combine
                (Application.persistentDataPath, "/objects.xml");
        if (Application.platform == RuntimePlatform.Android)
            dataPath = System.IO.Path.Combine
                (Application.persistentDataPath, "objects.xml");
        else
            dataPath = System.IO.Path.Combine
                (Application.dataPath, "Resources/objects.xml");
    }
```

4.4.4 Modo Play

Para pasar al modo Play se añade la lógica del botón de Play en ARTapToPlaceObject. También se determinan otras acciones desde el editor de Unity3D como en el ejemplo de la figura 19 para el propio botón Play y para el de iniciar/detener conexión y volver.

```
void Start()
{
                . . .
         playButton.onClick.AddListener(delegate {
            var objectsInScene = FindObjectsOfType<Edit>();
            foreach (var o in objectsInScene)
            {
                o.gameObject.SetActive(false);
            }
            SaveObjectData.Load(dataPath);
            debugText.text = "PLAY";
            ModeControl.CONTROL_MODE = ControlModes.Play;
        });
        goBackButton.onClick.AddListener(delegate {
            ModeControl.CONTROL_MODE = ControlModes.PlaceObject;
            var objectsInScene = FindObjectsOfType<Edit>();
            foreach (var o in objectsInScene)
            {
                o.gameObject.SetActive(false);
            }
            SetObject();
        });
}
```

Los cuadros de entrada de texto para definir la IP y el puerto a través del cual se enviarán los mensajes OSC se configuran también desde el editor:

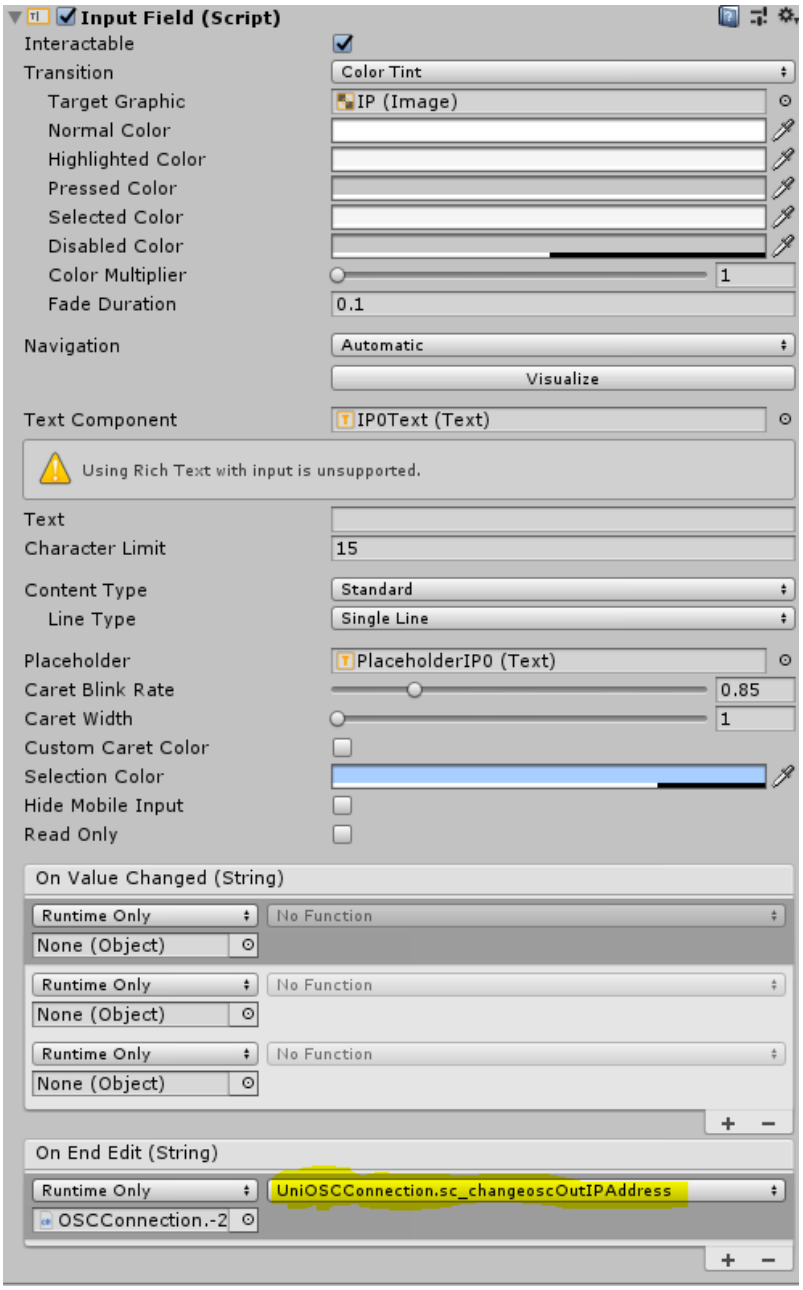

Figura 23. Ejemplo de vista del editor del cuadro de entrada de texto de IP

4.5 Resultados

Como resumen de todo el proceso de desarrollo, los resultados se aproximan muy bien a los objetivos establecidos al comienzo del proyecto. Se ha conseguido crear un prototipo funcional de aplicación en realidad aumentada mediante Unity3D donde se puede elegir uno de los múltiples objetos en la carpeta del proyecto para colocarlo en un espacio virtual. Cualquiera de los objetos puede seleccionarse para modificar su posición o borrarlo y además se le puede asignar una cadena de texto para enviar como mensaje OSC en el modo Play. El envío de OSC, aunque simple, también se ha implementado y, por último, se ha podido además incluir la posibilidad de guardar las coordenadas, el nombre y la cadena de texto de los objetos del espacio virtual en un fichero XML desde el que se pueden volver a cargar estos datos para situar de nuevo los objetos guardados. En resumen, se ha cumplido con las expectativas que se fijaron.

Capítulo 5. Conclusiones y propuestas de trabajo futuro

A lo largo del proceso de desarrollo del proyecto se ha aprendido a trabajar con diversas herramientas utilizadas actualmente en la industria, como pueden ser Unity3D, Git o Teams, lo cual resulta muy útil llegado el momento de pasar del ámbito universitario al laboral. También se han asentado conocimientos y competencias adquiridas durante el Grado, como son la programación, ciertas nociones de diseño 3D, etc. Además, el haber trabajado con realidad aumentada, que es aún una tecnología incipiente, se incluye en la utilización de nuevas tecnologías software y hardware en el contexto de un campo de estudio en constante cambio y evolución. Esta será la dinámica habitual en cualquier empleo de este sector, por lo que es casi necesario no dejar de entrenar las habilidades de ingeniero y estar preparado para lo que supone tener que aprender a manejar un dispositivo de cero o implementar una librería recién lanzada de la que no se tiene ningún conocimiento previo.

5.1 Próximos pasos

5.1.1 Cloud anchors

Como se ha puesto de manifiesto repetidamente a lo largo de este documento, tanto la realidad aumentada como la tecnología como las librerías ARKit, ARCore y ARFoundation son bastante nuevas y se actualizan constantemente. También se ha incidido en los planes de futuro para este proyecto, que son trasladar la aplicación a HoloLens e implementar en móviles y tablets la funcionalidad de poder observar en tiempo real el entorno generado por las gafas a través de la pantalla, de forma parecida a lo que se ve en la figura 6. Para incluirla se requiere el uso de cloud anchors o anclas en la nube. Esto permitirá, como su propio nombre indica, subir a la nube los puntos de referencia detectados por el dispositivo y las coordenadas de los objetos, de manera que otros puedan leerlas. Esta funcionalidad aún no ha sido implementado ni en ARKit ni en ARFoundation, por lo tanto sólo cabe esperar a su implementación. En la figura 21 puede verse una tabla con distintas funcionalidades importantes en un contexto de realidad aumentada, y cuáles se encuentran implementadas en las librerías de Apple, Google y Unity. Puede verse cómo muchas de las que se han utilizado se encuentran en esta lista y han sido comentadas a lo largo del documento y que las anclas en la nube sólo están disponibles en ARCore.

Supported Feature	AR Foundation	Google ARCore SDK for Unity	Unity ARKit Plugin
Plane Detection (Vertical)	✔	✔	✔
Plane Detection (Horizontal)	✔	✔	✔
Feature Point Detection	✔	✔ + Oriented Feature Points	✔
Light Estimation	✔	✔ + Color Correction	✔ + Color Temperature
Hit Testing (Feature point and Plane raycasting)	✔	✔	✔
Image Tracking	in development	✔ (Static Only)	✔
3D Object Tracking	in development		✔
Environment Probes	in development		✔
World Maps	✔		✔
Face Tracking (Pose, Mesh, Blendshapes)	✔		✔ iPhone X + Variants Only
Cloud Anchors	in development	✔	
Editor Remoting	in preview	✔ - Instant Preview	✔ - ARKit Remote
Editor Simulation	in preview		
LWRP support (+ Shader Graph*)	3.3.0 supported	in development	in development
Camera Image API	✔		

Figura 21. Funcionalidades soportadas por ARFoundation, ARCore y ARKit en Unity actualmente

5.1.2 Implementación de OSC

Por limitaciones de tiempo no se ha profundizado excesivamente en la implementación de OSC en la aplicación, sino que únicamente se envían mensajes a través de la clase UniOSCEventDispatcherButton. Si se estudia un poco la documentación podrá observarse que hay distintos tipos de componentes de UniOSC [6], siendo los principales los botones, toggles y sliders. También hay clases propias de Soundcool con las que se puede trabajar y probar su funcionamiento, pero la aplicación Souncool OSC contiene básicamente estos tres elementos.

Para poder añadir estas nuevas clases y funcionalidades OSC al proyecto posiblemente sea necesario crear nuevos prefabs con animaciones específicas, como por ejemplo un slider con el que se pueda interactuar durante el modo Play de forma que se pueda obtener un rango de valores en lugar de un simple ON/OFF, como con los botones o los toggles (figuras 5 y 6). Por el momento y dado que los prefabs que se han utilizado sólo tienen un estado de ON/OFF (animado y sin animar), sólo se les puede implementar de forma sencilla las funcionalidades de botón y toggle. En el proyecto se ha dejado código comentado que puede resultar útil para probar con diferentes tipos de scripts de UniOSC y ver su funcionamiento, pero para la demostración sólo se usarán botones por sencillez.

Capítulo 6. Bibliografía

La totalidad de la bibliografía utilizada se encuentra en internet. A fecha de septiembre de 2019, los enlaces a los documentos que se encuentran a continuación están disponibles y funcionando.

[1] Unity Technologies, The Unity documentation, "Manual" https://docs.unity3d.com/Manual/UnityManual.html [Online]

[2] Unity Technologies, The Unity documentation "Scripting API" https://docs.unity3d.com/ScriptReference/index.html [Online]

[3] Unity Technologies, The Unity Documentation, "About ARFoundation", https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@3.0/manual/index.html [Online]

[4] Monoflow UniOSC, "HowTo", http://uniosc.monoflow.org/howto/ [Online]

[5] Youtube, Omnirift, "How To Save Your Game In Unity with XML | Files Included", https://www.youtube.com/watch?time_continue=897&v=Y8Di-Q6qpU4 [Online]

[6] Soundcool, "Manual de usuario", [Online]

[7] Youtube, The Unity Workbench "Getting Started With ARFoundation in Unity (ARKit, ARCore)", https://www.youtube.com/watch?v=Ml2UakwRxjk [Online]